OpenStack-Ansible Documentation: specs role

Release 0.0.1.dev220

OpenStack-Ansible Contributors

May 27, 2023

CONTENTS

CHAPTER

ONE

SPEC TEMPLATES

1.1 Example Spec - Title of your spec

date
2016-10-13 22:00

tags
used, for, groupings, and, indexing

Update the date of your spec with the date that it was proposed. Add any tags to the spec that may be of use for people to get what its about at a glance.

Provide a synopsis as to why you are creating this spec/blueprint.

Include the URL of your launchpad blueprint:

- https://blueprints.launchpad.net/openstack-ansible/+spec/example

Introduction paragraph — why are we doing anything? A single paragraph of prose that operators can understand. Describe the problem in as much detail as you can.

Some notes about using this template:

- Your spec should be in ReSTructured text, like this template.
- Please wrap text at 79 columns.
- The filename in the git repository should match the launchpad URL, for example a URL of: https://blueprints.launchpad.net/openstack-ansible/+spec/awesome-thing should be named awesome-thing.rst
- Please do not delete any of the sections in this template. If you have nothing to say for a whole section, just write: None or N/A
- For help with syntax, see http://sphinx-doc.org/rest.html
- To test out your formatting, build the docs using tox, or see: http://rst.ninjs.org
- If you would like to provide a diagram with your spec, ascii diagrams are required. http://asciiflow.com/ is a very nice tool to assist with making ascii diagrams. The reason for this is that the tool used to review specs is based purely on plain text. Plain text will allow review to proceed without having to look at additional files which can not be viewed in gerrit. It will also allow inline feedback on the diagram itself.

1.1.1 Problem description

A detailed description of the problem:

- For a new feature this might be use cases. Ensure you are clear about the actors in each use case: End User vs Deployer.
- For a major reworking of something existing it would describe the problems in that feature that are being addressed.

1.1.2 Proposed change

Provide an overview of the changes youd like to see implemented. Here is where you cover the change you propose to make in detail. How do you propose to solve this problem?

Notable changes:

- list out all of the notable changes.

If this is one part of a larger effort make it clear where this piece ends. In other words, whats the scope of this effort?

Alternatives

What, if any, are the alternatives to the changes you are proposing? What other ways could we do this thing? Why arent we using those? This doesnt have to be a full literature review, but it should demonstrate that thought has been put into why the proposed solution is an appropriate one.

Playbook/Role impact

What impact will there be on the playbooks?

Upgrade impact

If this change set concerns any kind of upgrade process, describe how it is supposed to deal with that stuff. For example, if containers are removed and or have their specific purpose changed how do you indend to deal with the eventual upgrade from the prospective of an existing installation? Does this change require documentation to be fully supported or will there be specific tooling that has to be created in order for the upgrade to be completed?

Security impact

Describe any potential security impact on the system. Some of the items to consider include:

- Does this change touch sensitive data such as tokens, keys, or user data?
- Does this change alter a deployed OpenStack API in a way that may impact security, such as a new way to access sensitive information or a new way to login?
- Does this change involve cryptography or hashing?
- Does this change require the use of sudo or any elevated privileges?

- Does this change involve using or parsing user-provided data? This could be directly at the API level or indirectly such as changes to a cache layer.

For more detailed guidance, please see the OpenStack Security Guidelines as a reference (https://wiki.openstack.org/wiki/Security/Guidelines). These guidelines are a work in progress and are designed to help you identify security best practices. For further information, feel free to reach out to the OpenStack Security Group at openstack-security@lists.openstack.org.

Performance impact

Describe any potential performance impact on the system. For example, how is the code executed and does it depend on upstream resources that may be unavailable?

Examples of things to consider here include:

- Adding a new PPA for upgraded packages, who is the maintainer? Does this individual push often? How long has this individual/company maintain specific packages?
- Adding additional pinned packages for use in a python wheel. Does this package change often? Are there tests?

End user impact

How would the end user be impacted by this change? The End User is defined as the users of the deployed cloud?

Deployer impact

How would the deployer be impacted by this change? Discuss things that will affect how OpenStack will be deployed, such as:

- What config options are being added? Should they be more generic than proposed? Are the default values ones which will work well in real deployments?
- Is this a change that takes immediate effect after its merged, or is it something that has to be explicitly enabled?
- If this change is a new binary, how would it be deployed?
- Please state anything that those doing continuous deployment, or those upgrading from the previous release, need to be aware of. Also describe any plans to deprecate configuration values or features. For example, if we change the name of a play, how do we handle deployments before the change landed? Do we have a special case in the code? Do we assume that the operator will recreate containers within the infrastructure of the cloud? Does this effect running instances within the cloud?

Developer impact

How does this change impact future developers working on the ansible playbooks? Discuss things that will affect other developers working on OS-Ansible-Deployment, such as:

- If this spec proposes a new role, how will that role be deployed? Is this a new default role? Does this role have a host impact?

Dependencies

Does this blueprint/spec depend one another blueprint or spec?

- Include specific references to specs and/or blueprints in os-ansible-deployment, or in other projects, that this one either depends on or is related to.
- Is the new requirement due to an upstream change? If so document it and provide references to the change.

1.1.3 Implementation

Assignee(s)

Who is leading the writing of the code? Or is this a blueprint where youre throwing it out there to see who picks it up?

If more than one person is working on the implementation, please designate the primary author and contact.

Primary assignee:
<launchpad-id or None>

Other contributors:
<launchpad-id or None>

Please add **IRC nicknames** where applicable.

Work items

Work items or tasks break the feature up into the things that need to be done to implement it. Those parts might end up being done by different people, but were mostly trying to understand the timeline for implementation.

1.1.4 Testing

Please discuss how the change will be tested. You should be able to answer the following questions:

- Does this change impact how gating is done?
- Can this change be tested on a **per-commit** basis?
- Given the instance size restrictions, as found in OpenStack Infra (8GB Ram, vCPUs <= 8), can the test be run in a resource constrained environment?

- Is this untestable given current limitations (specific hardware / software configurations available)? If so, are there mitigation plans for this change to be tested within 3rd party testing, gate enhancements, etc?
- If the service is not OpenStack specific how can we test the change?

1.1.5 Documentation impact

What is the impact on the docs team of this change? Some changes might require donating resources to the docs team to have the documentation updated. Dont repeat details discussed above, but please reference them here.

1.1.6 References

Please add any useful references here. You are not required to have any reference. Moreover, this specification should still make sense when your references are unavailable. Examples of what you could include are:

- Links to mailing list or IRC discussions
- Links to relevant research, if appropriate
- Related specifications as appropriate
- Anything else you feel it is worthwhile to refer to

CHAPTER

TWO

ANTELOPE SPECIFICATIONS

2.1 Separated Haproxy Service Config

date
2023-01-19 22:00

tags
separated haproxy service config, internal tls

Currently, all haproxy services are configured during execution of haproxy-install.yml playbook. It may cause issues with variables scope or even completely break a service until service role is executed.

These issues may be avoided if the current behavior will be changed and haproxy services will be configured separately at the beginning of each service playbook.

2.1.1 Problem description

Preconfiguring all haproxy services may lead to some issues. There are 2 examples:

1. Variables scope

   Currently, haproxy service definitions stored in inventory/group_vars/haproxy/haproxy.yml refer to variables like `neutron_plugin_type`. It makes a problem because this is neutrons variable. If someone wants to change its value, they will probably set an override in neutron group variables. Its problematic because haproxy is not in neutron group, so all neutrons variables wont have an effect for haproxy role. In order to make haproxy respect this change, variable needs to be defined for all hosts, so both haproxy and neutron will have access to it.

   Additionally, we are currently working on encrypting traffic between haproxy and service backends. Proposed PoC [1] does the same thing as described above. It refers to `glance_backend_https` variable belonging to glance role.

2. Strong dependency between haproxy role and service roles

   Some changes in haproxy service need an immediate reaction from service role. For example: user enables TLS for communication between haproxy and glance backends. To fix that, haproxy role needs to be executed. It will configure glance service to communicate with its backends over TLS, but at this point backends are not ready to handle TLS connections. To fix it, glance role needs to be executed, but it takes time and increases downtime. Removing dependencies like this between roles would make the configuration process more reliable.

   Please note that downtime will still occur. It will start after haproxy service config step and finish after first backend host will be configured. To provide zero-downtime transition to TLS, further

work related to internal TLS project is required.

2.1.2 Proposed change

Add an extra step at the beginning of each service role to configure haproxy service(s) for it. In this case, haproxy services will be configured separately, so nova playbook will configure nova haproxy services, glance playbook will configure its own haproxy services etc. Haproxy playbook will be only responsible for configuring services not related to any openstack role(letsencrypt, ceph-rgw, custom user-defined services etc.)

Alternatives

No alternatives.

Playbook/Role impact

Each playbook will contain an extra step responsible for configuring haproxy service role(s) for it.

Upgrade impact

Some variables will be removed or replaced. Its already covered in release notes.

Security impact

No impact.

Performance impact

No impact.

End user impact

No impact.

Deployer impact

From now on, haproxy services will be configured separately when running service playbooks(like os-nova-install.yml)

Developer impact

No impact.

Dependencies

No dependencies.

2.1.3 Implementation

Assignee(s)

Primary assignee:
Damian Dabrowski <damian@dabrowski.cloud>

Work items

- configure haproxy_server role to support separated service config
- configure service playbooks to include an extra step to configure haproxy service
- solve all corner cases(like dependency between letsencrypt and horizon)

Changes Hierarchy

All changes are available on gerrit under separated-haproxy-service-config tag[2]. It may be hard to understand relations between them so here is a description Changes at the top should be merged first. Horizontal lines split dependency groups(changes in the same group may be merged independently)

- Blueprint for separated haproxy service config https://review.opendev.org/c/openstack/openstack-ansible-specs/+/871187
- [openstack-ansible] Define some temporary vars for haproxy https://review.opendev.org/c/openstack/openstack-ansible/+/872328

- [haproxy_server] Prepare haproxy role for separated haproxy config https://review.opendev.org/c/openstack/openstack-ansible-haproxy_server/+/871188

- [openstack-ansible] Prepare service roles for separated haproxy config https://review.opendev.org/c/openstack/openstack-ansible/+/871189

- [haproxy_server] [DNM] Remove temporary tweaks related to separated haproxy service config https://review.opendev.org/c/openstack/openstack-ansible-haproxy_server/+/871194

2.1.4 Testing

Special attention is required for gating. Merging this change for all roles may be complicated.

2.1.5 Documentation impact

Documentation needs to updated in a few places. Uploaded changes already contain these updates.

2.1.6 References

[1] https://review.opendev.org/c/openstack/openstack-ansible/+/821090

[2] https://review.opendev.org/q/topic:separated-haproxy-service-config

CHAPTER

THREE

ZED SPECIFICATIONS

3.1 Enabling TLS on Internal Communications

date
2022-11-16 21:00

tags
ssl, tls, certificates, https, security

To improve the security of an OpenStack-Ansible deployments all traffic, both internal and external should be encrypted. There is already support for encrypting external traffic from all public endpoints that reside behind haproxy, but this is not the case for all internal traffic.

3.1.1 Problem description

This problem can broadly be split into 3 sections:

- Securing internal communications to the internal haproxy VIP
- Securing internal communications from haproxy to backends
- Securing internal communications between services such as rabbitmq, galera, nova live migration and noVNC

Securing internal communications to the internal haproxy VIP

Support for using TLS on in the internal haproxy VIP is already present in haproxy role and is enabled for the AIO deployment, but not enabled for new or upgrades of existing deployments.

There are no issues with enabling TLS on the internal haproxy VIP for new deployments, but for existing deployments an upgrade process needs to be implemented. The reason an upgrade process is required is because currently if you enabled TLS on the internal haproxy VIP it would cause downtime, until each client is configured to use HTTPS instead of HTTP.

Problems to resolve:

- Haproxy configuration to allow TLS to be enabled without downtime of APIs on existing deployments
- OpenStack-Ansible upgrade process and upgrade scripts to enable TLS without downtime of APIs on existing deployments

Securing internal communications from haproxy to backends

Securing the communications from haproxy to the services backends is as important as securing communication to the internal haproxy VIP.

A large number of the services used with haproxy use UWSGI, meaning once TLS support is added to the UWSGI role there is only configuration to enable TLS and the generation of certificates required for each of the services.

For services that do not use USWGI, such a noVNC Proxy further investigation is required.

As with enabling TLS on the internal haproxy VIP for new deployments, there is no issue with enabling TLS from haproxy to backends, but an upgrade process for existing deployments is required. The reason an upgrade process is required is because if haproxy expects TLS backends, but TLS has not been enabled on the service yet the connection will fail and if you enable TLS on the service the connection will fail as haproxy is not configured for TLS.

Problems to resolve:

- Add TLS support to UWSGI
- Add configuration to role for each service that use UWSGI to enable TLS
- Add configuration to role for remaining services that do not use UWSGI
- Add configuration to OpenStack-Ansible to enable TLS on backend of each service
- OpenStack-Ansible upgrade process and upgrade scripts to enable TLS on backends without downtime of APIs on existing deployments

Securing internal communications between services

Many OpenStack services communicate directly with each other and do not use haproxy, these communications should also be secured. The work to secure these communications is already complete and enabled in the Yoga release of OpenStack-Ansible, for the following services:

- RabbitMQ
- Galera
- Nova live migrations
- noVNC (noVNC to compute nodes).

Problems to resolve:

- Secure the following services:
  - Memcached
  - etcd
  - OVN/OVS
- Are there any services missing from the list that do not go via haproxy that need their communications securing?

3.1.2 Proposed change

Enable TLS on all internal communications.

Internal communications could be encrypted using a self-signed certificate, but as OpenStack-Ansible has support for issuing certificates from a self-signed private certificate authority using the ansible-role-pki, this should be used instead as it both encrypts the data and allows a client to trust the server.

In all cases a user should be able to override the certificates issued by a self-signed private certificate authority, allowing them to provide their own certificate which may have been issued by a publicly trusted certificate authority.

Alternatives

None, internal communications should be protected and TLS is an appropriate and well used solution.

Playbook/Role impact

Roles:

- Support for generating certificates using the ansible-role-pki role will be added to each service
- Configuring to enable/disable TLS will be added

Upgrade impact

Enabling TLS could be performed during or post upgrade.

As discussed in the problem description section, enabling TLS on the internal haproxy VIP and service backends for existing deployments will cause downtime during an upgrade if enabled. The reason it will cause downtime is that for both communications from internal client => internal haproxy VIP (server) and haproxy (client) => openstack service backend (server), both the client and server need to be updated to use TLS at the same time.

To mitigate this issue I propose an intermediate step during an upgrade, where haproxy frontend will accept both HTTP and HTTPS communications. This would be achieved by adding a new TCP frontend to haproxy that accepts both HTTP and HTTPS traffic and redirects to correct frontend for each, and means that openstack clients can carry on using the same well known port and haproxy looks after redirecting them to the correct frontend; HTTP or HTTPS.

To mitigate issues with haproxy<>backend communication, I suggest implementing Separated Haproxy Service Config feature[1] that configures openstack service and its haproxy service in the same playbook.

The other issue to be aware of is that when user wants to use predefined certificate, this certificate will be used on all VIPs, both internal and external. This means that if TLS is enabled on haproxys internal VIP, internal clients must be able to trust the presented certificate if it is the same as the external certificate. This limitation does not apply to: - certbot, which can present a separate certificate on external interfaces. - PKI role which installs different certificates for external and internal VIPs by default

Security impact

This change will encrypt all internal communications, securing any sensitive data being sent, therefore security is improved.

Performance impact

Implementing TLS on all internal communications will lead to a small increase in the processing requirements and latency of servers and clients, but the increased security outweighs these.

End user impact

None, if the deployment is done correctly.

Deployer impact

- Deployers will need to add monitoring of certificate expiry dates and renew is necessary, if a certificates expires connections between services will be dropped.
- This change should have no impact to deployers of new deployments, OpenStack-Ansible will create the certificates, deploy them and configure all services to use them.
- This change will impact existing deployments and an upgrade process will be implemented to help minimise and possibly prevent this.

Developer impact

No impact, other that traffic will be encrypted meaning tools like tcpdump may provide less useful as they will not be able to the see the contents of packets.

Dependencies

None.

3.1.3 Implementation

Assignee(s)

Primary assignee:
Damian Dabrowski <damian@dabrowski.cloud>

Work items

- Enable TLS support to UWSGI role
- Enable TLS backend support to haproxy role
- Add configuration to openstack services that use UWSGI to create TLS certificate and enable TLS on UWSGI
- Add configuration to remaining openstack services that do not use USWGI to enable TLS support
- Add configuration in OpenStack-Ansible to allow TLS for all service to be enabled on both the server and haproxy
- Update documentation on TLS configuration options
- Add documentation for upgrade procedure
- Add script to automate as much as possible of the upgrade

3.1.4 Testing

These changes can be tested using the existing setup, but manual testing of upgrade procedure will be required to make this is does not cause any downtime, as the automated testing only confirms a working upgrade at the end.

3.1.5 Documentation impact

As this change will add extra configuration options these will need to be documented.

The upgrade procedure for existing deployments will also have be documented, as if this functionality is not deployed correctly it may cause system distribution.

3.1.6 References

[1] https://specs.openstack.org/openstack/openstack-ansible-specs/specs/antelope/separated-haproxy-service-config.html

CHAPTER

FOUR

XENA SPECIFICATIONS

4.1 Protecting OpenStack Plaintext Secrets Automation

date
2021-04-22 22:00

tags
protecting plaintext configs, oslo.config, castellan, vault, security

OpenStack services require sensitive data to be in configuration files. These are values for various configuration options such as `password`, `transport_url`, `connection`, and so on. The configuration files for OpenStack services are just plaintext files. Because of this, even with proper file permissions set on these files, secret data is kept there without any protection at present.

A specification on protecting plaintext secrets in OpenStack was created by **Raildo Mascena de Sousa Filho** [1]. Also, the osloconfig and castellan libraries support the following scheme of configuration files processing: oslo-config can read configuration options and their values with help of castellan, which is able to read that data from a protected key management solution, such as HashiCorp Vault.

The proof of concept for this scheme was made by **Moisés Guimarães de Medeiros** in his code [2]. So, the problem of securing sensitive options in OpenStack plaintext configuration is almost solved, with the exception of the automation of such a secure configuration.

4.1.1 Problem description

At present, `oslo.config` allows us to specify the `config_source` in the `DEFAULT` section of a configuration file, and to use `castellan` driver to read configuration options from the appropriate configuration file section:

```
[DEFAULT]
config_source = secrets

[secrets]
driver=castellan
config_file=castellan.conf
mapping_file=mapping.conf
```

The `castellan.conf` and `mapping.conf` configuration files includes information on how to read configuration options values from a secret store.

`castellan.conf` example:

```
[key_manager]
backend=vault

[vault]
kv_mountpoint=kv
vault_url='https://vault.enterprise.local:8200'
use_ssl=True
```

`mapping.conf` example:

```
[oslo_messaging_notifications]
transport_url=6d1c6b6bd925418eb3c99523750bc4be

[database]
connection=20921387a863462dae4db253198156ec
```

where the values of `transport_url` and `connection` parameters are IDs of appropriate records in the HashiCorp Vault.

The problem is that a system administrator needs to insert appropriate values to the HashiCorp Vault or another secret storage, and then reconfigure the OpenStack services either manually or with some preferred automation tools. There is no existing solution today for starting OpenStack with secured configuration files from the very beginning.

With all of this in mind, we are proposing that system administrators should be allowed to install OpenStack with secured configuration files.

4.1.2 Proposed change

So, there are two parts of protecting OpenStack plaintext configuration files:

- installation of HashiCorp Vault, in case it is not installed;
- proper configuration of all OpenStack services with plaintext configs.

This spec proposes to make the following additions to `openstack-ansible` [3]:

- add playbook for HashiCorp Vault installation;
- add additional tasks to OpenStack services playbooks;
- add additional parameters for system administrator to choose if OpenStack is going to be installed with protected plaintext configs or not.

Such additional parameters are:

- `vault_hosts` optional parameter which indicates hosts where HasiCorp Vault should be installed. HashiCorp Vault should have high availability installation as other OpenStack services have;
- `protected_configs` and `protected_configs_castellan_conf`, should be added to `openstack_user_config.yml` file. `protected_configs` parameter should take `true` or `false` values, and the default value should be `false`.

If the system administrator sets the value to `true`, then additional steps are performed in playbooks to address the below:

- check for required Python libraries (such as `castellan`) and install i them if needed;

- add appropriate values to secret store;
- prepare service configuration file without sensitive data but with `config_source` option and secrets section;
- add `castellan.conf` to service configuration directory;
- add `mapping.conf` to service configuration directory.

This change does not require any changes to `oslo.config` or `castellan`, since everything is already supported at present.

4.1.3 Alternatives

The protection of plaintext secrets can be implemented by using a separate Ansible playbook just for the services reconfiguration, after installing OpenStack with unsecured configuration files with `opentack-ansible`.

In that case, the list of services to secure should be given to the playbook, and the tasks of such a playbook should do almost the same steps as described above:

- install and initialize HashiCorp Vault;
- check for required Python libraries (such as `castellan`) and install them if needed;
- add appropriate values to secret store;
- prepare service configuration file without sensitive data but with `config_source` option and secrets section;
- add `castellan.conf` to service configuration directory;
- add `mapping.conf` to service configuration directory;
- restart OpenStack service.

Playbook/Role impact

Additional role on HashiCorp Vault installation should be added.

Additional tasks for proper configuration of OpenStack services should be added to playbooks.

Upgrade impact

No impact.

Security impact

Sensitive data is going to be removed from plaintext config files so security will be improved with this change.

Performance impact

No impact.

End user impact

No impact.

Deployer impact

No impact in case deployer does not want to protect plaintext configs. Otherwise deployer will be able to configure additional options to remove sensitive data from plaintext configs as described above.

Developer impact

No impact.

Dependencies

Here is initial blueprint regarding protection of sensitive data inside plaintext configs:

https://blueprints.launchpad.net/oslo.config/+spec/protect-plaintext-passwords

4.1.4 Implementation

Assignee(s)

Primary assignee:
yeremko <Alexander.Yeremko@walmart.com>

Other contributors:
<Misha.Vorona@walmart.com>

Work items

- Implement changes to `openstack-ansible` [3]
- Implement changes to `openstack-ansible-tests` [4]
- Update documentation.

4.1.5 Testing

Additional tests in `openstack-ansible-tests` [4] will be required to cover the added functionality.

4.1.6 Documentation impact

The documentation will need to be updated to illustrate how to protect plaintext configs and work with them further.

4.1.7 References

[1] https://specs.openstack.org/openstack/oslo-specs/specs/stein/secret-management-store.html

[2] https://github.com/moisesguimaraes/ep19

[3] https://opendev.org/openstack/openstack-ansible

[4] https://opendev.org/openstack/openstack-ansible-tests

[5] https://www.vaultproject.io/

CHAPTER

FIVE

WALLABY SPECIFICATIONS

5.1 SSL Root Certificate Authority

date
2020-10-19 14:00

tags
ssl, haproxy, nova, galera, infra

Blueprint on Launchpad:

- https://blueprints.launchpad.net/openstack-ansible/+spec/ssl-root-ca

Having SSL encryption is a vital part of every service in the modern world. OpenStack-Ansible already provides deployers options how to cover public and internal endpoints with SSL certificates and allows the generation and use self-signed certificates. However we can make these certificates verified by usage of the root CA, which will be distributed across all containers and services. This will increase security as users will be alerted in case of the certificate verification failure, since certificate verification will be enabled.

5.1.1 Problem description

At the moment several openstack-ansible roles do create self-signed certificates their own way which does not provide any consistency and is pretty hard to maintain. Moreover, these certificates are self-signed ones, which means that certificate verification has to be disabled for such certificates. So in case of the certificate impersonation you wont be alerted, and encrypted data might be not secure anymore.

Futhermore, for mysql SSL usage, it is required to place Root CA on the client containers for mysqlclient to reach the database, while at the moment we have only server side encryption covered, but Root CA distribution remains the responsibility of the deployer. Another good example is nova, where in order to disable tunneling for the live migrations and to use block device migrations, we should be securing the connection with mutually verifiable certificates.

To resolve all of the problems above we need:

- Root CA certificate (and corresponding key which may not be available)
- An intermediate signing certificate and key

5.1.2 Proposed change

In order to resolve issue we need to create a root certificate authority on the deploy host, which will be used for the futher creation of the certificates.

We need this for (but not limited to):

- Creating a self-signed certificate for HAProxy in CI
- Creating ssh signed certs to replace ssh keys in nova and keystone roles
- To use TLS for live migration
- To use TLS for galera and other infrastructure services
- To use TLS for connection between HAProxy and uwsgi

Implementation

Create role in separate repo which would consist from several parts which would be included wherever needed:

> # Create/rotate or verify existing provided CA on the deploy host. Included in openstack_hosts, to distribute CA to certificate storage We should implement proper Root CA rotation mechanism including usage of the OldWithOld, OldWithNew, and NewWithOld. # Create/rotate or verify existing key and certificate, which would be also stored on the deploy host. Will be included in required roles during their runtime Each instance of each component uses a unique certificate. # Decide what we call the internal VIP if its needed to verify the dns name against subject name in an SSL certificate. Consider having support for SAN certificates that will include several names or IPs. # Create a signed ssh key which can be validated against the CA certificate to avoid needing to distribute all keys to all hosts. # Role should have a specific key to rotate self-signed certificates and root CA when its asked to do so. Its up to deployer to keep track on the certificates exipration date. Since they are placed on the deploy host, it should be pretty straightforward to configure monitoring tool for that.

5.1.3 Roles/service impact

For all hosts/containers

The Root CA is installed into the system trust store Consider pointing REQUESTS_CA_BUNDLE to the system trust store rather than certificate bundle.

For HAproxy

We will have the following user scenarios:

- The user supplies their own cert and key and points variables to the files
- Letsencrypt creates the certificate
- A self signed certificate and key is created at deploy time by the OSA role
- Disable SSL certificate usage

Note: Self signed cert is required to bootstrap LE for the first run

For Galera (or other infrastructure service)

We will have the following user scenarios:

- The user supplies their own cert and key and points variables to the files
- A self signed certificate and key is created on the deploy host at deploy time by the OSA role (stored in a well known location on the deploy host). The existing ansible roles pick these up
- Disable SSL certificate usage

For service components such as Nova and Octavia which can use TLS

We will have the following user scenarios:

- The user supplies their own cert and key and points variables to the files
- A self signed certificate and key is created on the deploy host at deploy time by the OSA role (stored in a well known location on the deploy host) The existing ansible roles pick these up
- Disable SSL certificate usage

To replace ssh keys

- Signed ssh certificates are created on the deploy host and copied to the relevant .ssh user directories. The signing CA is installed into the relevant ssh_config file in /etc/
- The deployer may already be using signed ssh keys for access to hosts so any implementation should work alongside existing configuration. It may be necessary to investigate supporting more than one trusted CA in the ssh_config file.

Upgrade impact

By default self-singed alternatives will be used for all types of services. In case deployer would like to omit that, he will need to explicitly disable that behaviour before upgrade. No other impact for upgrades is planned at the moment.

Security impact

Realization of this blueprint should make systems more secure because of the SSL usage for most of the interactions and enabling SSL verification even for the self-singed sertificates.

Performance impact

This may decrease performance slightly because SSL encryption requires several extra CPU cycles and TLS handshake to be performed, however this drawdown can be neglected.

End user impact

No end user impact

Deployer impact

The deployer will be able to provide:

- Root CA
- An intermediate cert and key

Also we should leave possible to disable SSL usage for services.

Developer impact

This blueprint will ease maintenance of the roles, because all SSL-specific parts will be moved to the standalone role. So in case of the need to change specific task it would be done in a single place rather than in each role.

5.1.4 Implementation

Assignee(s)

Primary assignee:
noonedeadpunk

Other contributors:
jrosser

Work items

TBD

5.1.5 Testing

We will enable self-singed certificates usage in CI

5.1.6 Documentation impact

Documentation of the added options and architecture should be added at the end of the day, as well as release notes.

5.1.7 References

- Etherpad discussion: https://etherpad.opendev.org/p/osa-certificates-refactor
- Root CA Key Update https://tools.ietf.org/html/rfc4210#section-4.4

CHAPTER

SIX

ROCKY SPECIFICATIONS

6.1 Use nginx as centralized reverse proxy for API services

date
2017-12-02 00:00

tags
nginx, load balancing, wsgi, API

In the previous cycle, most OpenStack API services provided a WSGI application which is being serviced through uWSGI. The aim of this spec is to outline a plan for taking advantage of the greater flexibility uWSGI provides and making use of nginx as a centralized reverse proxy.

https://blueprints.launchpad.net/openstack-ansible/+spec/centralized-nginx

6.1.1 Problem description

Within most roles, OpenStack API services are now deployed and served through uWSGI. Some however, also include the installation of a web server proxy in front of that. The web server is currently assumed to be installed on the host in isolation and managed through the installing role. This causes issues with converged installations, such as an all bare metal scenario. SSL encryption of service requests is also currently difficult because of this inconsistency between individual service deployments. For the most part, SSL termination in OpenStack-Ansible deployments is expected to be handled at the load balancer but deployers may require additional controls over the handling of encrypted traffic.

6.1.2 Proposed change

Remove the installation of nginx from any OpenStack role that includes it today and instead deploy it separately as a shared reverse proxy with individual sites configured for each OpenStack API service. The OpenStack roles will only need to provide site configurations for an existing nginx installation.

For management of the uWSGI backends that nginx will be proxying, install a uWSGI FastRouter alongside nginx. The FastRouter will create a shared socket that nginx can connect to, and a subscription server for the applications to subscribe to and automatically provide load balancing for.

Alternatives

nginx could instead be deployed alongside each OpenStack service, but that could have an effect on performance due to the additional processes running on the host and wouldnt address the duplication of tasks and conflicts caused when multiple roles are attempting to manage the same web server on the same physical host or container.

The load balancer backends could use uWSGI http/https directly, most projects recommended a dedicated web server however. With the proposed change, there is a clear separation between the handling of http requests and running of Python code. Load balancer configurations can also be simplified since all OpenStack API backends would use the same set of nginx hosts, instead of requiring knowledge of individual containers. And for scaling purposes, a new or replaced API service installation will only need to subscribe to the FastRouter to be included in the pool.

Playbook/Role impact

A new playbook will need to be created to install nginx and a uWSGI FastRouter. An existing nginx role from galaxy would be preferred, but one may need to be created or contributed to if there is not one that supports all of the same operating systems that OpenStack-Ansible does or that does not provide sufficient configuration options for our needs.

Each existing OpenStack role that deploys a web server will need to have those tasks removed and replaced with a task to configure and enable an nginx site, delegated to any hosts running nginx. The uWSGI configuration within those roles will also need to be updated with options to subscribe to an existing FastRouter.

The HAProxy server role may require more flexibility, such as allowing SSL passthrough and just-in-time configuration of services and backends.

Upgrade impact

Since load balancer backends will likely be changing, the load balancer will need to initially contain both the existing backends and the centralized nginx backends to limit downtime during upgrades.

Security impact

The proposed change should increase the security posture of OpenStack-Ansible since it will allow deployers to have more flexibility over where SSL is terminated, including preventing traffic from leaving a host once it is unencrypted.

Performance impact

nginx is a high performance web server and reverse proxy. There may be some host performance increase with less web servers running on a single controller host. Deployment times may also be improved by removing the duplicated tasks of installing and configuring a web server within multiple roles.

End user impact

N/A

Deployer impact

Deployers will need to be aware of the deployment architecture changes this change includes. Additional options will be available to deployers to configure a shared nginx instance, uWSGI FastRouter, and the distribution of SSL certificates. Deployers must also be made aware of any upgrade impact this change may have.

Developer impact

Future OpenStack roles will need to include the patterns established by this change. Tests will require additional Ansible roles, playbooks, and variables.

Dependencies

The HAProxy server role may require some changes and increased configuration flexibility.

The keystone role currently conditionally installs Apache when used as an identity provider or service provider for an external identity provider. This will need to be investigated further to ensure that nginx can provide the same level of support for those roles, including gated test scenarios.

The horizon role also currently installs Apache and uses it to serve static content. When running on a container, that static content may need to move to a mount to remain accessible to an external web server.

6.1.3 Implementation

Assignee(s)

Primary assignee:
jimmy-mccrory (jmccrory)

Other contributors:
<launchpad-id or None>

Work items

- Evaluate existing nginx roles in galaxy
- Develop new nginx role if necessary
- Develop playbook for deploying nginx and uWSGI FastRouter
- Adapt HAProxy role
- Evaluate bind mounting of files statically served by web server
- Update OpenStack roles to create nginx site configuration and subscribe to FastRouter for API services

6.1.4 Testing

Individual roles and the integrated repo will test the changes involved as they are implemented.

6.1.5 Documentation impact

The changes to the deployment architecture and any additional options for configuring nginx, uWSGI FastRouter, HAProxy, and SSL will need to be documented.

The impacts to upgrades and steps to minimize, and hopefully avoid, API downtime will also need to be documented.

6.1.6 References

https://uwsgi-docs.readthedocs.io/en/latest/Fastrouter.html

6.2 Integration of Masakari with OpenStack-Ansible

date
2018-03-22 14:00

tags
openstack, masakari, masakari-monitors

Blueprint on Launchpad:

- https://blueprints.launchpad.net/openstack-ansible/+spec/masakari-ansible-plugin

Masakari provides Virtual Machine High Availability (VMHA) service for OpenStack clouds by automatically recovering the KVM-based Virtual Machine(VM)s from failure events such as VM process down, provisioning process down, and nova-compute host failure. It also provides API service for managing and controlling the automated rescue mechanism. The Masakari service consists of the following components:

- masakari-api: An OpenStack-native REST API that processes API requests by sending them to the `masakari-engine` over *Remote Procedure Call (RPC)*.
- masakari-engine: Processes the notifications received from `masakari-api` by execcuting the recovery workflow in asynchronus way.
- masakari-monitors: Monitors for Masakari provides Virtual Machine High Availability (VMHA) service for OpenStack clouds by automatically detecting the failure events such as VM process down, provisioning process down, and nova-compute host failure. If it detects the events, it sends notifications to the masakari-api.

This spec outlines the steps required to integrate Masakari with OpenStack-Ansible.

6.2.1 Problem description

Masakari provides Instances High Availability Service for OpenStack clouds by automatically recovering failed Instances. However, it needs to be installed manually with OpenStack-Ansible. No role exists to deploy it as other services are deployed.

6.2.2 Proposed change

The proposed changes would include:

- Import a proof of concept role for Masakari from https://github.com/NirajSingh90/openstack-ansible-os_masakari to `openstack-ansible-os_masakari`
- Follow the usual path described in the developer documentation.

Alternatives

There are no alternatives.

Playbook/Role impact

This is a new feature added into OpenStack-Ansible. No role currently exists. Therefore, new role, *openstack-ansible-os_masakari* needs to be written from scratch.

Upgrade impact

No upgrade impact since this would be the first implementation of the proposed change.

Security impact

No security impact.

Performance impact

No performance impact.

End user impact

End user will be able to use masakari as a service within OpenStack-Ansible.

Deployer impact

Deployers will need to enable Masakari deployments if they choose to use this. Masakari will not be deployed by default.

Developer impact

No impact.

Dependencies

By employing a combination of Corosync and Pacemaker, OpenStack Masakari creates a cluster of servers, detecting and reporting failure of hosts in the cluster. So masakari is dependent on Corosync and Pacemaker.

We will reuse an external role for corosync and pacemaker to not re-invent the wheel, like the one found in https://github.com/leucos/ansible-pacemaker-corosync .

6.2.3 Implementation

Assignee(s)

Primary assignee:
Niraj Singh (IRC: niraj_singh)

Work items

Masakari is not available as a service for OpenStack-Ansible. No role already exists. A new role will be developed from scratch in compliance with the standards set by the community. It will be added under https://github.com/openstack/openstack-ansible-os_masakari

Note: Masakari role will install below services: masakari-api masakari-engine masakari-processmonitor masakari-hostmonitor masakari-instancemonitor

masakari-processmonitor, masakari-hostmonitor and masakari-instancemonitor will be installed only on nova-compute nodes

6.2.4 Testing

Tests will be developed to ensure that deployment of Masakari works. Masakari doesnt have tempest tests therefore we will start by testing the API responses codes. Masakari-monitor and Masakari-engine services tests will be added in future using third party CI tests.

6.2.5 Documentation impact

As this would be new feature added to OpenStack-Ansible, it needs to be documented, explaining all the configuration parameters.

6.2.6 References

Masakari Overview

- https://wiki.openstack.org/wiki/Masakari

Masakari developer/operator documentation

- https://docs.openstack.org/masakari/latest

6.3 Install OpenStack services from distribution packages

date
2018-03-27 00:00

tags
roles, deployment

Blueprint on Launchpad

- https://blueprints.launchpad.net/openstack-ansible/+spec/openstack-distribution-packages

This spec outlines the work required to enable the OpenStack-Ansible roles to install the OpenStack services using the distribution packages from the distribution Cloud repositories.

6.3.1 Problem description

OpenStack-Ansible installs the OpenStack services from the source. Whilst this is great in terms of flexibility, it creates some problems such as:

- Long deployment times since wheel packages need to be built and distributed.
- Unsupported installations by distributions. The versions of OpenStack services built from source do not necessarily match what distributions test together as part of their integration and verification process so its hard for them to provide support for such installations. As a result of which, operators have limited options when seeking technical support for their deployments.

6.3.2 Proposed change

Add an additional installation method to all the OpenStack-Ansible roles in which the services will be installed using the packages provided by the distributions themselves. The default installation method will not change.

Alternatives

N/A

Playbook/Role impact

All the OpenStack Ansible roles which install OpenStack services (os_*) will be impacted by the proposed change. A new variable will be made available on per-role basis to allow deployers to select the preferred installation method.

Switching from one installation method to the other will not be supported. This can be clarified on the Deployers documentation and also explicitly detected and prevented in the Ansible playbooks possibly by storing a local fact on the host to denote the installation method and checking it during upgrades.

Upgrade impact

Upgrades should not be impacted since the default installation method will not change.

Security impact

The security of the overall installation will not change since distributions normally backport security fixes which are already present in the upstream packages so both installations methods will offer the same level of security reassurances.

Performance impact

The overall performance of the deployment will likely be improved since the distribution packages normally have their default settings tweaked and optimized to match each distributions environment and needs.

End user impact

N/A

Deployer impact

The benefit of this new method for deployers is twofold:

- Use supported packages by distributions and provide feedback back to them. This benefits both distributions and operators since both ends use packages which have passed integration and functional testing before being released.
- Shorten deployment times since distribution packages are used instead of building new ones from source.

Developer impact

N/A

Dependencies

N/A

6.3.3 Implementation

Assignee(s)

Primary assignee:
Markos Chandras (hwoarang)

Work items

The following work items are the same across all impacted roles

- Move existing installation tasks to a new file (`${role}_install_source.yml`)
- Create new file (`${role}_install_distro.yml`) with a set of tasks for distribution installations if necessary.
- Add new variable to allow deployers to select installation method (`${role}_install_method`)
- Dynamically include the appropriate installation file based on the variables value

6.3.4 Testing

Since the default installation method does not change, no new tests are required. However, developers may choose to add new jobs on per distribution basis to test the new installation method.

6.3.5 Documentation impact

Documentation needs to be modified to explain how to use the `distribution` installation method.

6.3.6 References

N/A

6.4 Refactoring OSA inventory

date
2018-04-12 22:00

tags
osa, inventory

The inventory as it stands today has been growing in complexity and has only grown organically since its first implementation in icehouse. Given that Ansible has changed a lot and has added capabilities which were not available in those early versions, it is time to take a step back and look at how it can be re-worked to reduce technical debt and make it easier to maintain.

6.4.1 Problem description

The current OpenStack-Ansible inventory provides the following features:

- Assignment of hosts into groups
- Generating the group structure
- Assigning host variables
- Generating container inventory_hostnames
- Assigning and tracking container IPs based on cidr_networks, reserved IPs, and already allocated IPs.

All these features are included into a single dynamic inventory script, because at the time of its creation, only one inventory was allowed at a time in an ansible cli call.

The dynamic inventory shipped by OSA is core of the functionality of OpenStack-Ansible, yet it is not well understood, neither by the core maintainers nor by new contributors.

As a result, the inventory has grown organically, both in code and in memory usage (changes in the way we deploy things, adding new groups, adding edge cases), and has not seen much maintenance to reduce its scope or the technical debt.

At this point, due to a lack of tests and the complexity of the code, it is difficult to work on without causing hidden breakages which are often only found months later. Adding tests is unrealisticly hard for this legacy code.

The problems can therefore be summarized in a few points:

- The inventory needs to be cleaned up of unnecessary groups and assignments, but it is difficult to clean up effectively without causing hidden breakages.
- We have to carry code in openstack-ansible that is not actively maintained
- We have to execute code thats not actively audited, while it would be technically possible to avoid the execution of code with very few limitations for the end-user.
- Introducing tests to verify regressions was attempted during the Newton, Ocata and Pike development cycles - but that has done nothing more than increase the code complexity and has done nothing to improve the reliability.

6.4.2 Proposed change

Now that we are using Ansible 2.4, we can:

- Stack inventories together, and therefore we can split inventories into smaller inventories if necessary
- Import, and convert inventories to a more readable format.

What I am proposing is to use static files for inventory. It is easier for people to edit the inventory, and review it. Its easier to manipulate, and doesnt require our code to run or edit it.

Host vars, group vars, and inventory structure would be static files, and slimmed down to the minimum.

Here are two example of slimming down (hosts vars, and inventory):

- For me, the features to track proper IP assignment is the scope of a CMDB/IPAM. We shouldnt reinvent the wheel there. Instead this should be spun out of the inventory. People should either:
  - use the old inventory to keep the same features, but we add a warning that the code is deprecated
  - provide their own IP addresses in a static file
  - provide their own dynamic inventory script or use a lookup to fetch data from their IPAM.

  With the generation of IPs outside the scope of the inventory, we could simplify the dynamic inventory further.
- For me, the groups like haproxy, haproxy_all, haproxy_hosts or haproxy_containers are all confusing. Some are used interchangeably, which led to bugs. The proliferation of groups is only due to our inventory. These can all be consolidated into a single group, by changing the playbooks and roles. This is not only restricted to haproxy, and this pattern of group reduction should be extended to all our inventory.

So, at first we need to keep the same configuration style (conf.d/env.d/openstack_user_config). The generated json would then go through a script to generate and clean the static files.

That script would be part of the deploy and upgrade process.

Later, we could re-think the conf.d/env.d/openstack_user_config, or keep it the same but completely change the underlying code. That wouldnt be a problems, because it could be done on the side, as a different inventory system. We would have, on the way, documented the input and outputs of the inventory, which could then be used for building test cases.

Alternatives

Do nothing

Playbook/Role impact

Removing references to old inventory data like old groups. Use lookups or ansible_facts better to reduce the amount of hostvars.

Upgrade impact

Because our inventories are already in a bad state, we already have hosts in the wrong groups.

Upgrade would need to run the tool to migrate the groups to the new groups presented in the playbooks.

Security impact

By ultimately shipping less code, we would marginally improve our security.

Performance impact

- Moving from dynamic to static file with the same format doesnt change performance
- Moving from static json to static yaml may or may not improve performance in your deployment by reducing memory usage. It fully depends on the inventory. Large inventories are more likely to lose performance by switching to yaml for the same input.
- Cleaning up the inventory have a positive performance impact.

End user impact

The end users will not notice any change.

Deployer impact

The deployer will have a different user configuration to deal with (static files)

Hopefully it shouldnt be too hard to understand for an existing openstack-ansible user, or an experienced ansible user.

Developer impact

No change for the development of roles or playbooks.

At the same time we are removing technical debt, we are adding new technical debt by adding these new tools.

With the hope this tools would be easier to understand, read, review, and having more tests, it would overall reduce risks for the project.

Dependencies

None

6.4.3 Implementation

Assignee(s)

Primary assignee:
evrardjp

Other contributors:
None for now.

Work items

Use static files is not without downsides: We are losing some key features if we just use a static inventory which is created by the user, like the dynamic hostname generation, the dynamic IP allocations.

So I propose the following path:

1. We list the groups required for a successful ansible deploy, and document those in the reference guide.

   Positive improvements:

   - For deployers that dont want to use our inventory, we would now have an explicit contract of what they should do to run openstack-ansible with their own inventory groups

   Drawbacks:

   - All changes in groups now needs proper documentation
   - Thats not enough to come with your own inventory

2. Keep the conf.d/env.d, and dynamic inventory script for now. We use it for generating a json that stays static during the lifecycle of the cloud, or until re-generated manually. The env.d/conf.d/openstack_user_config.yml are used as input for this one-off run of the dynamic inventory.

   To make sure deployers dont misunderstand the static json file or confuse it with the current openstack_inventory.json, we should move the current files to a cache folder, and generated the static inventory into a `inventory` folder.

   Positive improvements:

   - No hidden failures, the generation of the inventory becomes a part of the deploy. We can add health checks easily.
   - Our code run only once, during the generation. Therefore we are not vulnerable to issues appearing when running multiple ansible simulatenously, or other side effects.
   - We keep the container name generation, provider networks, and IP assignments for free.

   Drawbacks:

   - Edition of static file will not be in sync with conf.d/env.d, but that was already the case with a manual change to openstack_inventory.json

- The inventory_manage script becomes useless

3. We provide default child mapping: we create the x_all groups in an easy to read .ini file in the openstack-ansible repo.

   Positive improvements:

   - All our users with their own inventory wont have to create EXACTLY the same code to do child group mapping. Sharing is caring.
   - We would cary a lot of empty groups, and maybe people dont need them.
   - The mapping could then be used to partially replace the documentation of step 1, and will fully replace the step 1 documentation when the groups will be cleaned in the playbooks and roles.

4. We export the host vars into a static files inside the userspace inventory folder.

   Positive improvements:

   - Having static yaml files will make it easy to see repetitions, and things that can move to group vars

   Drawbacks:

   - More static files to maintain by the deployer. If we change a host var, we could change the inventory and it was applied everywhere. It would not be the case anymore.

5. We write a tool manipulating the inventory json. By default, that tool would:

   - discard all the groups that arent listed in the reference guide
   - discard all the _all groups from the inventory, as they would not be required in the json anymore (handled at a previous step)
   - discard all the host variables (handled at a previous step)
   - discard groups that can be generated from facts/host variables, like all_containers (using group_by would provide the same result).

   Positive improvements:

   - The inventory would be lighter, and therefore require less memory to run. It would also run faster and require less computing power.

   Drawbacks:

   - All the changes in groups now require a modification of said tool, so a good design is necessary to make it easy to change.

6. We document a list of the expected and required host/groups variables.
7. We remove all the unnecessary group and host variables that were part of the inventory but arent important anymore by using/providing a tool manipulating variable files (yaml), or by providing release notes.
8. We document how to export the cleaned up inventory into a new YAML file.
9. The generation of conf.d, env.d, and openstack_user_config becomes totally optional at this point: We know what is required in a build, and ask deployers to provide their own group/host mapping.

   At this point its optional because:

1. Assignment of hosts into groups can be done by the user with a simple .ini/.yaml file + documentation
2. Standard group structure is provided by default
3. We have documented the list of host variables, so they can be provided by the user
4. Generating container with their inventory_hostnames can be done by the user. Its just a series of host variables: ansible_host, container_name, container_tech, physical_host. It can even be done with a add_hosts and a loop based on a new variable like container_names (property of the host).
5. Assigning and tracking container IPs based on cidr_networks, reserved IPs, and already allocated IPs are also host variables. Deployers are responsible to provide an IP for their containers. Example, the lxc_container_create role creates IP, network, and interfaces configuration based on lxc_container_networks_combined, which a variable taking information from the inventory, by combining default lxc_container_networks with the container_networks variable, which is part of the inventory. Note: this part can be later replaced by a lookup. By using a lookup, we would simplify the inventory, by completely removing its container networks of the host vars.

10. We provide a script that runs all these actions for the user, but also allow step by step editions and manipulations.
11. We provide a new tool to generate a new kind of inventory based on what we learned from users, which wont necessary use the openstack_user_config, conf.d, or env.d. But we have all the time we need to do it better, because the expected inventory is not the same as the one we did the past.
12. We spin the old inventory out.

6.4.4 Testing

All the work items would be separately tested in the integrated gates.

6.4.5 Documentation impact

Large. The inventory would need a refactor to explain the expectations for people coming with their inventory, and for people that will use our generation tool. At the last step, if another tool is provided, it would also require documenting.

Each step would require modifying the reference, and maybe the operations guide.

6.4.6 References

None

CHAPTER

SEVEN

QUEENS SPECIFICATIONS

7.1 Generalize Infrastructure Roles

date
2017-09-10 14:00

tags
ansible, roles, mariadb, rabbitmq

Provide a synopsis as to why you are creating this spec/blueprint.

Include the URL of your launchpad blueprint:

- https://blueprints.launchpad.net/openstack-ansible/+spec/ansible-roles-reuse

Currently openstack-ansible is maintaining infrastructure roles that are used to deploy general infrastructure services such as MariaDB and RabbitMQ, which are applicable in non-OpenStack ansible environments also. With little to no refactoring these roles can be used to deploy the services in other Ansible managed environments also.

By maintaining robust, generalized service roles, they are more likely to be consumed, improved, and maintained by other operators in the greater Ansible community. This will benefit us by training us to keep a modular mindset when building the roles, which leads to better maintainability and wider testing for OSA consumers also.

In some cases we may wish to deprecate our openstack-ansible roles and consume more generalized upstream alternatives.

7.1.1 Problem description

In some of the roles (such as haproxy), we implement a very OSA specific deployment with very little reusability or configurability for a typical HAProxy deployer.

Other roles, such as Galera server, are fairly generalized and robust, but carry the openstack-ansible-service_name naming scheme, making it less likely for anyone NOT using openstack-ansible to use the role in their deployments.

pip_install is an example of a role that will require some minor refactoring to generalize it. The role performs some very out of scope tasks, such as repo management, which have nothing to do with installing pip. These features should be moved to appropriately modularized roles (a general repo management role?), so that pip_install is only doing the work it is meant to do.

7.1.2 Proposed change

Examine the following roles to identify and refactor out of scope tasks and orchestrate openstack-ansible specific configurations at the integrated repo level. If the role is built properly it should offer the necessary service configuration to be injected from the inventory and playbooks.

Roles to examine initially:

- openstack-ansible-pip_install
- openstack-ansible-lxc_hosts
- openstack-ansible-lxc_container_create
- openstack-ansible-haproxy_server
- openstack-ansible-memcached_server
- openstack-ansible-galera_server
- openstack-ansible-rabbitmq_server
- openstack-ansible-ceph_client

Once the work outlined above has progressed sufficiently, we should consider renaming some of the roles to a more appropriate naming, ie. openstack-ansible-galera_server becomes ansible-mariadb-cluster, etc.

Alternatives

N/A

Playbook/Role impact

The playbooks and especially inventory should eventually contain all of our openstack-ansible specific configurations. The infrastructure roles themselves should be generalized without an assumption or skew toward being consumed only by openstack-ansible.

In some cases this is already implemented, but in other cases the role will undergo significant changes or wholesale replacement to accomplish this.

Upgrade impact

Consumers of the roles will need to adjust to any major refactorings that take place, including possible renaming of the git sources and role names.

Security impact

N/A

Performance impact

N/A

End user impact

N/A

Deployer impact

Deployers who work frequently with openstack-ansible will benefit from the ability to use the same roles to deploy applicable services for other projects they work on besides OSA.

Developer impact

It is possible that this could draw more developers to assist in maintaining some of the roles. Cosmetic changes such as renaming may also help veteran OSA developers take a more abstract approach when crafting changes to these roles, which should make them more maintainable in the long run.

Dependencies

N/A

7.1.3 Implementation

Assignee(s)

Primary assignee:
Logan Vig (LP: loganv; IRC: logan-)

Work items

- Examine the infrastructure roles for out of scope tasks or reusability concerns. Address the issues by refactoring or replacing the role.
- Improve the role documentation if necessary with example playbooks demonstrating ad-hoc usage of the role.
- Rename the role and repo to a globally namespaced ansible role such as ansible-service-name.

7.1.4 Testing

N/A

7.1.5 Documentation impact

Improving and expanding the role documentation will be beneficial for reusability also.

7.1.6 References

- openstack-ansible 5/18 community meeting: http://eavesdrop.openstack.org/meetings/openstack_ansible_meeting/2017/openstack_ansible_meeting.2017-05-18-16.01.log.html#l-136

7.2 Integration of Blazar with OpenStack-Ansible

date
2017-12-17 00:02

tags
openstack, blazar, opnfv, promise

Blazar is a resource reservation service for OpenStack. It is used to book or reserve specific resources for a particular amount of time. This spec outlines the steps required to integrate Blazar with OpenStack-Ansible.

7.2.1 Problem description

Blazar is used to reserve OpenStack resources in advance for a specific amount of time. However, it needs to be installed manually with OpenStack-Ansible. No role exists to deploy it as other services are deployed.

7.2.2 Proposed change

The change consists of creating a new role for Blazar integration with OpenStack-Ansible. It will make it possible to deploy Blazar as part of the installation of OpenStack-Ansible, rather then requiring to install and configure it manually.

Alternatives

There are no alternatives.

Playbook/Role impact

This is a new feature added into OpenStack-Ansible. No role currently exists. Therefore, a new role, *openstack-ansible-os_blazar* needs to be written from scratch.

Upgrade impact

No upgrade impact.

Security impact

No security impact.

Performance impact

No performance impact.

End user impact

End user will be able to use Blazar out of the box, without going through any manual installation and configuration. One of the endusers is Promise, an OPNFV project, which is using Blazar, in an NFV context.

Deployer impact

No impact.

Developer impact

Little or no impact, since this feature will be optional and can be safely ignored.

Dependencies

No dependencies.

7.2.3 Implementation

Assignee(s)

Primary assignee:
Taseer Ahmed (Taseer)

Other contributors:
Fatih Degirmenci (fdegir)

Work items

Blazar is not available as a service for OpenStack-Ansible. No role already exists. A new role will be developed from scratch in compliance with the standards set by the community. The steps for developing this new role are as follows:

1. Create a new repository on GitHub.
2. Add tasks to the role.
3. Add tests for the new role.
4. Ensure that the role works well with AIO.

7.2.4 Testing

Tests will be developed to ensure that deployment of Blazar works and also to test the functionality of the deployed service.

7.2.5 Documentation impact

As this would be new feature added to OpenStack-Ansible, it needs to be documented, explaining all the configuration parameters.

7.2.6 References

Blazar Overview

- https://wiki.openstack.org/wiki/Blazar

Blazar Installation steps

- https://docs.openstack.org/blazar/latest/install/install-without-devstack.html

OPNFV Promise

- https://wiki.opnfv.org/display/promise/Promise

7.3 Integration of Congress with OpenStack Ansible

date
: 2017-08-30 00:02

tags
: openstack, congress

Blueprint on Launchpad:

- https://blueprints.launchpad.net/openstack-ansible/+spec/role-congress

Congress is the policy framework for OpenStack. This spec introduces the work required to deploy Congress, as a service for OpenStack Ansible.

7.3.1 Problem description

There are many policy frameworks for OpenStack. However, very few of them come with OpenStack Ansible. They need to be manually configured and installed. The aim of this spec is to deploy Congress with OpenStack Ansible, provided as a service to OpenStack Ansible and OpenStack users in general.

7.3.2 Proposed change

The change consists of integrating Congress with OpenStack Ansible during deployment phase of OpenStack.

Alternatives

Many policy frameworks for OpenStack exist. Tacker is one of them and has already been integrated with OpenStack Ansible. However, Tacker is more of a VNF Manager, mostly used for NFV related activites such as Service Function Chaining etc.

Playbook/Role impact

This is a new feature being introduced.An existing role does not already exist. A new role will be developed, e.g *openstack-ansible-os_congress*. This new role will be developed as per the steps outlined by the community.

Upgrade impact

No upgrade impact since this would be the first time implementation of the proposed change.

Security impact

No security impact.

Performance impact

Performance impact should be very low, it only needs a few preliminary packages.

End user impact

Congress uses a simple declarative language to define real world policies. Currently it needs to be manually configured and deployed. This feature would enable the users to use Congress as a service, and be able to manage OpenStack more efficiently.

Deployer impact

No default policies will be enforced. If the deployer chooses to enable Congress service, policies need to be defined as per the requirements.

Developer impact

Little or no impact, since this feature will be optional and can be safely ignored.

Dependencies

No dependencies.

7.3.3 Implementation

Assignee(s)

Primary assignee:
Taseer Ahmed (Taseer)

Other contributors:
Fatih Degirmenci (fdegir)

Work items

Congress is not available as a service for OpenStack Ansible. No role already exists. A new role will be developed from scratch in compliance with the standards set by the community. The steps for developing this new role are as follows:

1. Create a new repository on GitHub.
2. Add tasks to the role.
3. Add tests for the new role.
4. Ensure that the role works well with AIO.

7.3.4 Testing

Tests will be developed to ensure that deployment of Congress works and also test the functionality of the deployed service.

7.3.5 Documentation impact

As this would be new feature added to OpenStack Ansible, it needs to be documented, explaining all the configuration parameters.

7.3.6 References

Congress Overview

- https://wiki.openstack.org/wiki/Congress

Congress Installation steps

- https://docs.openstack.org/congress/latest/install/index.html#separate-install

7.4 Implement deployment stages for optimised execution

date
2017-09-14 12:00

tags
optimise, lifecycle

In order to improve ease of use, optimise execution and provide the ability to make use of pre-built artifacts in deployments this spec proposes the implementation of deployment stages.

- https://blueprints.launchpad.net/openstack-ansible/+spec/deployment-stages

7.4.1 Problem description

- In production environments with many target hosts there are sometimes transient failures that happen. When they happen the deployer is forced to re-execute playbooks which may go through many tasks which are already complete and do not need to be executed again. While a knowledgable deployer will make use of tag skipping and host scoping to reduce the execution time, this is not a skill the novice deployer has. In order to improve ease-of-use it should be possible for the playbooks to simply skip over the stages which have already completed on each host.
- In production environments it may be desired to make use of a fully artifacted deployment in order to ensures that multiple regions are deployed using exactly the same software. Currently there is no tooling included to facilitate the complete stack of artifacts (apt, git, python, container) that need to be built.
- Deployments currently do a lot of outgoing internet interaction in order to fetch packages, keys and other artifacts. The outgoing access is often a problem for deployers with a high security environment as the hosts are not able to access the internet directly. This access is also slower than it would be if these artifacts were locally staged before deployment.
- Deployments currently mix the build of artifacts with their installation and activation. This results in very long deployment times which often exceed maintenance periods available for operations. If the artifact build process could be executed and the artifacts could be staged without operationally impacting a production environment, then these could be executed prior to a maintenance slot and only the final step of implementing changes to use the new artifacts could be done in the maintenance slot.

- Deployments currently do a lot of staging actions in serial due to the combined install/config tasks in each role. This takes a very long time and is not necessary. If the build and stage tasks are properly split from the configuration changes then the build/stage tasks could be executed in parallel and only the configuration changes executed in serial, significantly speeding up large deployments.

7.4.2 Proposed change

The stages proposed are as follows:

1. Build: This stage prepares artifacts which are general purpose. This stage could be executed by a CI process in order to prepare the appropriate artifacts and stored on a server to be used across multiple regions. Alternatively it could be executed in-line for a single build (using developer_mode. Artifact examples include distribution software packages, container rootfs tarballs, python venvs, etc. If not executed in-line, the build process should be executed on any designated host and produce artifacts which can be copied to a web server. There must be a well defined manifest detailing the artifacts produced which can easily be used for a staging process to understand which items to fetch.
2. Stage: This stages all artifacts from the Build stage using the manifest produced. The stage is optional and will only be executed if the Build stage was executed to build all artifacts. The stage will most likely only be a playbook rather than something in the role, making it easy to allow deployers to implement alternative staging mechanisms if they choose to. This stage will be executed in parallel across all hosts/containers to ensure that it executes quickly.
3. Install: This stage executes the code path which uses the staged or built artifacts and the prepared OSA configuration to create containers and install all services. This process should not restart containers or services or enact any changes to an existing environment which will disrupt it. This stage will be executed in parallel across all hosts/containers to ensure that it executes quickly.
4. Configure: This stage executes the implementation of configuration changes to configuration files and starts/restarts the applicable services or containers. This stage will be executed serially to ensure that service disruption is minimised.

The tasks for each stage will be explicitly broken into task files, for example:

- <service>_build.yml
- <service>_install.yml
- <service>_install_apt.yml
- <service>_install_nginx.yml
- <service>_configure.yml
- <service>_configure_nginx.yml
- <service>_configure_ssl.yml
- <service>_configure_keys.yml

The general idea with breaking out the task files is to implement conditional and/or dynamic inclusions where appropriate to ensure that the tasks are not even evaluated unless a broad condition is met. This is different to having a bunch of tasks in a single file which all have conditions because Ansible will not have to evaluate each task in turn, but instead evaluate whether a block of tasks should be evaluated. This reduces execution time.

Some examples:

1. If pre-built artifacts are available when the role executes, skip the build stage tasks.
2. If there is no repo server in the environment, do not try to download any python venvs or other artifacts.
3. If `ansible_pkg_mgr == 'apt'`, do not evaluate any tasks related to yum.

As part of this solution, the build and install stages should drop local facts on to target hosts when the stage completes. The local fact will prevent that stage being executed again through a conditional include. This provides a checkpoint restart mechanism so that if a deployer executes setup-everything the execution will be much faster because it will skip whole stages and continue from where it left off. This also means that if pre-built artifacts are used, these stages will be skipped and the deployment in an environment will be much, much quicker.

The facts dropped would be tag-specific - for example the fact dropped would indicate that the cinder service has the 14.2.0 release installed on the host, meaning that the build and staging tasks do not need to be run if the proposed tag and the tag deployed are the same. This behaviour will be overridable via another variable which enables a forced rebuild or forced reinstall.

Alternatives

1. Put up with long deployment times.
2. Document in better detail how to reduce deployment times using package mirrors, proxies and such.

Playbook/Role impact

New playbooks will be implemented which allow the deployer to executed the more targeted build process and to prepare the artifacts. The existing playbooks will continue to work, but will be adjusted to make use of the appropriate facts to skip the previously executed build process if that has already been executed.

The roles will be where the greatest impact will be as many of the tasks will be re-organised to facilitate the staged process.

Upgrade impact

Being able to make use of pre-built artifacts for an environment will mean that an upgrade process should be able to more easily roll back to a previous state if need be.

Security impact

As this process will improve the ability to ensure a consistently built environment, this will likely improve the security posture of a deployment.

Performance impact

Hopefully the deployment and upgrade performance will be far better than it is now. The running deployment performance should be no different.

End user impact

There will be no difference to end-users of the deployed OpenStack environment.

Deployer impact

Deployers will continue to have the same entry points, but will gain the ability to pre-build artifacts for their environment in order to ensure that deployments and upgrades execute more quickly and reliably.

Developer impact

These changes should improve the developer experience by reducing the time taken to implement an AIO.

Dependencies

None

7.4.3 Implementation

Assignee(s)

Primary assignee:
jesse-pretorius (odyssey4me)

Work items

Each of the roles implemented in the default AIO will be worked through in sequence to re-arrange and optimise based on this workflow. The work items are not being detailed here but will be reflected in gerrit through the blueprints topic and will be visible in launchpad.

7.4.4 Testing

It may be possible for us to make use of pre-built artifacts for gate testing in order to reduce the time take for integrated tests. The option of publishing the last successful builds artifacts for each branch on OpenStack Infrastructure will be explored. These artifacts will be for development tests only and not useful for production environments.

7.4.5 Documentation impact

The staged deployment process will need to be documented and the details of how to opt-in to make use of an artifacted build will need to be included.

7.4.6 References

None

7.5 Documentation improvements

date
2018-01-08 22:00

tags
docs, user stories, walkthrough

Include the URL of your launchpad blueprint:

- https://blueprints.launchpad.net/openstack-ansible/+spec/example

People are often confused when they deploy with openstack-ansible, because they only partially read the documentation, or landed on the wrong documentation.

Our deployment guide is already close to what I call a wizard/ walkthrough, but some parts are easily missed by the deployers.

On top of that, some very nice advanced documentation are often skipped, or arent promoted to their right value.

7.5.1 Problem description

When people land on our deployment guide, which is probably the first link they access, whether they come from the OpenStack deployment guides https://docs.openstack.org/pike/deploy/ or from our main developer page https://docs.openstack.org/openstack-ansible/latest/ , they are facing the following issues:

- The landing page is overwhelming, as its a series of link. What do you click?
- The first links clicked (example: https://docs.openstack.org/project-deploy-guide/openstack-ansible/pike/overview.html#) is just more clicks towards content, and doesnt provide any useful information (the structure is already displayed on the left side of the page).
- Anyone wanting to quickly deploy an openstack-ansible cloud has no way to know we have an AIO toolkit that could help.
- Anyone wanting to deploy a production cluster with Netapp for example will most likely not find the appropriate documentation while reading the deploy guide: Its easy to miss the importance of configuration on https://docs.openstack.org/project-deploy-guide/openstack-ansible/latest/configure.html#advanced-service-configuration
- The AIO is listed in the contributor guides, where it could be available on any deployment part.
- There is an hard to find advanced configuration section, hidden inside the operations guide. It should probably be an appendix of the deploy guide.

- We have many overlaps of documentation doing about the same thing, we should clean things up (for example the advanced configurations in deploy + operations, the inventory in operations + contributors + reference)
- There are too many appendices in the deploy guide. Some deserve their own section. According to the spec http://specs.openstack.org/openstack/docs-specs/specs/pike/os-manuals-migration.html, end-user content such as concept guides, advice, tutorials, step-by-step instructions for using the CLI to perform specific tasks, etc. I think we could technically move scenarios there, as they are step-by-step instructions using shell scripts to perform some specific deploys.
- The role maturity is hard to find. If I were a new deployer, Id like to know what I could do with OpenStack-Ansible, and the role maturity matrix would tremendously help. Sadly Id never see it in my first read of the documentation.
- The upgrade guide is our user guide. Why not considering upgrades as a specific kind of operation? In my opinion, it should be part of the operations guide, as a major chapter in the operations, in the same way as minor updates, or scaling the environment.
- We dont motivate people to contribute back directly from the deploy guide. The last step after verifying that the system works should be how to extend and contribute to OpenStack-Ansible.
- People could be confused on how to best contribute to the project.

7.5.2 Proposed change

Have some kind of notice at the beginning of the deploy guide, pointing to our user stories (but advising to read the deploy guide first). The first user story would be the AIO, with the quickstart AIO content, for those who want devstack-like easiness, developers, or for those who want to prototype. Add more user stories into the user guide, for ceph (test, prod and ceph-ansible integration), for l3 routed scenarios (tests and prod), for offline installs.

Move advanced topics like inventory, container networking, custom layouts and security principles into a new reference section of the documentation. This section should probably hold the links to roles documentations, and should also be linked from the deploy guide where appropriate.

Highlight the importance, in the deploy guide, of our advanced topics (reference). Its important for new deployers to know where to find documentation on how to do X thats not part of a user guide.

At the end of the deploy guide, continue the deploy story by pointing to our operations and contributions guide. That could be added into a next steps section.

The contributors guide can also be enhanced by listing where the help is wanted: docs (and their manually testing, like for the operational guide), bugs (triaging and fixing the low hanging fruits), test coverage, This section could be altered when the priorities change.

For improving the reading experience, ensure that each page has a proper structure:

- Only content should appear in the content part of the page
- The chapters should only be in the upper-left section of the page ToC, and pointing to this guide chapters, not the whole documentation items (avoiding something like https://docs.openstack.org/openstack-ansible/latest/user/index.html)
- The page headers should only be in the lower-left section of the page ToC.

Alternatives

Not changing the docs, or partially implementing those changes.

Playbook/Role impact

None

Upgrade impact

None

Security impact

None

Performance impact

None

End user impact

None

Deployer impact

New deployers should be less overwhelmed by Openstack-Ansible

Developer impact

None

Dependencies

None

7.5.3 Implementation

Assignee(s)

Primary assignee:
jean-philippe-evrard

Other contributors:

- TODO

Work items

Each paragraph of the proposed change can be considered as a work item.

7.5.4 Testing

Nothing new.

7.5.5 Documentation impact

This is a docs only change, so this whole change has a documentation impact. However, because we dont change the structure of the docs themselves, it should not be very difficult to implement.

7.5.6 References

This improvements only happen to improve our readability, and to follow whats generally expected to find in each of the documentations:

7.6 ELK Stack

date
2017-12-11 11:00

tags
logging, monitoring, operations

Blueprint on Launchpad:

- https://blueprints.launchpad.net/openstack-ansible/+spec/elk-stack

Log file analysis is an important part of maintaining and troubleshooting OpenStack clouds, but using traditional single server methodology to analyze the logs on clouds with tens, hundreds or thousands of servers can become problematic and unwieldy. By leveraging the search, collation and analysis features of the ELK (Elasticsearch[1], Logstash[2] and Kibana[3]) stack we can provide a cloud level view of all of the log files. The ELK stack also provides the ability to correlate log messages across various services, perform detailed log analysis and do trending based on metrics derived from log messages.

[1] https://elastic.co/products/elasticearch
[2] https://elastic.co/products/logstash
[3] https://elastic.co/products/kibana

7.6.1 Problem description

For deployers and operators findings specific events in the myriad log files produced by the various OpenStack, system and ancillary services can be tedious and error prone. With traditional tools the possibility of missing critical log entries grows as the size of the cluster increases. Log file analysis provides vital information about the state of the OpenStack services as well as the underlying hardware. Currently there are no tools provided by OpenStack-Ansible to detailed log analysis, correlation and trending.

7.6.2 Proposed change

Utilizing the logging/utility node we install the ELK stack in containers, logs are shipped from the individual nodes/containers using the Filebeat package. Using Filebeat to perform the initial log shipping allows us to do initial multiline parsing distributing the load away from a single Logstash container. Version requirements of the ELK packages will be maintained in the ELK roles and barring security fixes the major version of those packages should not change during the release cycle of Openstack. The ELK roles are consumed via Ansible Galaxy pointing to specific SHAs.

Notable changes:

- Create 3 containers on the logging/utility node, one each for Elasticsearch, Logstash and Kibana. (Additional containers can be created to facilitate HA if needed.)
- Install the Filebeat package on all nodes/containers
- ELK and Filebeats galaxy role SHAs added to *ansible-requirements.yml*

Alternatives

Logs are currently shipped to a centralized rsyslog-server container on the logging/utility server allowing for some sort of centralized log parsing using command line utilities. There are other 3rd party solutions with various levels of cost, adoption and support.

Playbook/Role impact

The changes required are located in stand alone playbooks. Additional roles will need to be created for Logstash, Kibana and Filebeat, the *ansible-elasticsearch*[4] maintained by elastic.co provides Elasticsearch. Configuration can be stand-alone or integrated into the *user-variables.yml* and *user-secrets.yml* files.

Upgrade impact

As this is the initial implementation there is no upgrade impact. Future versions will require upgrade planning as it may be necessary to upgrade versions of the ELK packages, OpenJDK packages and possibly the Elasticsearch database itself.

[4] https://github.com/elastic/ansible-elasticsearch

Security impact

This software provides a web based front end as well as API access to any information contained in the Openstack, service and system logs that are shipped to it. As such it will need to be only visible to authenticated users. All access can be secured through the traditional hardening that is applied to any standard web service, namely TLS and an authentication mechanism. Furthermore since the ELK stack is behind a VIP we can limit access to certain IPs and/or networks via a number of ACLs.

By default logs are shipped in plaintext, it is possible, however, to enable SSL encryption on this transport should it be needed.

Performance impact

Based on testing and real-world analysis the largest performance impact will be on the logging/utility server. As this devices original intent was to perform log processing this is expected and not unusual. The filebeat service running in each node/container has demonstrated a negligible performance impact, but certain best practices such as limiting logging levels and eliminating tracebacks in the logs will help maintain the light footprint. Filebeat should not impact the operation of any Openstack services as it is simply a log file processor/shipper, although network utilization could be a concern should debug logging be enabled on a particularly busy service.

Elastic.co is the maintainer of all of the software other than Java, which is maintained by Oracle corporation. Both of these entities provide enterprise software and thus follow strict release schedules and have reliable upstream repositories for their software.

End user impact

End users should not notice the changes from this work. This is primarily intended for deployers and operators. This change does give operations teams more insight into the environment and will hopefully facilitate a more performant and stable deployment.

Deployer impact

The ELK stack is an optional component and does not directly interact with any Openstack services. All of the ELK packages are provided via apt/yum repositories. An additional secret will need to be created for the *kibana* user. The filebeat package will be installed in all containers and on all nodes but it is extremely lightweight, with configuration stored in */etc/filebeat*. Java is required for ELK so the *openjdk* (default) or JDK implementation of the deployers choosing will need to be installed in three containers on the logging/utility node.

Developer impact

This should be a minimal change for developers, the one thing that they will need to keep in mind is if additional log files are added they will need to be added to the filebeat configuration, this can be handled by re-running the filebeat play against the containers with the new logs.

Dependencies

There are no dependencies.

7.6.3 Implementation

Assignee(s)

Primary Assignee:
David Wilde (d34dh0r53)

Work items

1. Create ELK and filebeats roles in openstack-ansible, these roles will be generic enough to be published to ansible-galaxy so that they are usable by the Ansible community at large.
2. Create playbook(s) to install the ELK stack and filebeats, these playbooks will install the OpenStack specific configuration and parsing files.
3. Create testing procedures for the stack
4. Documentation

7.6.4 Testing

The ELK stack should be tested on each commit by ensuring that the services start and that logs are flowing into the system and being parsed correctly. This can be acomplished by injecting a line into a services log file and then using the elasticsearch API via curl to verify that the line was correctly inserted into the database with the expected fields parsed.

7.6.5 Documentation impact

Along with the general installation procedures and configuration the key points of documentation will be:

- Filebeats parsing rules
- Logstash parsing rules
- Kibana dashboard configuration
- The default Kibana dashboard
- Performance impact and tuning of the ELK stack

7.6.6 References

7.7 Provide option of hybrid messaging backends

date
2017-09-31 10:00

tags
messaging, rabbitmq, qpid

OpenStack services make use of a message bus system for both remote procedure calls (RPC) between components and to emit notifications. The aim of this spec is to layout a plan for providing an alternative to RabbitMQ for RPC messaging.

https://blueprints.launchpad.net/openstack-ansible/+spec/hybrid-messaging

7.7.1 Problem description

RabbitMQ is currently used as the message bus system for all remote procedure calls (RPC) and notifications of OpenStack services deployed by OpenStack-Ansible. While RabbitMQ is well tested and has wide acceptance across OpenStack projects and deployments, it may not be the most efficient option for RPC messaging. A brokerless message queue may provide greater performance of messaging throughput and be less of a bottleneck, particularly in larger scale deployments.

7.7.2 Proposed change

This spec proposes offering Qpid Dispatch Router as an alternative option for RPC messaging within an OpenStack-Ansible deployment.

Deployers will be able be given more options for messaging backends:

- RabbitMQ for both RPC and notifications (will remain the default deployment)
- Qpid Dispatch Router for RPC (with no dedicated backend for notifications)
- Qpid Dispatch Router for RPC and RabbitMQ for notifications (hybrid messaging)

Alternatives

Leave RabbitMQ as the sole option for messaging within OpenStack-Ansible deployments.

Playbook/Role impact

Playbooks that deploy OpenStack services will need to be modified to make any required against the deployers messaging backend of choice. Roles will need to include additional package dependencies to connect to the Qpid Dispatch Router.

Upgrade impact

An upgrade scenario will test the migration of a deployment from using RabbitMQ.

Security impact

The default deployment of Qpid Dispatch Router should provide as close as possible parity with OpenStack-Ansibles default RabbitMQ deployment including use of TLS/SSL encryption and virtualhost namespacing of messaging data.

Performance impact

Especially in larger scale deployments, there is a potential improvement in the throughput of messages and lowered CPU utilization.

End user impact

When chosen to be implemented by a deployer, the changes involved should be transparent to end users.

Deployer impact

There would be no immediate impact to deployers as the changes involved would be entirely opt-in initially. For deployers choosing to deploy Qpid Dispatch Router, the service will be installed, likely in a new container, and OpenStack services will be configured to make use of it.

Developer impact

New roles for OpenStack projects should include configuration options to allow for using either RabbitMQ or Qpid Dispatch Router and testing of each.

Dependencies

N/A

7.7.3 Implementation

Assignee(s)

Primary assignee:
jimmy-mccrory (jmccrory)

Work items

- Create a new role for the installation of Qpid Dispatch Router
- Create a playbook to deploy Qpid Dispatch Router
- Modify OpenStack service configuration templates within each role to allow a transport URL other than RabbitMQ and default variables to support that
- Add required client package dependencies to roles
- Create test scenarios in the roles to deploy using Qpid Dispatch Router as the messaging backend for RPC
- Create a common playbook for any Qpid Dispatch Router configuration changes required by individual OpenStack projects that the OpenStack project playbooks will consume
- Create test scenarios in the integrated gate for greenfield and upgrade deployments

7.7.4 Testing

A Qpid Dispatch Router scenario would be created within the roles of OpenStack projects which make use of a message queue and the integrated OpenStack-Ansible repo to ensure installations and deployments, including upgrades, remain functional.

7.7.5 Documentation impact

Documentation will need to be added for the configuration options of Qpid services, the configuration options for OpenStack services to make use of Qpid services, and any associated maintenance tasks within the Operations Guide.

7.7.6 References

AMQP 1.O (Qpid Dispatch Router) Oslo Messaging Driver Reference:

- https://docs.openstack.org/oslo.messaging/latest/admin/AMQP1.0.html

Message Routing- A Next-Generation Alternative to RabbitMQ:

- https://www.youtube.com/watch?v=R0fwHr8XC1I

Hybrid Messaging Solutions for Large Scale OpenStack Deployments:

- https://www.youtube.com/watch?v=o30YaqfLV9A

7.8 Hyper-Converge Containers

date
2017-09-01 22:00

tags
containers, hyperconverged, performance

Reduce container counts across the infra structure hosts.

To lower our deployment times and resource consumption across the board. This spec looks to remove single purpose containers that have little to no benefit on the architecture at scale.

This change groups services resulting in fewer containers. This does not mix service categories so theres no worry of cross polluting a different service with unknown packages or unknown workloads. Were only look to minimize the container types we have and simplify operations. By converging containers were removing no less than 10 steps in the container deployment process and the service setup. Operationally were reducing the load on operations teams managing clouds at any scale.

7.8.1 Problem description

When we started this project we started with the best of intentions to create a pseudo micro-service model for our system layout and container orchestration. While this works today, it does create a lot of unnecessary containers in terms of resource utilization.

7.8.2 Proposed change

Converge groups of containers found within the *env.d* directory into a single container where at all possible. Most the changes we need to get this work done have already been committed. In some instances we will need to revert a change to get the core functionality of this spec into master but there will be little to no development required to get the initial convergence work completed.

Once the convergence work is complete we intend to develop a set of playbooks which will allow the deployer to run an opt-in set of tasks which will cleanup containers and services wherever necessary. Services behind a load balanacer will need to be updated. Updates to the load balancer will be covered by the opt-in playbooks provided the environment is using our supported software LB (HAProxy). The opt-in playbooks will need to be codified, tested, and documented. Should it be decided that the hyper-converged work is to be cherry-picked to a stable branch, the new playbooks will need to first exist and be tested within our periodic gates. We should expect no playbook impact in-terms of the general deployer workflow.

Alternatives

We could leave everything as-is which carries the resource requirements we currently have along with an understanding that the resources required will grow given the fact OpenStack services, both existing and net new, are ever expanding.

Playbook/Role impact

At least one new playbook will be added allowing a deployer to cleanup old container types from the run-time and inventory should they decide to. The cleanup playbook(s) will be opt-in and will not be part of our normal automated deployment process.

Upgrade impact

There is no upgrade impact with this change as any existing deployment would already have the all required associations within inventory. Services would continue to function normally after this change. Greenfield deployments on the other hand would have fewer containers to manage which reduces the resource requirements while also ensuring we retain the host, network, and process separation we have today.

We will create a set of playbooks to cleanup some of the redundant containers that would exist post upgrade however the execution of this playbook would be opt-in.

Security impact

Security is not a concern within this spec however reducing the container count would reduce the potential attack surface we already have.

Performance impact

Hyperconverging containers will reduce resource consumption on physical host. Reducing the resources required to run an OpenStack cloud will improve the performance of the playbooks and the system as a whole.

End user impact

N/A

Deployer impact

Deployers will have fewer containers to manage and be concerned with as they run clouds for long periods of time.

- Within an upgrade scenario a deployer will have the option to opt-in to a hyperconverged setup. This change will have no service impact on running deployments by default.

Developer impact

N/A

Dependencies

- If were to test the opt-in cleanup playbooks well need a periodic upgrade gate job. The playbooks would be executed by the upgrade gate job and post results to the ML/channel so that the OSA development team is notified of the failure.

7.8.3 Implementation

Assignee(s)

Primary assignee:
Kevin Carter (IRC: cloudnull) Major Hayden (IRC: mhayden)

Work items

- Converge the containers into fewer groups
- Create the opt-in container reduction playbooks
- Document the new playbooks

7.8.4 Testing

- The core functionality of this patch will be tested on every commit.
- If the upgrade test dependencies are met we can create a code path within the periodic gates and test the opt-in cleanup playbooks.

7.8.5 Documentation impact

Documentation will be created for the opt-in container cleanup playbooks created.

7.8.6 References

N/A

7.9 OpenDaylight with BGPVPN support in Neutron

date
2017-11-17 16:30

tags
OpenDaylight,Open vSwitch,neutron,BGPVPN,L3,DC-GW

Blueprint on Launchpad

- https://blueprints.launchpad.net/openstack-ansible/+spec/opendaylight-with-bgpvpn-support

This spec introduces the work required for OpenDaylight configured with BGPVPN through Openstack-Ansible to enable Openstack deployments with extended L3 support.

7.9.1 Problem description

The support for BGPVPN is available from OpenDaylight since its Beryllium release. Openstack can make use of this feature by configuring neutron to use BGPVPN service plugin.

" https://docs.openstack.org/networking-bgpvpn/latest/user/drivers/opendaylight/index.html " " https://docs.openstack.org/networking-bgpvpn/latest/user/usage.html "

In addition to it, quagga/zrpcd and its dependent packages have to be installed along with OpenDaylight for configuring OpenDaylight as a BGP speaker.

7.9.2 Proposed change

For the configuration of OpenDaylight as a BGP speaker that integrate into deployers infrastructure, a new OpenStack-Ansible playbook with required ansible tasks for installing quagga and its required packages will be written. The wiring of the OpenDaylight configuration as a BGP speaker will be done inside the neutron role, which configures OpenDaylight (see playbook/role impact for details).

The initial supported distros would be CentOS and Ubuntu.

7.9.3 Alternatives

There are other bgpvpn backend drivers available with neutron like BaGPipe, OpenContrail driver and Nuage Network driver to configure the BGPVPN.

Playbook/Role impact

The new playbook will be added in OpenStack-Ansible which installs quagga and configure OpenDaylight for BGP speaker. This playbook would get executed after neutron playbook in neutron server node (in case of ha deployment, among three neutron server containers, one is chosen), because quagga just needs to get installed in one of the OpenDaylight node and run additional karaf CLI commands to make it as BGP speaker.

The proposal is to add a extra variable in neutron_plugin_base, overriding the default ODL behavior, and trigger the usage of BGPVP. When `neutron_plugin_type` variable set to `ml2.opendaylight`, `neutron_plugin_base` list variable having *network-*

ing_bgpvpn.neutron.services.plugin.BGPVPNPlugin' item, then neutron server node will be installed/configured with OpenDaylight and Quagga.

Upgrade impact

This is the first implementation of OpenDaylight with Quagga, so no upgrade concerns yet.

Security impact

Networking-bgpvpn configuration requires the setup of a username and password for northbound authentication towards OpenDaylight. The deployer should be able to configure those credentials.

Communication between the controller and the switches will not be secured by default. Using TLS to secure the communications is considered a stretch goal, and deployers need to consider this security implication, specially in production environments. For more information on secure communications between OpenDaylight and OpenvSwitch, see the *References*.

Performance impact

For those choosing to opt-in this deployment method, some extra packages need to be installed on the neutron server, which would make installation last a bit longer.

Extra resources are needed to run the OpenDaylight SDN controller on the system as well. However, performance in Neutron API calls should be minimum.

End user impact

End users would have a new networking and BGPVPN API available through Neutron. This would enable them to create bgpvpn scenarios (e.g. Router and Network association with BGPVPN). This will require some documentation with troubleshooting steps to verify that OpenDaylight is working properly, as well as pointers to OpenDaylights official documentation.

No changes to Horizon or other OpenStack components are expected.

Deployer impact

New artifacts are being deployed, namely the Karaf runtime for OpenDaylight, quagga/zrpcd, thrift and the networking-odl pip package. OpenDaylight requires around 2.5G of RAM to work properly, with OpenStack, that would need to be considered when dimensioning the host where it will run.

Also deployers need to ensure that OpenvSwitch with version >= 2.8 is deployed in all networking nodes, namely compute hosts and hosts where neutron agents are running.

Developer impact

Developers have a new playbook to maintain, whose scope is very reduced and not in the path of all deployments.

Developer impact is very low, all tasks for BGPVPN deployment will be optional and can be ignored. The tasks wont be skipped, but instead no host will be matched for the new playbooks. This way, if we put the playbook on the path for every developer/deployer, the impact will be minimum.

Dependencies

There are no dependencies

7.9.4 Implementation

Assignee(s)

Primary assignee:
Periyasamy Palanisamy (epalper) Dimitrios Markou (mardim)

Work items

1. Add new playbook for installing/configuring quagga/zrpcd
2. Task to configure ODL as a BGP speaker
3. Make neutron role to get configured with OpenDaylight BGPVPN driver
4. Create a new test and verify that it passes
5. Document the new functionality

7.9.5 Testing

As a replacement of Neutron backend, this new scenario should provide the same capabilities of existing backends, so existing tests should be run.

A test specific for OpenDaylight can also be implemented, in the same way as there are currently tests for Calico or DragonFlow.

7.9.6 Documentation impact

The new scenario *OpenDaylight+BGPVPN* will be documented, explaining the configuration parameters required to deploy it.

7.9.7 References

OpenDaylight scenario with OpenStack-Ansible

- https://docs.openstack.org/openstack-ansible-os_neutron/latest/app-opendaylight.html
- https://git.openstack.org/cgit/openstack/openstack-ansible-specs/tree/specs/pike/opendaylight.rst

packaging and installing quagga/zrpcd packages

- https://github.com/opnfv/apex/blob/master/build/build_quagga.sh

BGP peering with OpenDaylight

- https://github.com/opnfv/sdnvpn/blob/master/sdnvpn/test/functest/testcase_3.py

Enabling BGPVPN mechanism driver at neutron

- https://docs.openstack.org/networking-bgpvpn/latest/user/drivers/opendaylight/index.html

7.10 Python Build/Install Process Simplification

date
2017-09-06 13:00

tags
python, build, source, repo

The current python wheel/venv build process is not easily understood, and the install process has become complicated. This blueprint aims to work towards making it simpler to deploy, simpler to understand and to make many of the current features which are forced on all deployers to be opt-in.

Launchpad Blueprint: https://blueprints.launchpad.net/openstack-ansible/+spec/python-build-install-simplification

7.10.1 Problem description

Building

The Python repository used for OpenStack-Ansible deployments is used to prepare Python wheels for any git- or pypi-sourced packages for an environment. Using wheels speeds up the installation of the package and takes away the need to install the distribution packages required to compile the package when installing.

The repository preparation process also prepares Python virtualenvs for all OSA roles with the prefix `os_` (which are expected to be OpenStack services) in order to speed up the deployment of the services by downloading a complete virtualenv instead of installing the packages individually for every host that needs the service.

The `py_pkgs` lookup, which pulls together the information used by the build process. It is a black box in terms of what it does, making some decisions about the information it reads and outputs which are not documented anywhere other than in the code itself. The code is not easily modified without breaking the process and is therefore most often left alone and not well maintained, resulting in an increasing amount of technical debt. The subsequent jinja in the repo-build role which processes it is tough to work through

and not easily maintained. While both of these could be adjusted to make use of different plugins or filters, it would remain a set of black boxes which are complex to untangle.

The way that git repositories are specified and parameters are provided to the build process does not scale very well. Each git repo requires at least two flat variables to be set (`git_repo` and `git_install_branch`) and can optionally have more set. This model of setting variables makes it really easy to override individual settings, but requires the use of a pattern match mechanism to discover all the settings (which is why we use the lookup to do it). The settings are also put in disparate places, making them hard to find - defaults/repo_packages, role/defaults. It is not very obvious to most newcomers how to change them and it is not obvious to many veterans what many of the settings mean. It often requires a lot of code walking to understand the meaning of some settings like `venvwithindex` and `ignorerequirements`.

The git clone process used to fetch the git sources in order to use when building is done asynchronously in order to improve the time to completion, however individual asynchronous tasks cannot be retried in Ansible, and the git clones commonly fail. This is an Ansible limitation which we could work around by implementing our own action module, but this would increase the technical debt as we would have to constantly keep the module code updated as we update to later versions of Ansible.

When building wheels, pip has no way of resolving all dependencies up-front. The only capability it has is to resolve the requirements for the current package. It then processes each package requirement in turn. To do so requires downloading the package and unpacking it to read the requirements. This is a sequential process and therefore takes a long time when processing packages with a lot of requirements as is typical for OpenStack projects.

In Kilo the OpenStack requirements management process did not have the jobs which tested the co-installability of all OpenStack packages and produced the `upper-constraints.txt` file as a manifest of which package versions worked together. We therefore needed to do our own processing of all python packages which would be installed by the roles and had to compile a set of requirements and constraints across them all for the purpose of building the wheels, and ensuring that the installed set were consistent for a build. When the OpenStack requirements repository started publishing the upper-constraints file we adopted it immediately to help keep builds more consistent. However, we still produce our own `requirements_absolute_requirements.txt` file which is used for all pip install tasks in order to ensure consistency and to ensure that the packages we built from git are used (instead of making the install in the role do the install from the git source, it installs from the wheel held in the repo server). However this is not practical any more as there are requirements for different services and needs which are not resolvable down to a common set - we need to be able to allow the installation of any version of packages and only apply constraints when needed.

Some of the venvs we build do not adhere to the OpenStack requirements process and therefore sometimes cannot be built using the upper constraints file. There has also been some interest in being able to do mixed series deployments instead of homogenous deployments. This would involve preparing a venv containing packages from a different series with a different set of constraints. Currently the constraints used in the repo build process are global - we only have the ability to enable/disable their use when building venvs. It would be better to be able to specify a global fallback for constraints, but to allow per venv constraints too.

The use of Python 2.7 for OpenStack and Ansible is waning and the need to shift everything to use Python 3.5 has arisen as a new requirement. The tooling will need to be shifted to implement the venvs using Python 3.5 where applicable, but may still need to prepare venvs using Python 2.7 if a service does not yet support running in a Python 3.5 environment.

In Newton we introduced the ability to do multi-architecture builds to cater for multiple architectures, then had to also split out multi-distro builds due do wheel/venvs references to C libraries being different for each distro due to the libraries available being different. Currently this is working, but it makes

the repo build process much more complex and take a lot more time. The process to synchronise the per-distro and per-architecture built artifacts is error prone and confuses many newcomers to the project.

In order to facilitate using the repo-server to respond to pip index queries, multiple directories and symlinks have been used to prepare the appropriate structure so that the correct responses are given back to pip. The process of setting up all the symlinks is very time consuming and in some places the process may cause dead links, especially when rebuilding for a specific release tag.

Storing

Once the wheels, venvs and other artifacts are built for an environment they are stored and synchronised between the repo containers using a combination of rsync and lsyncd. While this sync process is generally OK, it is commonly a cause for confusion and requires a complex troubleshooting process to figure out why packages are not present.

Installing

The consumption of the prepared wheels and virtualenvs has changed over time. With the introduction of `developer_mode` into the roles there is a lot of code and functionality duplication between the repo build process and the role installation process.

The need to cater for the optional inclusion of a variety of plugins/drivers in the venvs either through the use of additional Python packages or by symlinking system packages into the venv (when the package is proprietary or unavailable via git or pypi) causes further complexity in the process.

When executing a pip installation, pip always looks for packages in the following order: local cache, local folder, default index, extra indexes. Pip will always check all locations before deciding which to use for the installation. This means that if there are multiple indexes used, it queries them all. This can be slow if any of those are not local to the environment.

7.10.2 Proposed change

- Change the repo build process to, by default, only build wheels for git sources it is given without also building the dependencies. The ability to build all wheels will still be there, but will not be the default behaviour. This will cut down the time taken in this process when in CI, development environments or small online environments where it is not necessary to build/store all the wheels. The full build will only be necessary for offline deployments and for environments where the deployer specifically opts-in to ensuring that everything is built.
- Replace the current storage structure for wheels with a flat directory. This directory will be served via the pypi API provided by the very simple pypiserver application. If we need to continue to provide per-distro or per-architecture wheels then we could implement distro/arch indexes which are supplied by individual folders. However, it is unlikely that this will be necessary.
- Use nginx as a reverse proxy which responds to requests from pip by first trying against the local pypiserver, then against tarballs.openstack.org and then against pypi. This will allow nginx to cache all downloaded packages (speeding up subsequent requests) without the repo server having build them.
- Implement changes to the roles to allow service-specific constraints to be applied when building venvs. This allows a CI process to build service venvs and to publish the list of tested versions for that service. Then for production builds the published list can be used as a constraint for the venv

to ensure that production builds use the same versions. This solves a problem we have today where some projects (eg: tempest, rally, gnocchi) have to be built unconstrained as they do not conform to the global requirements process.

- Implement changes to each role to handle the wheel building and venv building, but do it in such a way that only the build can be executed by using tags, setting a specific flag, or include_role and tasks_from. The specific dependencies can then be itemised in the role and the role can be used for artifact preparation.
- Remove all pip install activities from hosts, replacing them with the use of distro packages exclusively for any python requirements on the hosts. We should avoid implementing as many python packages on the host as possible and focus all efforts on implementing everything we need (including the Ansible requirements for targeted hosts) into venvs. All Ansible tasks should then specifically use the appropriate venv when executing tasks, avoiding the use of any python libraries on the host. This prevents system package conflicts and will reduce the host package installation requirements.
- Implement a playbook which is optionally used to prepare pre-built venvs for an environment as they are today. If a deployer wishes to prepare the venvs in a build process, the playbook should be exercised in the build process and should be executed on a designated build host which will make use of ephemeral containers and/or virtual machines on the build host to exercise the builds for the necessary distribution and architecture combinations.
- Remove the complex git caching/staging process which exists today and make the use of the repo server for git caching for the services that require it (eg: nova-console uses novnc/spice from git) entirely optional.
- Implement a playbook which can be used to stage offline installs by downloading all built artifacts (completed, perhaps by a CI job) to the deployment host, then distributing them appropriately.
- Simplify the constraints management by implementing the use of constraints in the following order:

  constraint user-specified-constraints.txt constraint openstack-ansible-pins.txt constraint openstack-upper-constraints.txt

  This would replace the current method which merges the various constraints into one file, requiring a fair amount of jinja magic because a single file cannot have two constraints and resolve successfully into a single result as we need in our current mechanism.
- Implement changes to roles to ensure that the build process and the packages only required when building (dev headers, etc) are only used when a build is being executed. The build packages and the runtime packages will be changed into separate lists so that the runtime environment is only installing the packages it needs.
- Ensure that optional pip packages are installed into the venv during the build stage, rather than during the install stage.

Alternatives

- The build process can remain as-is, continuing to confuse deployers and difficult to maintain.
- The build process can be changed to only build and store wheels for packages which are pip installed onto the hosts, and only to build and store the venvs for distribution.

Playbook/Role impact

Playbooks will be added to cater for the build process and the staging process. The roles will be adjusted to properly separate out the build tasks and the distro packages to install for the build (versus those required when using pre-built wheels).

Upgrade impact

Care will be taken to ensure that upgrades happen as they do today.

Security impact

The security posture should be improved by the reduction of packages installed onto hosts and containers when a full set of artifacts are built.

Performance impact

The performance of the deployment should be improved due to the reduction in time taken to deploy with pre-built packages if a full set of artifacts are built.

End user impact

There is no end-user impact for consumers of an OpenStack cloud, except perhaps that upgrades will be quicker to execute, thus resulting in reduced maintenance slot requirements.

Deployer impact

- As deployments and upgrades will be quicker to execute, deployers will be able to execute them in shorter maintenance slots.
- Deployers will need to understand how better to utilise the CI process to prepare the required artifacts to speed up deployments.

Developer impact

As the build process will be integrated into the roles, it will be easier to understand how it works and what it does.

Dependencies

This spec will be implemented in partnership with https://blueprints.launchpad.net/openstack-ansible/+spec/deployment-stages

7.10.3 Implementation

Assignee(s)

Primary assignee:
jesse-pretorius (odyssey4me)

Work items

Each of the roles implemented in the default AIO will be worked through in sequence to re-arrange and optimise based on this workflow. The work items are not being detailed here but will be reflected in gerrit through the blueprints topic and will be visible in launchpad.

7.10.4 Testing

As this process matures, it may be simpler to use the integrated build for all role testing instead of having two seperate test implementations. This reduces technical debt for the project.

7.10.5 Documentation impact

This work will need to include documentation updates which describe the new way that deployments can be implemented using full artifact builds and how to implement offline installs.

7.10.6 References

- https://12factor.net/
- http://www.clearlytech.com/2014/01/04/12-factor-apps-plain-english/

CHAPTER

EIGHT

PIKE SPECIFICATIONS

8.1 Use dnf with CentOS

date
2017-07-28 00:00

tags
centos, dnf, packaging

Blueprint: Use dnf with CentOS

CentOS 7 currently uses `yum` as its default package manager. However, Fedora has moved to `dnf` for several releases and it provides significant performance benefits. It can make the metadata cache, evaluate dependencies, and handle fastest mirror checks much more efficiently.

The `dnf` and `yum` package managers can co-exist together without causing conflicts. Several Fedora releases ran both of these simultaneously. The `dnf` packages are available in the EPEL repositories (which we currently enable). It uses all of the existing `yum` repositories and GPG keys as well.

8.1.1 Problem description

The CentOS gate jobs are notoriously slow and the integrated gate times out on tempest runs frequently. The longest running tasks in each role involve the installation of distro packages because these tasks use `state: latest` the `yum` tasks.

When Ansible sees `state: latest`, it goes through a fairly tedious process:

- Run `check-update`, which checks the **entire** system for updates.
- If some packages are returned (they need updates), Ansible searches the list to see if any packages from the `yum` task are in that list.
- If some packages need updates, Ansible calls `yum` to install those packages.

This process can take 5-8 seconds even for *one* package. In comparison, `dnf` completes the task in 0.8-1.6 seconds. This should give us some wiggle room to get CI jobs completed sooner and convert more of the CentOS jobs from non-voting to voting.

8.1.2 Proposed change

On CentOS systems, we should install `dnf` and `python-dnf` (for Ansible compatibility). Ansible will prefer `dnf` over `yum`, so we would need to ensure that each role has support for `dnf` tasks. Since both package managers are interchangeable, this could be done by symlinking the `*_install_dnf.yml` task files to `*_install_yum.yml` and using the `package` module in those task files.

Alternatives

If `dnf` isnt preferred, we could avoid using `state: latest` for CentOS installations. This would cause CentOS deployments to diverge from Ubuntu and OpenSUSE deployments and it would make bug triage more challenging.

Another option is to update the entire system when `state: latest` is provided but switch all of the package installation tasks to use `state: present`. This will save us a small amount of time since Ansible will skip the `check-update` step and go straight into updating all packages. This would be another diversion from the Ubuntu/OpenSUSE process, however.

Playbook/Role impact

Each role with a set of `yum` tasks would need to be converted to use `package`. A symlink would be needed so that CentOS systems with `dnf` installed would use the same tasks.

Upgrade impact

During the upgrade process, `dnf` would be installed on CentOS systems. Ansible would begin to use `dnf`, but the deployer could continue using `yum` for their own administration tasks if they prefer it.

Security impact

The `dnf` package manager supports the same configuration options as yum for checking GPG keys of packages and repositories.

Performance impact

The `dnf` package manager will provide better performance when managing packages, but the rest of the system will perform at the same levels.

End user impact

End users will not notice this change or gain any benefits from it.

Deployer impact

Deployers may notice that some roles use `dnf` while others use `yum` until all of the patches have merged. This wont affect the running system, but it may make some playbooks faster than others.

Deployers would continue to deploy in the same ways that they currently do today.

Developer impact

Developers must be aware that `dnf` is present on CentOS systems and that Ansible will prefer it over `yum`. Any new roles/playbooks or updates to existing ones will need to include support for `dnf` via the `dnf` module or the `package` module (which selects `dnf` over `yum` already).

Dependencies

This spec is not dependent on any other spec or blueprint.

8.1.3 Implementation

Assignee(s)

Primary assignee:
Major Hayden (IRC: mhayden, Launchpad: rackerhacker)

Work items

- Add `dnf` patches to the base roles first (openstack_hosts, lxc_hosts, etc)
- Continue moving up the dependent roles until all roles include `dnf`-compatible tasks
- Ensure that the integrated repository and openstack-ansible-tasks use `dnf`

8.1.4 Testing

The existing testing done in the OpenStack CI jobs will be sufficient for this work. If `dnf` is not installing packages properly or efficiently, we will see that reflected in the testing playbooks.

8.1.5 Documentation impact

This work will require some release notes to notify developers and deployers of the `dnf` change. However, theres no need for extensive documentation since `dnf` supports the same configurations and arguments as `yum`.

8.1.6 References

- Test patch for openstack-ansible-openstack_hosts: https://review.openstack.org/488268
- Vultr docs for dnf on CentOS 7: https://www.vultr.com/docs/use-dnf-to-manage-software-packages-on-centos-7

8.2 Pluggable Inventory Backends

date
2016-12-13 22:00

tags
inventory, craton

This spec is intended to provide guidance and a longer term goal for migrating the existing inventory system from a single, coupled filesystem interface to a pluggable system supporting multiple backends for storage.

- https://blueprints.launchpad.net/openstack-ansible/+spec/inventory-pluggable-backends

Currently, the generated inventory is kept in one place - the `/etc/openstack_deploy/openstack_inventory.json` file on the deployment node. While this has worked, it is not very robust. In order to accommodate more deployer flexibility, the inventory system should be reworked to use a pluggable system for storing necessary info. A filesystem plugin would provide backwards compatibility, and a Craton plugin will also be developed.

8.2.1 Problem description

While there are multiple issues to be addressed in the current codebase, this spec focuses solely on the storage of inventory facts.

Since OpenStack-Ansibles creation in Icehouse, the source of truth for a completed inventory has been the `/etc/openstack_deploy/openstack_inventory.json` file. While this has worked, it has a few drawbacks:

- As a single file, it may be deleted by accident. If the configuration files have changed, getting an exact copy back without the tar backup files is impossible due to UUID suffixes.
- There are only UNIX file permissions managing access to the file.
- No accounting of changes, besides a simple tar backup, exists. This is useful for documentation and auditing of a running cluster.

This spec does not aim to define a fully robust inventory management system. Instead, the OpenStack-Ansible inventory system can be made more modular, especially in its storage, so that deployers can take advantage of other, dedicated systems.

Doing so requires refactoring the dynamic inventory generation code so that storage concerns, such as writing the output and reading output from previous runs, are no longer directly tied to the generation of values.

8.2.2 Proposed change

Any code that interfaces with the filesystem currently will be moved into its own Python module, so that it exists separate from the generating logic. This work is already happening, and doesnt rely on a particular plugin system in order to be completed.

Once that is done, a plugin system will be used to provide a generic interface for storage actions to the rest of the codebase. This code will connect to the plugin library, and return a compatible instance of a storage plugin.

Plugin Python API

Plugins should support the following public methods:

register
: registers a plugin with the plugin system. Receives a dictionary of all arguments specified for plugins, and should extract any information it needs, such as file locations, connection strings, or URLs.

load
: loads data from source into a Python dictionary in the current dynamic inventory scripts. Source may be a file, a database, or any other backend system.

write
: Writes the inventory dictionary to the specified backend. Receives the inventory dictionary as an argument

Configuration Changes

Some new configuration will likely need to be presented to the user, so that the appropriate plugins for storage can be identified. This configuration is not well suited to the `etc/openstack_deploy/openstack_user_config_file`, since part of the scripts job entails accessing that file.

Instead, the `/etc/ansible/osa.ini` file is proposed, to match the style ec2.py scripts method laid out in Ansibles dynamic inventory documentation.

The specific format of this file will be left to implementation reviews, however at a high level it will likely include a Python import path to the inventory module to use, and any settings it may need such as connection strings or API URLs.

Alternatives

Nothing could be done and the inventory could continue to rely on the JSON backend; its worked reliably until now. However, its inherent problems would remain.

In the current state, deployers must fork the repository to modify the dynamic inventory code, which is a maintenance burden long term, likely unsustainable.

Alternate inventory scripts could also be placed in `playbooks/inventory`, either replacing or adding to the current `dynamic_inventory.py` file. Replacements may or may not use the current `openstack_user_config.yml` file and environment structure. Additional scripts would produce output merged from all sources, read using `os.listdir` and then processed according to alphanumerical file name.(see the Ansible script loader).

Configuration Alternatives

Rather than the `/etc/ansible/osa.ini` file, environment variables could be used to set plugin options.

Plugin Implementation Alternatives

Existing plugin implementations include PluginBase and Stevedore.

Stevedore is an OpenStack project, and using it would align the project more closely with the community. Stevedore requires registering plugins via *setup.py entry_points*. The `setup.cfg` for OpenStack-Ansible needs modifications to install the python code, and the `lib` directory needs to be renamed in order to work within pbrs design. A *prototype review <https://review.openstack.org/#/c/418076/>* for these changes has been submitted.

PluginBase is not directly part of the OpenStack ecosystem, but is simple, standalone plugin system. It has no external dependencies, and can be used via the standard Python import system without requiring a *setup.py* file for our existing code.

The PluginBase method is less impactful for in-tree code, however any external plugins should be pip-installable anyway, thus having a *setup.py*. Therefore, either is viable as an implementation option, with Stevedore requiring slightly more upfront work.

Playbook/Role impact

Ideally, playbook impact should be minimal or non-existent. The inventory generated should look the same from a playbook perspective, regardless of backend plugins used.

Upgrade impact

A migration path from the existing JSON source should be provided, so that existing environments can move to new systems, should they choose. However, migrating will likely involve implementation detail knowledge that differs per system, so each one will likely need to have its own import functionality.

An export function has already been implemented to provide the inventory in a per-host format. This could be used as a basis for external systems to use in their own import systems.

This export/import process is assumed to be out-of-band from the playbook runs.

Security impact

This change introduces communication to outside systems - there is inherent risk in doing so. These systems are assumed to be using secure channels and trusted by the deployers.

Secrets could theoretically be stored in these backends, though the `openstack-ansible.sh` wrapper script currently references the `user_secrets.yml` file instead of placing those in the inventory. The system should not dictate that this is the only solution for secrets, however. Where deployers choose to put these is up to them, though storing secrets in any sort of unencrypted or unprotected backend is not advised.

Performance impact

Inventory may take longer to generate or look up depending on the system used as a backend. If said system is used via a network interface, latency and caching are concerns.

Since this category is fairly broad and different systems will have different characteristics, more detail is best left to specific plugin implementations.

End user impact

Users of the deployed clouds, those consuming virtual machines and networks, should not see much of a difference. This change is largely to facilitate deployer concerns.

Deployer impact

Deployers may have entirely new inventory backend sources. Configuration options for reading from said sources would have to be provided. The default, in-tree implementation is likely to remain the JSON file for the time being.

Changes to existing clusters will require deployer intervention to migrate relevant data from the file into their new system, which may or may not be managed externally to an OpenStack-Ansible deployment.

Any helper scripts that relied on the `openstack_inventory.json` file will need to be modified, preferably to take advantage of the new plugins/APIs.

The `inventory-manage.py` script currently only provides a management interface for the JSON file, and is not intended to be a universal inventory management tool. Different systems will have their own clients or front ends for doing such management and querying.

Developer impact

Developers of roles should be able to rely on the inventory information staying the same.

Developers working on the inventory generation must account for multiple backend sources of data, however the intention is to provide a uniform API for working with that data.

Dependencies

While not a strict dependency, this is closely related to the dynamic inventory lib blueprint/spec. It specifically tries to solve the problem of external backends.

8.2.3 Implementation

Assignee(s)

Primary assignee:
nolan-brubaker (IRC: palendae)

Other contributors:
steve-lewis (IRC: stevelle)

Work items

- Implement existing code as a separate module. This work has largely been done, see code reviews *https://review.openstack.org/#/c/392056/*, *https://review.openstack.org/392417*, and *https://review.openstack.org/399303*.
- Implement plugin system. Whether this is Stevedore or PluginBase, the code for interfacing with the system will need to be written.
- Implement the file system code as a plugin. This may be done in tandem with the previous item in order to fully test it.

8.2.4 Testing

Unit and integration tests should be written to ensure that the existing JSON code continues to work. Also, sample plugins should be written to exercise the system, even if they are dummy systems.

Since this fits into inventory tests, it should not affect the integrated gate build times. It should also be tested per-commit.

Testing will also be implicit in the integrated build, but not necessarily targeted for easy troubleshooting.

Individual plugins external to this repo will need to be gated separately.

8.2.5 Documentation impact

Guidance on writing plugins and migrating to new systems should be provided

8.2.6 References

This spec has been informed by discussions on etherpads such as:

- *https://etherpad.openstack.org/p/osa-dynamicinventory-plugins*
- *https://etherpad.openstack.org/p/craton_osa*
- *https://etherpad.openstack.org/p/openstack-ansible-newton-dynamic-inventory*

8.3 Monitoring for an OpenStack-Ansible deployment

date
: 2017-02-21 00:00

tags
: monitoring, operations

Blueprint on Launchpad:

- https://blueprints.launchpad.net/openstack-ansible/+spec/monitorstack

The goal of the efforts described in this spec is to provide an easy method for monitoring an OpenStack cloud. This would initially include basic service state monitoring with extra functionality added as it matures.

8.3.1 Problem description

OpenStack clouds are complex systems of hardware, software, and networks. Deployers need to monitor the health of all of these components to ensure that end users have access to resources. OpenStack-Ansible does not offer any components for monitoring at this time, and this forces deployers to build their own monitoring plugins and tool stacks.

Deployers and operators need to know:

- Are my OpenStack services up or down?
- Are my additional services (Galera, RabbitMQ, etc) up or down?
- What is the state of cluster partitions for Galera and RabbitMQ?
- Are my APIs responding within a reasonable time period?
- Are my management and tenant networks accessible?
- Is the hardware underneath my cloud operating normally?

8.3.2 Proposed change

The proposed changes fall into two main buckets:

Monitoring plugins
: This is the **primary** work effort for the spec.

  Deployers need a solid set of monitoring plugins that gather information from various services or entities, and those plugins should output data in common formats for the most popular monitoring tool stacks.

Monitoring tool stack
: This is the **secondary** work effort for the spec.

  There are many open source and commercially available monitoring tool stacks available for Linux. Deployers should have the option to deploy an opinionated tool stack via Ansible if they dont have one of their own already. The tool stack should offer up its time series data for searching and also have an alerting mechanism that can hook into a deployers existing notification tools.

Alternatives

There are loose collections of monitoring plugins available within OpenStack, but those plugins arent being actively maintained. Many of the other plugin sets available today only output their data in a specific format. Deployers could choose to use these plugins instead.

Deployers could also deploy their own monitoring tool stacks if needed. They could use the monitoring plugins created in this spec with their existing tools.

Playbook/Role impact

The monitoring plugins should be installable via pip and they can be added into existing roles or playbooks *(perhaps the openstack-ansible-openstack_hosts role)*. The plugins themselves should be released independently of an OpenStack release.

The monitoring tool stack would be implemented in a new role with an additional playbook. The role that deploys this stack would be versioned along with OpenStack-Ansible releases so that it can utilize the existing variables and modules from each release.

Upgrade impact

This would be the first implementation of monitoring plugins and tools in OpenStack-Ansible, so there is no upgrade concern at the moment. However, the plugins and tool stack installation should be written such that upgrades are reliable.

Security impact

Some monitoring plugins will need some level of privileged access to OpenStack services or the other services running in the cloud. This requires accounts to be created and new secrets to be stored. It is possible to use accounts that have fewer privileges so that a compromise of a monitoring plugin would have a limited security impact.

The monitoring tool stack itself has important security concerns to address. Data from the monitoring plugins running on each host or container must be able to reach a centralized database for storage and processing. Access to any web frontends or databases should be handled carefully, just as we do for Horizon or Galera today.

Performance impact

Some monitoring plugins will need to make requests to OpenStack APIs or access certain other services. These plugins must be written carefully to avoid negative performance impacts on the system.

End user impact

End users should not notice the changes from this work.

However, they should get a better user experience if the environment is closely monitored and operations teams have access to valuable performance data.

Deployer impact

The monitoring plugins should be distributed as a pip package, so this should have a small impact on deployers. Some plugins will need accounts on the system, so deployers will need to create additional secrets for those accounts.

Deployers would have the option to deploy the monitoring tool stack if they do not have one of their own.

Developer impact

The developer impact from these changes is very low. The monitoring plugins should be easy to use and heavily tested. Developers should be able to modify existing plugins and create new ones with ease.

Dependencies

There are no dependencies.

8.3.3 Implementation

Assignee(s)

Primary assignee:
Major Hayden (mhayden)

Other contributors:
Kevin Carter (cloudnull) Antony Messerli (antonym)

Work items

1. Write a small class that can be extended for new monitoring plugins.
2. Begin writing monitoring plugins that are executable via setuptools entry points.
3. Ensure that tests are available for each plugin as well as the base class.
4. Create a role to deploy a monitoring tool stack that uses these plugins.
5. Document the plugins and the tool stack.

8.3.4 Testing

The monitoring plugins should be tested on each commit using tox.

The monitoring tool stack role should be tested independently (like the other IRR repos) and added to the integrated build as an optional component.

8.3.5 Documentation impact

The plugins should be documented and there should be developer guides that explain how to modify existing plugins or add new ones. The monitoring tool stack role will need documentation that explains the new variables and functionality available.

8.3.6 References

Notes from the OpenStack PTG in Atlanta (Feb 2017):

- https://etherpad.openstack.org/p/osa-ptg-pike-monitoring

8.4 Integration of OpenDaylight SDN controller with Neutron

date
2017-06-08 11:00

tags
opendaylight, neutron, SDN, openvswitch

Blueprint on Launchpad:

- https://blueprints.launchpad.net/openstack-ansible/+spec/opendaylight

This spec introduces the work required to have the OpenDaylight (ODL) SDN controller deployed as a Neutron backend, using networking-odl ML2 mechanism driver to connect to Neutron.

This spec also covers the connection of OpenvSwitch (OvS) to OpenDaylight.

8.4.1 Problem description

OpenStack networking (Neutron) uses a modular approach that allows different backends to be used, by means of mechanism drivers. Although Neutron can handle simple deployments, more advanced networking capabilities are better addressed with an advanced SDN controller, such as OpenDaylight.

8.4.2 Proposed change

The proposed change consists on using the existing OpenDaylight Ansible role and optionally deploying it along with Neutron. The connection of OpenDaylight and Neutron is done by using the networking-odl ML2 mechanism driver, and configuring Neutron to use it.

After OpenDaylight and networking-odl are installed, some configuration is required in `ml2.ini` file to instruct Neutron to use the mechanism driver, as well as setup OpenDaylights endpoint and credentials.

Final step consists on connecting the traffic forwarding elements with the OpenDaylight controller. This spec will use OpenvSwitch for this task. Also, neutron-openvswitch-agent needs to be stopped and disabled, as OpenDaylight is the responsible for data plane management.

Alternatives

There are other networking backend for Neutron already available with Openstack-Ansible, namely Calico (https://www.projectcalico.org/tag/openstack/) and DragonFlow (https://wiki.openstack.org/wiki/Dragonflow). These backends are optionally deployed depending on the ML2 configuration that is passed to `os_neutron` Ansible role.

Playbook/Role impact

The os_neutron role will be modified to optionally deploy OpenDaylight. The proposal is to use the same approach as DragonFlow: when the `neutron_plugin_type` variable is set to `ml2.opendaylight`, OpenDaylight would be used as Neutron backend. There will be a new taskfile, `neutron_opendaylight_setup.yml`, which would be included in os_neutrons playbook when the above condition is fulfilled.

The OpenvSwitch scenario would be leveraged to work with OpenDaylight, being OvS the only switch supported in the first release.

Upgrade impact

This is the first implementation of OpenDaylight with Openstack-Ansible, so no upgrade concerns yet.

Security impact

Networking-odl configuration requires the setup of a username and password for northbound auhentication towards OpenDaylight. The deployer should be able to configure those credentials.

Communication between the controller and the switches will not be secured by default. Using TLS to secure the communications is considered a stretch goal, and deployers need to consider this security implication, specially in production environments. For more information on secure communications between OpenDaylight and OpenvSwitch, see the *References*.

Performance impact

For those choosing to opt-in this deployment method, some extra packages need to be installed on the system, which would make installation last a bit longer.

Extra resources are needed to run the OpenDaylight SDN controller on the system as well. However, performance in Neutron API calls should be minimum.

End user impact

End users would have a new networking API available through OpenDaylight. This would enable them to create advanced networking scenarios (e.g. Service Function Chaining). This will require some documentation with troubleshooting steps to verify that OpenDaylight is working properly, as well as pointers to OpenDaylights official documentation.

No changes to Horizon or other OpenStack components are expected.

Deployer impact

New artifacts are being deployed, namely the Karaf runtime for OpenDaylight, and the networking-odl pip package. OpenDaylight requires around 2.5G of RAM to work properly, with OpenStack, that would need to be considered when dimensioning the host where it will run.

Also deployers need to ensure that OpenvSwitch is deployed in all networking nodes, namely compute hosts and hosts where neutron agents are running.

Developer impact

Developer impact is very low, all tasks for OpenDaylight deployment will be optional and can be ignored when extending or modifying Neutron role.

Dependencies

There are no dependencies

8.4.3 Implementation

Assignee(s)

Primary assignee:
Juan Vidal (jvidal)

Other contributors:
Fatih Degirmenci (fdegir) Daniel Farrell (dfarrell07)

Work items

1. Install OpenDaylight SDN controller
2. Configure Neutron to use OpenDaylight
3. Deploy and configure OpenvSwitch to work with OpenDaylight
4. Set OpenDaylight as OpenvSwitch manager
5. Create a new test and verify that it passes
6. Document the new scenario

8.4.4 Testing

As a replacement of Neutron backend, this new scenario should provide the same capabilities of existing backends, so existing tests should be run.

A test specific for OpenDaylight can also be implemented, in the same way as there are currently tests for Calico or DragonFlow.

8.4.5 Documentation impact

The new scenario *OpenDaylight+OpenvSwitch* should be documented, explaining the configuration parameters required to deploy it.

8.4.6 References

Deploying OpenDaylight using Ansible:

- https://wiki.opendaylight.org/view/Deployment#Ansible_Role

Ansible role for OpenDaylight:

- https://git.opendaylight.org/gerrit/p/integration/packaging/ansible-opendaylight.git

Setting up OpenDaylight on OpenStack:

- https://wiki.opendaylight.org/view/OpenStack_and_OpenDaylight

Networking-odl mechanism driver:

- https://github.com/openstack/networking-odl

Networking-odl installation and configuration:

- https://docs.openstack.org/developer/networking-odl/installation.html

OpenvSwitch scenario with Openstack-Ansible:

- https://docs.openstack.org/developer/openstack-ansible-os_neutron/app-openvswitch.html

TLS Support on OpenDaylight OpenFlow plugin:

- https://wiki.opendaylight.org/view/OpenDaylight_OpenFlow_Plugin:_TLS_Support

Secure Communication Between OpenFlow Switches and Controllers

- https://www.thinkmind.org/download.php?articleid=afin_2015_2_30_40047

8.5 Openvswitch with NSH support in Neutron

date
: 2017-06-21 15:00

tags
: Openvswitch,neutron,SFC,NSH

Blueprint on Launchpad

- https://blueprints.launchpad.net/openstack-ansible/+spec/openvswitch-with-nsh-support

This spec introduces the work required to have Open vSwitch with NSH protocol support which is used in Service Function Chaining.

8.5.1 Problem description

According to * https://datatracker.ietf.org/doc/draft-ietf-sfc-nsh/ Network Service Header (NSH) is inserted to a packet or a frame to realize service functions paths. Also provides a mechanism for metadata exchange along the instantiated service path. The NSH protocol is used as an SFC encapsulation which is required for the support of the Service Function Chaining (SFC) Architecture as it defined in RFC7665.

The Openvswitch currently doesnt support the NSH protocol. So the only way to add NSH support to Open vSwitch is through Yi Yangs patches (https://github.com/yyang13/ovs_nsh_patches).

8.5.2 Proposed change

The proposed change is the use of the existing Neutron Ansible Role for the installation of Open vSwitch with NSH support when the user selects that functionality through specific configuration in Openstack-Ansible project. We intent to configure only Neutron component and not use the aforementioned functionality for end to end testing.

The installation of Open vSwitch with NSH support will be addressed by the use of specific packages which are going to be maintained in private repositories unti the NSH functionality will be included in a subsequent release of Open vSwitch project.

8.5.3 Alternatives

An alternative to create a SFC without NSH is the port chaining technique. The aforementioned technique uses Neutron ports to steer the traffic to a service chain and has no notion of the actual services which are attached to those Neutron ports.

Playbook/Role impact

The os_neutron role will be modified to optionally install Open vSwitch with NSH support. The proposal is to add an extra variable so the user can decide whether or not he needs to add NSH support with the Open vSwitch installation. When the `neutron_plugin_type` variable is set to `ml2.ovs` or `ml2.dragonflow` and the `ovs_nsh_support` variable is set to `true` then the Open vSwitch will be installed with NSH support. So there will be an extra task in the `neutron_pre_install.yml` which will add the distribution specific repositories with the ovs_nsh packages.

Upgrade impact

This is the first implementation of Open vSwitch with NSH support in OpenStack-Ansible,so no upgrade concerns yet.

Security impact

No security impact

Performance impact

The added NSH support to Open vSwitch will not have any performance impact to the current OpenStack-Ansible installation because the system will need to install only some extra packages.

End user impact

The end users will have the capability to create service function chains with the use of the NSH protocol. Also they can use OpenDaylight as networking backend which via the `sfc` component supports the creation of SFCs through the NSH protocol.

Deployer impact

The deployer needs to ensure that the specific repositories which hold the ovs_nsh packages are added to the system and the proper Open vSwitch packages are installed.

Developer impact

The developer impact is really low because the NSH support for Open vSwitch is optional and can be ignored when extending or modifying Neutron role.

Dependencies

There are no dependencies

8.5.4 Implementation

Assignee(s)

Primary assignee:
Dimitrios Markou (mardim)

Work items

1. Add specific PPA for ovs_nsh packages
2. Install Open vSwitch with NSH protocol suppport
3. Document the new functionality

8.5.5 Testing

Existing tests should be run because the only thing that change is that the installation of Open vSwitch is managed by specific repositories when NSH support is selected.

8.5.6 Documentation impact

The new functionality *Open vSwitch with NSH support* should be documented, explaining the required configuration parameters which are necessary for this deployment.

8.5.7 References

Open vSwitch scenario with OpenStack-Ansible

- https://docs.openstack.org/openstack-ansible-os_neutron/latest/app-openvswitch.html

NSH ietf draft

- https://datatracker.ietf.org/doc/draft-ietf-sfc-nsh/

SFC RFC 7665

- https://tools.ietf.org/html/rfc7665

PPA for Openvswitch-NSH packages

- https://launchpad.net/~mardim/+archive/ubuntu/mardim-ppa

Openvswitch-NSH packages for Centos

- https://copr.fedorainfracloud.org/coprs/mardim/openvswitch-nsh/

8.6 Replace IP Generation Code

date
: 2017-1-11 22:00

tags
: inventory, ip, networking

The current inventory code uses a simple set to manage assigned IPs (`USED_IPS`) and complex queues to pull from the available subnets.

This code can be simplified and made more modular.

Launchpad blueprint:

- https://blueprints.launchpad.net/openstack-ansible/+spec/replace-ip-generation

The current IP generation code is tightly couple to the configuration loading, writing, and inventory manipulation code. To help provide better, more focused test coverage, this code can be updated and replaced.

8.6.1 Problem description

The current IP generation code is difficult to maintain, despite mostly being moved into a separate `ip.py` module. The code uses the external `Queue` class, which is slightly more complex than necessary. The `USED_IPS` set and the pools of available IPs are not managed together, and could easily become out-of-sync.

New code has been written to add an IPManager class, but it is not currently integrated into any other code. Such integration is a somewhat large task, and would be error-prone to do in a single review. This spec is intended to serve as a road map to guide small, focused changes towards using it.

Note that while the IPManager includes an API for external IPAM systems, this spec is only focused on using this class within the code, not on any sort of plugin system.

8.6.2 Proposed change

An initial draft of new IP management code has been written in the IPManager class.

After that, the existing `get_ip_address`, and `set_used_ips` were refactored to still use the existing data structures, but in a way that would allow usage of the new IPManager class. See review 403915.

Some refactors may be necessary for the IPManager class to facilitate this and further codify assumptions.

Alternatives

The code be left as is, with the assumption that it will be replaced wholesale by some other system in the near future. That replacement might happen via plugins or a new inventory codebase. This has not been deeply explored in the context of the IP management/generation.

One such replacement system, for example, could be using LXD to entirely manage container creation, which is where IP generation is primarily used.

Playbook/Role impact

No noticeable impact on the playbooks and roles should be seen; this is largely facilitating code maintenance and should produce the same output.

Upgrade impact

There should be no upgrade impact - the IPManager class should be loaded with the already-generated IP addresses in upgraded installations.

Security impact

This change should not affect any sensitive data. It is unrelated to secret storage.

Performance impact

Generating IPs may be slightly faster, since this approach doesnt rely on delayed access from `Queue` objects. However, the overall runtime of the inventory is negligible in the overall speed of the system and hasnt been profiled.

End user impact

This change would be invisible to users of the deployed cloud.

Deployer impact

No configuration or output changes should be introduced. The current configurations should be used as-is.

Developer impact

This should improve quality of life for developers debugging the IP generation behavior.

Dependencies

This has no direct dependencies on other blueprints or specs.

8.6.3 Implementation

Assignee(s)

Primary assignee:
nolan-brubaker, IRC: palendae

Other contributors:
steve-lewis, IRC: stevelle

Please add **IRC nicknames** where applicable.

Work items

- Refactor current IP loading/management functions to be amenable to replacing the data structures.
- Replace the data structures and update the objects being passed between functions.

8.6.4 Testing

Unit and integration tests should be added for all code changes to confirm there are no regressions.

8.6.5 Documentation impact

Developer documentation should be updated to reflect the new mechanism used, preferably included with implementation patches.

8.6.6 References

N/A

CHAPTER

NINE

OCATA SPECIFICATIONS

9.1 Create Operations Guide

date
2016-12-19 17:00

tags
docs, ops

Blueprint: Create OpenStack-Ansible Operations Guide * https://blueprints.launchpad.net/openstack-ansible/+spec/create-ops-guide

This specification proposes the development of an OpenStack-Ansible Operations Guide for the Ocata release.

9.1.1 Problem description

During the Newton development cycle, the Installation Guide was revised which focused on providing a method for installing OpenStack for a test environment and production environment. As noted in the Installation Guide spec, the operations content did not belong in the Installation Guide as it reduced the users focus to install OpenStack, and was temporarily relocated to the following Developer Documentation pages:

- http://docs.openstack.org/developer/openstack-ansible/developer-docs/ops.html
- http://docs.openstack.org/developer/openstack-ansible/developer-docs/extending.html

There is a need to develop a standalone Openstack-Ansible operations guide that will address an operators need for information on managing and configuring an OpenStack cloud using OpenStack-Ansible.

9.1.2 Proposed change

The main focus of the operations guide is to re-organise the current content and develop new content so an OpenStack operator can easily search for information on maintaining their environment, troubleshooting, and resolving issues.

The proposed changes are:

- A new ToC with input from developers and operations: https://review.openstack.org/#/c/409854/
- Removal of duplicated content from the OpenStack manuals operations guide (so that this guide focuses primarily upon OpenStack-Ansible operations).

- Structuring the guide in a runbook format for the following reasons:
  1. Ensuring the guide includes lower-level how-tos for anyone starting to operate their own cloud.
  2. Ensuring the guide includes higher-level troubleshooting information for more experienced operator.
  3. It is structured to make it easy for operators to find the information they are looking for.
- Review and update current operations content to follow the openstack-manuals documentation conventions.

Alternatives

- The current operations content and any future content will remain in the Developer Documentation.

Playbook/Role impact

N/A

Upgrade impact

N/A

Security impact

N/A

Performance impact

N/A

End user impact

These changes will improve the end user experience, by providing a more structured and better flow of information to operate your OpenStack cloud.

Deployer impact

N/A

Developer impact

N/A

Dependencies

N/A

9.1.3 Implementation

Assignee(s)

Primary assignee:
Alexandra Settle (asettle)

Other contributors:
Andy McCrae (andymccr), OpenStack-Ansible PTL Darren Chan (darrenc) Robb Romans (rromans)

Work items

- Clarify and obtain consensus on the content structure
- Gather information from SMEs as needed
- Create a draft directory for operations guide changes
- Create a work items list and allocate resources
- Ensure documentation meets openstack-manuals writing conventions
- Test draft documentation before publication

9.1.4 Testing

The testing will be conducted by the community once a draft is available. OpenStack-Ansible users will be asked to utilise the new operations guide to perform the OpenStack operations and evaluate if the information provided is accurate, clear, and concise.

9.1.5 Documentation impact

This is a documentation change, N/A.

9.1.6 References

- ToC planning
  - https://docs.google.com/document/d/1xeJ_lep7P2e7HLbRFG57Dx4W9s8brkuNIqJmOvheWKI/edit?usp=sharing
  - https://review.openstack.org/#/c/409854/

9.2 Octavia

date
2016-11-01 00:00:00

tags
lbaas, octavia, load balancer, neutron

Blueprint: Deploy Octavia (LBaaS) with OpenStack-Ansible
Link: https://blueprints.launchpad.net/openstack-ansible/+spec/octavia

The Octavia project deploys load balancers that are more scalable and resilient than the original neutron-lbaas agent-based load balancers. Octavia has a few daemons that handle the build- out, configuration, and tear-down of load balancers.

9.2.1 Problem description

There are two main load balancer offerings in OpenStack right now:

- LBaaSv2 w/agent: Uses the neutron-lbaasv2 agent with haproxy running in a namespace
- LBaaSv2 w/Octavia: Deploys load balancers into virtual machines and manages them using the LBaaSv2 API

Agent-based load balancers have scalability and reliability limitations since the haproxy instances only run in one place without failover.

Octavia offers some helpful improvements for load balancing:

- Load balancers are deployed into virtual machines, which allows them to be sized appropriately and segregates them from the control plane.
- Putting load balancers into the virtual machines brings them closer to the resources that they are balancing. This increases load balancer performance, especially in clouds where the control plane is deployed on weaker hardware than the data plane (hypervisors).
- Octavia can deploy load balancers in a highly available configuration (currently active/passive) which helps with failures as well as patching/updates.

9.2.2 Proposed change

The proposed changes would include:

- Create a role for Octavia (possibly *openstack-ansible-os_octavia*‘)
- Add Ansible code to deploy Octavia within an OpenStack-Ansible environment
- Add centralized tests for the Ansible role
- Add documentation to the role itself
- Integrate the role with OpenStack-Ansible without dislodging the existing neutron-lbaasv2 + agent support
- Allow deployers to choose LBaaSv2+agent or LBaaSv2+Octavia
- Add documentation to OpenStack-Ansibles main docs to explain how to deploy Octavia as part of the integrated build

Optionally, work could be done to enable SSL offloading support, which requires a deployment of Barbican.

Alternatives

We could keep using LBaaSv2 with the agent architecture until that code is deprecated. This is not ideal.

Playbook/Role impact

Playbooks will need to be added to OpenStack-Ansible to deploy Octavia, but this would be very similar to the existing work done for other services, like Neutron.

Upgrade impact

Octavia hasnt been deployed previously, so theres nothing to upgrade here. However, deployers who are currently using LBaaSv2+agent will have the option of changing the backend LBaaSv2 driver to use Octavia instead. They will need to delete all existing load balancers prior to making this change and recreate them.

Security impact

The main security concern is that the Octavia load balancer virtual machines will need to be on some type of management network that can be reached by Octavia services that are running within the control plane. Those virtual machines will have one network connection into a tenant network and one connection into a management network.

This could allow an attacker to move from a compromised load balancer VM into the control plane. We will need to determine some ways to mitigate those types of attacks. This could be done with iptables or other network filtering.

Performance impact

Load balancing performance should be better with Octavia-based load balancers. However, we will need to generate or download a VM image for the load balancer virtual machines. This could take time and it will need to be optimized.

End user impact

End users that already use the LBaaSv2 API wont notice a change. The API contract and endpoints remain the same. Only the backend LBaaSv2 driver will be changed.

Deployer impact

Deployers will need to enable Octavia deployments if they choose to use them. Octavia will not be deployed by default. Deployers will also need to do their capacity planning a little differently since load balancer virtual machines will take up space within the data plane that would normally be occupied by tenant virtual machines.

Developer impact

The new Octavia role will follow the same deployment/testing patterns as other roles. It should be just as approachable as other OpenStack-Ansible independent roles.

Dependencies

The work for the Octavia role has no dependencies that are unsatisfied.

9.2.3 Implementation

Assignee(s)

Primary assignee:
Major Hayden (IRC: mhayden)

Work items

See the **Proposed change** section above for an itemized list.

9.2.4 Testing

The Octavia role should use the standard centralized testing repository as other roles. Octavia will need keystone, nova, neutron, glance deployed for proper testing.

Barbican will be required for SSL offloading if that feature is enabled.

9.2.5 Documentation impact

Documentation will be needed for the role itself, as well as in the integrated repository. This documentation should match up with the docs written for other services, like neutron or nova.

9.2.6 References

Octavia wiki: https://wiki.openstack.org/wiki/Octavia Octavia roadmap: https://wiki.openstack.org/wiki/Octavia/Roadmap

CHAPTER

TEN

NEWTON SPECIFICATIONS

10.1 Add support for SystemD

date
2015-07-14

tags
systemd

The purpose of this spec is to adjust our current upstart only init process to allow us to leverage SystemD. While SystemD is not present within the Ubuntu 14.04 LTS OS that we use today it is something that is coming within the next LTS release and something that we should begin implementing as an alternative to upstart.

https://blueprints.launchpad.net/openstack-ansible/+spec/add-support-for-systemd

10.1.1 Problem description

OSAD presently only support Ubuntu 14.04 LTS using upstart. In the next LTS upstart will no longer be an option. For this reason I believe its time to begin implementing SystemD support within the OpenStack roles.

10.1.2 Proposed change

The basic change is more of a structural one. Essentially adding SystemD support will be a new template and will follow much of the same pattern found within our current upstart process.

Alternatives

n/a - SystemD is coming and the sooner we have an oppinion on it the better off we will be.

Playbook impact

The playbooks will not be impacted however the roles will have a new SystemD template and set of tasks that will enable the ability for the system to use SystemD.

Upgrade impact

Adding in SystemD support will ensure that deployers are able to upgrade to future OSs that only have SystemD available.

Security impact

n/a

Performance impact

n/a

End user impact

n/a

Deployer impact

n/a

Developer impact

n/a

Dependencies

n/a

10.1.3 Implementation

Assignee(s)

Primary assignee:
https://launchpad.net/~kevin-carter `cloudnull`

Work items

- Add SystemD templates to all OpenStack roles.
- Add SystemD tasks to all OpenStack roles.

10.1.4 Testing

Being that we do not gate on anything that uses SystemD at the moment this will be a set of changes that are being implemented to future proof OSAD. This change will also allow us to being looking into other OS support which will likely carry with it an implementation of SystemD, such as Debian Jessie.

10.1.5 Documentation impact

n/a

10.1.6 References

n/a

10.2 Gate Split

date
2015-09-07 12:00

tags
gate, mitaka

The current integration gate check relies on an All-In-One (AIO) build which is running low on resources and does not adequately test all code paths that matter for the primary use-cases of the project.

This spec outlines a proposal to switch to using multiple gate checks which are focused on testing multiple code paths that better reflect the primary use-cases.

- https://blueprints.launchpad.net/openstack-ansible/+spec/gate-split

10.2.1 Problem description

The current AIO gate check:

1. Is severely limited by the resources available in OpenStack-CIs 8 vCPU, 8GB RAM per instance. While this is adequate for basic developer testing it is not a suitable reflection of the way deployments are done for production.
2. OpenStack-CI currently only provides for single- and two-node gate checks and have specifically asked that single-node checks be used as far as possible before implementing two-node checks.
3. Does not provide adequate code path coverage. It does not test the Ceph client configuration for Glance/Cinder, the NFS client configuration for Glance/Cinder, a standalone Swift deployment, or a deployment without Swift.

4. Tries to test as much as possible in one monolithic test, making the check difficult to understand, to maintain and to diagnose faults for.

5. Fails far too often. Reducing the container affinity as tested in https://review.openstack.org/221957 has identified that the resource constraints are most likely the primary reason for the regular tempest test failures in the HP Cloud provider of OpenStack-CI..

10.2.2 Proposed change

Implement individual gate checks for OpenStack covering the following use-cases using an AIO:

1. Compute with an NFS-backed Image and Block Storage service. This is a very commonly deployed design for environments with existing storage hardware investments. This AIO would be built with the following characteristics: - An NFS service on the host - Compute service on the host - HAproxy service on the host - Cinder built in a container, configured to use the NFS service - Glance built in a container, configured to use the NFS service - Single affinity for Keystone, Horizon, Galera, Repo, RabbitMQ containers - Ceilometer and Neutron deployed as in the AIO currently

2. Compute with a Ceph-backed Image and Block Storage service. This design is becoming more and more popular for deployments. This AIO would be built with the following characteristics: - Compute service on the host - HAProxy service on the host - An simple Ceph cluster running in three containers - Cinder built in a container, configured to use the Ceph service - Glance built in a container, configured to use the Ceph service - Single affinity for Keystone, Horizon, Galera, Repo, RabbitMQ containers - Ceilometer and Neutron deployed as in the AIO currently

3. Object Storage with Keystone. This is a typical Standalone Swift design. For the sake of using the common infrastructure, we can add Glance to this for the purpose of verifying that Glance with a Swift back-end is still working correctly. This AIO would be built with the following characteristics: - HAProxy service on the host - Swift Account, Container and Object Storage on the host - Glance built in a container, configured to use Swift as a back-end - Single affinity for Keystone, Galera, Repo, RabbitMQ containers - Ceilometer deployed as in the AIO currently

4. Keystone Only. This is a specific gate test to verify the code paths for a cluster of three Keystone servers. This AIO would be built with the following characteristics: - HAProxy service on the host - 3 Keystone containers - Single affinity for Galera, Repo, RabbitMQ containers

5. Keystone with LDAP. This is a specific gate test to verify the code path for Keystone with an LDAP back-end. This AIO will be built with the following characteristics: - HAProxy service on the host - OpenLDAP on the host - 3 Keystone containers - Single affinity for Galera, Repo, RabbitMQ containers

6. Keystone with SSL. This is a specific gate test to verify the code path for Keystone with SSL enabled. This AIO will be built with the following characteristics: - HAProxy service on the host - 3 Keystone containers, with SSL enabled on Keystones Apache - Single affinity for Galera, Repo, RabbitMQ containers

7. A high availability RabbitMQ cluster. This is to test both the deployment and the availability of the cluster when its taken through a series of known failure scenarios. The details of the tests themselves would need to be clearly defined and implemented over time, so the gate check would start with what we have today - a simple test that the deployment works. The AIO would be built with the following characteristics: - Three RabbitMQ containers - A utility container for executing tests from

8. A high availability Galera cluster. This is to test both the deployment and the availability of the cluster when its taken through a series of known failure scenarios. The details of the tests themselves would need to be clearly defined and implemented over time, so the gate check would start with what we have today - a simple test that the deployment works. The AIO would be built with the following characteristics: - HAproxy service on the host - Three Galera containers - A utility container for executing tests from
9. Repo Only. This is a specific gate test to verify the code paths for a cluster of three Repo servers and to take it through a series of known failure scenarios. The details of the tests themselves would need to be clearly defined and implemented over time, so the gate check would start with what we have today - a simple test that the deployment works. This AIO would be built with the following characteristics: - HAProxy service on the host - 3 Repo containers - A utility container for executing tests from

Each use-case gate check must have reference documentation covering the design, the configuration implemented and the tests that are executed against it.

Also switch from our current lint check which combines Ansible syntax and lint checks with python pep8 checks into the following checks which, where possible, make use of the same OpenStack-CI jobs as are used by other projects:

1. bashate lint checks for bash scripts
2. pep8 lint checks for python scripts
3. Ansible syntax and lint checks for Ansible playbooks and roles

Alternatives

Leave the current gate checks as they are.

Playbook/Role impact

There will be no changes to the playbooks or roles as part of this work.

Upgrade impact

n/a

Security impact

n/a

Performance impact

n/a

End user impact

n/a

Deployer impact

n/a

Developer impact

1. More code paths will be tested.

Dependencies

In order to implement variable load balancing configuration, this work depends on: https://blueprints.launchpad.net/openstack-ansible/+spec/role-haproxy-v2

10.2.3 Implementation

Assignee(s)

Primary assignee:
https://launchpad.net/~jesse-pretorius `odyssey4me`

Other contributors:
https://launchpad.net/~hughsaunders `hughsaunders`

Work items

For each use-case:

1. Develop and document the design.
2. Implement a non-voting experimental gate check.
3. Push the code and documentation up for review and use check experimental to validate its functionality.
4. Switch the gate check to the normal check queue, leaving it as non-voting, in order to do final functional validation.
5. Switch the gate check to voting and add it to the merge queue.

10.2.4 Testing

Please see Work items.

10.2.5 Documentation impact

As indicated in the proposed change, each gate check should be properly documented for easier reference and understanding.

10.2.6 References

None.

10.3 IPv6 Project Support

date
2015-09-09 22:00

tags
ipv6

ospenstack-ansible should support IPv6 for project networks. To that effect we should make sure that the necessary components and configurations are installed so that openstack can expose and route IPv6 for project networks.

10.3.1 Problem description

Neutron currently (in kilo) has the ability to manage and route IPv6 data. OpenStack Ansible currently has a few holes in IPv6 support on Neutron tenant networks (not installing the radvd package in the neutron-agents container for instance).

10.3.2 Proposed change

Add a test case for proving IPv6 access on project networks works as expected

Alternatives

Dont explicitly support IPv6

Playbook impact

As the primary change is adding a test case this is somewhat open ended. As the support for IPv6 via Neutron is already mostly there this should be low impact, will likely only be adding the missing package and test support.

Upgrade impact

None

Security impact

Low, at the moment the only known change is to ensure that radvd is installed so that Neutron can configure/control it.

Performance impact

None

End user impact

The end user will be able to configure IPv6 in the project networks.

Deployer impact

None

Developer impact

None once spec is implemented.

Dependencies

None

10.3.3 Implementation

Assignee(s)

Primary assignee:
prometheanfire

Work items

- add test support for IPv6 in OpenStack Ansible
  - This would be via configuring a RFC4193 network and connecting from the neutron radvd namespace to the instance.
  - It would also test unicast routing between neutron networks using RFC4193.
- ensure that tests pass

10.3.4 Testing

Ensure that the instance gets an IP in a certian address space and can ping the gateway.

Test for routability, ping between instances on two neutron network segments.

10.3.5 Documentation impact

Should be minimal

10.3.6 References

https://bugs.launchpad.net/openstack-ansible/+bug/1492080

10.4 Monasca High Availability & Monasca-Agent Role

date
2015-10-07 17:00

tags
Ansible, Monasca, Monasca-agent, High Availability, Clustering

Currently, the Monasca role for Openstack-Ansible does not configure any of the services in HA. Installation of the monasca-agent into hosts and containers is also not handled.

https://blueprints.launchpad.net/openstack-ansible/+spec/monasca-ha

https://blueprints.launchpad.net/openstack-ansible/+spec/monasca-agent

10.4.1 Problem description

Most modern monitoring systems have the ability to cluster its database for resiliency and present the user interface in a highly available fashion. The Monasca role as currently defined only launches single hosts with the supporting services configured as single hosts. If you wanted to put the monasca-api behind a VIP, you would have three different apis with disparate information.

There is currently no monasca-agent Openstack-Ansible role defined. To get monasca-agent on a Openstack-Ansible installation, you have to manually install it, or use the ansible-monasca-agent playbook to install it into hosts and containers after the fact.

10.4.2 Proposed change

Notable changes:

- Monasca-api & friends
  - The following services must be placed under a VIP
  - Monasca-api
  - Influxdb
  - MySQL
  - The following services must have clustering configured:
  - MySQL
  - Influxdb
  - Storm Nimbus & Supervisor
  - Storm Nimbus & Supervisor
  - Zookeeper
  - Grafana & Monasca-ui
    - Not clustered, but for HA to work correctly on Grafana, session sharing and configuration database connection must be set to the Openstack-Ansible Galera cluster
  - Add python-monascaclient to the utility containers
- Monasca-agent
- The os_monasca-agent role must be developed to install the agent onto all hosts during the pre-install
- Improvements must be made to monasca-agent to correctly identify all services

Alternatives

Only alternative is using another monitoring system if you plan on using Openstack-Ansible, or living with the eventual doom of your monitoring data.

Playbook/Role impact

If the user does not include the os_monasca role, there will be no impact other than the inclusion of the python-monascaclient into the utility containers. If included, the user will likely have to add a few configuration options, like VIP ranges, replication factors, things that related solely to monitoring and data resiliency.

Upgrade impact

Monasca & friends work independently from Openstack-Ansible so no impact to the openstack upgrade process is expected.

If the containers are trashed during the upgrade, monasca-agent will have to be re-installed. If not, monasca-reconfigure script will have to be ran to discover changes to hosts/containers.

Security impact

- Does this change touch sensitive data such as tokens, keys, or user data?
  - Only sensitive item the playbook would need would be the admin password to enroll the monasca users & endpoints.
- Does this change alter a deployed OpenStack API in a way that may impact security, such as a new way to access sensitive information or a new way to login?
  - Monasca uses its own API and only consumes from the keystone API.
- Does this change involve cryptography or hashing?
  - No
- Does this change require the use of sudo or any elevated privileges?
  - Yes but only on the dedicated Monasca hosts to install & configure services. Monasca-agent install does not require escalation.
- Does this change involve using or parsing user-provided data? This could be directly at the API level or indirectly such as changes to a cache layer.
  - Only Monasca specific variables will be provided by users.

Performance impact

Monasca runs on its own dedicated hosts.

The impact of the monasca-agent on hosts & containers is minimal.

End user impact

Addition of monitoring as a service. No changes to existing user exposed services.

Deployer impact

- What config options are being added? Should they be more generic than proposed? Are the default values ones which will work well in real deployments?
- Only configuration options related to Monasca will be added.
- Is this a change that takes immediate effect after its merged, or is it something that has to be explicitly enabled?
- Monasca must be explicitly enabled.

- If this change is a new binary, how would it be deployed?
- N/A
- Please state anything that those doing continuous deployment, or those upgrading from the previous release, need to be aware of. Also describe any plans to deprecate configuration values or features. For example, if we change the name of a play, how do we handle deployments before the change landed? Do we have a special case in the code? Do we assume that the operator will recreate containers within the infrastructure of the cloud? Does this effect running instances within the cloud?
- Does not affect current Openstack Installation without Monasca.
- Monasca specific upgrade instructions will be provided if needed.

Developer impact

Will not affect developers not working with Monasca.

Dependencies

No blueprint dependencies.

10.4.3 Implementation

Assignee(s)

Primary assignee:
rmelero

Work items

Same as Proposed changes.

10.4.4 Testing

The os_monasca role already has testing. I will look at similar roles and determine what best tests to implement for testing the HA aspect of Monasca.

10.4.5 Documentation impact

Openstack-Ansible documentation will not be affected.

New os_monasca documentation will be written.

10.4.6 References

https://github.com/b-com/ansible-monasca

10.5 Multiple CPU Architecture Support

date
2016-06-14 12:00

tags
openstack, ansible, power

The purpose of this spec is to enable OpenStack-Ansible to deploy OpenStack in clouds with multiple CPU architectures across the nodes to be deployed.

https://blueprints.launchpad.net/openstack-ansible/+spec/multi-arch-support

The purpose of this spec is to add support for multiple CPU architectures to OpenStack-Ansible. With the introduction of OpenStack services running on other architectures such as PPC64LE, OpenStack-Ansible needs to be extended to support environments where a mixed environment of CPU architectures exists.

10.5.1 Problem description

OpenStack-Ansible was initially built to support deployments to a single architecture, primarily x86. With the extension of OpenStack and OpenStack-Ansible support to other platforms such as POWER, support for deployments running a combination of different CPU architectures is needed.

For each deployment, OpenStack-Ansible creates and builds a repo containing necessary artifacts for the OpenStack deployment. This repo holds components such as pip wheels, virtualenvs, and source trees for the different services to be deployed. For each deployment a single master repo is designated where artifacts are built, then synchronized out to the rest of the slaves.

This creates problems in a multi-architecture deployment for nodes where the repo masters CPU architecture is different than the architecture of other nodes. For example, deployment of a KVM on POWER compute node will fail when the artifacts were built on an x86 repo master used for the control plane.

10.5.2 Proposed change

Update OpenStack-Ansible and necessary roles to support building and deploying with multiple CPU architectures. This includes changes to:

- Look at the Ansible facts for all hosts and determine the set of CPU architectures to build artifacts for.
- Add support for assigning a build master for each CPU architecture.
- Support building copies of CPU architecture specific artifacts on each build master, which will be synchronized to all slaves regardless of architecture.
- Support tagging all architecture-specific build artifacts with the corresponding CPU architecture.
- Ensure pre-built binaries used during installation (apt packages, etc) exist or are made available for the supported CPU architectures (x86, ppc64el) where possible, or documentation added to note limitations where not available.

Alternatives

- Cross-compiling artifacts - This would remove the need for building, tagging and synchronizing between repos built on different architectures, but so far a reliable way to do this for wheels/venvs has not been found. This also introduces increased risk of architecture-specific cross-compiling issues as support for additional architectures is added.
- Support only single-architecture OpenStack-Ansible deployments. This is harmful to deployers by limiting the potential integration of servers from other architectures into new and existing OpenStack-Ansible deployments

Playbook/Role impact

There will be impact to the repo playbooks to handle building and synchronizing across multiple architectures. There will also be minor impact to each role to support building and tagging artifacts with the required architecture information.

Upgrade impact

N/A

Security impact

N/A

Performance impact

Both repo build and synchronization operations will take longer to complete proportionate to the number of CPU architectures being deployed.

- Repo builds will take longer as artifacts for each repo must be built serially to avoid collisions during synchronization of non-architecture specific packages.
- Synchronizations will take longer and require more bandwidth to transfer when there are multiple architectures due to the increased number of artifacts being built.

End user impact

No end user impact is expected.

Deployer impact

Deployers will now be able to create deployments with servers of multiple CPU architectures included.

No other deployer impact is expected as the detection of multiple architectures, deploy of the required number of repo masters for each architecture, and build of artifacts for each architecture should happen dynamically.

Developer impact

Developer impact will be minor. Developers will now be able to develop for and test deployments across multiple CPU architectures. It also enables development of roles specific to other CPU architectures in the future, such as ARM.

Dependencies

N/A

10.5.3 Implementation

Assignee(s)

Primary assignee:
ashana@us.ibm.com/ashana

Other contributors:
adreznec@us.ibm.com/adreznec thorst@us.ibm.com/thorst

Work items

- Add a repo discovery task to determine the CPU architectures used across all hosts, and assign and store facts about the master for each architecture.
- Modify the repo-server playbook to support deploying a build master for each CPU architecture.
- Modify the repo-build playbook to support building repos for each CPU architecture in serial.
- Update the build tasks to support tagging each artifact with a corresponding CPU architecture.
- Update the repo synchronization to support synchronizing artifacts from each build master out to all other slaves, regardless of CPU architecture.
- Updates across tasks to use the new tagged artifact names.
- Ensure binary packages are available on all package mirrors/locations for ppc64el in addition to i386/amd64, or that documentation is added to note where packages arent available.

10.5.4 Testing

A new test job will be added that deploys a two-node configuration, with one node belonging to the upstream/default architecture (x86) and the other of an additional CPU architecture to be supported (ppc64el, arm64, etc).

10.5.5 Documentation impact

Documentation covering how to configure multi-arch support will be added to the user guide.

10.5.6 References

- http://specs.openstack.org/openstack/openstack-ansible-specs/specs/newton/powervm-virt-driver.html

10.6 Only support venv installs

date
2016-06-27 13:30

tags
python, venv, deployment

The purpose of this spec is remove support for installing OpenStack services and dependent pip packages outside of Python virtual environments.

- https://blueprints.launchpad.net/openstack-ansible/+spec/only-install-venvs

10.6.1 Problem description

Conflicts between system packages and globally installed Python pip packages can lead to broken services and strange behavior. The default installation option of OpenStack services since the Liberty release has been to use virtual environments to isolate each individual service. This should be the only supported option going forward.

10.6.2 Proposed change

Each role will be updated to remove tasks and variables related to allowing the option of installing pip packages outside of a virtual environment. The tasks which currently handle installing virtual environments will also be updated to ensure that they are idempotent and can recover properly from an interruption in a previous run of the same role.

Alternatives

Leave the roles as they are. Deployment of OpenStack services would continue being supported through either virtual environments or installed as global system Python packages.

Playbook/Role impact

See *Proposed change*.

Upgrade impact

Installing services to virtual environments has been the default since the Liberty release. If any Mitaka deployments are still configured to not install services to virtual environments, they will be forced to beginning in the Newton release.

Security impact

N/A.

Performance impact

Tasks which are currently being skipped will be removed, which could slightly decrease role run times.

End user impact

N/A.

Deployer impact

The `*_venv_enabled` variables will no longer exist and will have no effect if set by a deployer.

Developer impact

N/A.

Dependencies

N/A.

10.6.3 Implementation

Assignee(s)

Primary assignee:
https://launchpad.net/~jimmy-mccrory (jmccrory)

Work items

- Remove tasks related to installation of pip packages outside of a venv from each role
- Remove variables which currently toggle installation of pip packages to a venv from each role
- Update each role to make tasks which create and install packages to a venv more resilient and idempotent

10.6.4 Testing

Both integrated and independent role gate testing are already only installing services to virtual environments.

10.6.5 Documentation impact

Should be minimal.

10.6.6 References

N/A.

10.7 Overhaul of the current OpenStack-Ansible Installation Guide

date
2016-05-31 00:00

tags
docs

Blueprint: Overhaul of the current OpenStack-Ansible Installation Guide

- https://blueprints.launchpad.net/openstack-ansible/+spec/install-guide
- https://blueprints.launchpad.net/openstack-ansible/+spec/osa-install-guide-overhaul

After the 2016 Austin summit, there was a discussion and a consensus surrounding the current state of the OpenStack-Ansible Installation Guide.

Note: A blueprint and spec were previously created with the intention of improving the documentation that pushed the summit discussion.

Currently, the OpenStack-Ansible install guide has minimal installation information, and a lot of configuration information. This specification proposes a more formalized plan to separate this information and streamline the installation guide to make it easier and quicker to install OpenStack.

10.7.1 Problem description

The OpenStack-Ansible Installation Guide contains information that does not necessarily pertain to that of an installation guide structure. It has accumulated a lot of configuration information and reference information that reduces the users focus and simplicity to install OpenStack.

The current installation guide also does not follow the openstack-manuals documentation conventions.

10.7.2 Proposed change

The main focus of the installation guide is reorganising and developing content so a deployer makes very few decisions and minimal configuration to deploy an OpenStack test environment and production environment.

The proposed changes are:

- Clearly define reference architecture and develop use case configuration examples in an appendix.
- Removal of the configuration information from the current installation guide and including it in the OpenStack-Ansible role documentation.
- Migrate operations content temporarily to openstack-ansible-ops repo until an operations guide can be produced.
- Restructure the guide to include basic deployment configuration.
- Appendices that include configuration file examples, neutron plugins, cinder options and additional resources relevant to an OpenStack-Ansible installation.
- Include links to role based documentation from the Installation Guide.

Alternatives

- Leaving the installation guide as is, and migrating only the configuration information to the developer docs.
- Consider revising the installation guide to meet criteria in project-specific installation guide and publish to docs.openstack.org

Playbook/Role impact

N/A

Upgrade impact

N/A

Security impact

N/A

Performance impact

N/A

End user impact

These changes will hopefully improve the end user experience, by providing a more structured and better flow of information to install OpenStack.

Deployer impact

N/A

Developer impact

Move existing content over to the roles first, then developers must submit any new documentation to the role repositories.

Dependencies

N/A

10.7.3 Implementation

Assignee(s)

Primary assignee:
Alexandra Settle (asettle)

Other contributors:
Darren Chan (darrenc), Jesse Pretorius (odyssey4me), Travis Truman (automagically), Major Hayden (mhayden)

Work items

- Clarify and obtain consensus on the content structure
- Gather information from SMEs as needed
- Create a draft directory for installation guide changes
- Create a work items list and allocate resources
- Ensure documentation meets openstack-manuals writing conventions
- Test draft documentation before publication

10.7.4 Testing

The testing will be conducted by the community once a draft is available. OpenStack-Ansible users will be asked to follow the new installation guide to install OpenStack and evaluate if the information provided is accurate, clear, and concise.

10.7.5 Documentation impact

This is a documentation change, N/A.

10.7.6 References

- Design Summit discussion
- ToC planning

10.8 Support PowerVM Virt Driver

date
: 2016-03-18 14:45

tags
: ansible, powervm

The purpose of this spec is to add support for the PowerVM compute platform to OpenStack-Ansible. This will enable deployment of PowerVM systems as OpenStack compute nodes alongside the core OpenStack components.

https://blueprints.launchpad.net/openstack-ansible/+spec/powervm-virt-driver

10.8.1 Problem description

The PowerVM Compute Driver[1] is currently an out of tree, OpenStack compliant Nova Driver. The Nova compute team has asked for us to grow our usage before inclusion into the main Nova project. Our potential users have cited to us that they need OpenStack-Ansible support to bring PowerVM into their environments.

Bringing this support for the PowerVM driver expands the number of operators that will make use of the compute driver. It also helps grow the usage so that it meets the requirements from the Nova team for inclusion into the main source tree.

Openstack-Ansible supports provisioning kvm/qemu compute nodes, and is introducing support for other virt driver types such as Ironic. This blueprint would add support for the PowerVM Nova Virt Driver.

10.8.2 Proposed change

The PowerVM platform runs virtualization and management resources in a VM. This privileged VM has authority to manage the system and is where the nova-compute driver runs. It currently supports running on Ubuntu 15.10[2], and will run on Ubuntu 16.04 in the Newton timeframe.

The PowerVM compute driver can be paired with the standard OVS Agent today, with support for the Linux Bridge network agent planned for the Newton release. This blueprint covers deploying the nova-powervm compute driver with the standard networking agents that OpenStack-Ansible supports.

The proposed changes include: * Add support for installing/configuring the PowerVM virt driver and dependencies * Tests to verify changes to the os_nova role required for PowerVM support

Note: PowerVM also supports a platform-specific Shared Ethernet ML2 Agent, which is not covered in this blueprint.

Alternatives

- Maintain independent PowerVM Ansible playbooks - This requires reinvention of base function and does not meet operator requirements.

Playbook/Role impact

See the Work Items for the playback/role impact. There will be a new nova-compute tag of nova-powervm that operators would use to support the PowerVM compute driver, and references to nova_virt_type will be updated to reflect a powervm option.

Upgrade impact

None. The nova-powervm driver is new for OpenStack-Ansible, and as such has no upgrade impact.

Security impact

None.

Performance impact

None.

End user impact

Deployers will be able to deploy compute nodes with the PowerVM virt driver.

Deployer impact

PowerVM specific configuration options will be added to the OpenStack-Ansible os_nova role. When support for PowerVM as a virt driver is enabled these config options will be used during deploy; however it is expected PowerVM support will be disabled by default, requiring that deployers explicitly enable PowerVM support and configure hosts for openstack-ansible to use.

Documentation of these new configuration items will be provided and a set of defaults will also be provided. The PowerVM driver has limited its configuration to be minimal, so the operators should only have a few required options to set when PowerVM is selected as the virt driver.

Developer impact

The existing development team will be asked for reviews and approvals of the change sets. The PowerVM driver team will do the necessary implementation and support of this function.

Dependencies

- Ironic nova_virt_type enhancements [4] - (Merged) This introduces support for additional nova_virt_types, which this work will expand on.
- Open vSwitch agent support [5] - Soft dependency on this to introduce OVS support, which the PowerVM computer driver supports today.
- Ubuntu 16.04/Multi-Host Support [6] - Needed to support Ubuntu 16.04, which will be the target OS for the PowerVM driver in the Newton timeframe.

10.8.3 Implementation

Assignee(s)

Primary assignee:
Wang Qing wu - wangqwsh on IRC and Launchpad

Other contributors:
Drew Thorstensen - thorst on IRC and Launchpad Adam Reznechek - adreznec on IRC and Launchpad

Work items

Multiple changes would be needed:

- Update the openstack_other.yml in the main openstack-ansible project to include the nova-powervm project.
- Define a new tag called nova-powervm. This will be used across the openstack-ansible projects.
- Add powervm to the nova_virt_types structure alongside the necessary variable requirements for driver configuration, matching the other compute types.
- As required, add nova.conf templating for powervm-specific configuration options that is conditionally included when nova_virt_type is powervm.
- Create a new nova_compute_powervm.yml in the openstack-ansible-os_nova project. This will contain the tasks needed to ensure the powervm driver is installed and configured on the system.
- Update the existing nova_compute.yml to include the nova_compute_powervm.yml and add the appropriate conditionals for that import.
- Create a new nova_compute_powervm_install.yml, which will be included by nova_compute_powervm.yml. It will ensure that the necessary configuration and dependencies for running the PowerVM driver are in place.
- Update documentation and comments indicating the new PowerVM nova_virt_type and how to configure OpenStack-Ansible for the PowerVM driver.
- Automated unit test (see Testing)

10.8.4 Testing

The PowerVM Driver CI System is currently using devstack for its set up. This cloud will be updated to make use of OpenStack-Ansible to deploy the operator cloud that runs the CI infrastructure.

A new test-install-nova-powervm.yml will be created for validating the new powervm playbooks within the openstack-ansible-os_nova project.

10.8.5 Documentation impact

Documentation covering how to enable and configure PowerVM support will be added to the user guide.

10.8.6 References

1. nova-powervm driver: https://github.com/openstack/nova-powervm
2. PowerVM NovaLink: https://www.ibm.com/developerworks/community/wikis/home?lang=en#!/wiki/Power%20Systems/page/Introducing%20PowerVM%20NovaLink
3. PowerVM Mitaka Update: https://www.ibm.com/developerworks/community/wikis/home?lang=en#!/wiki/Power%20Systems/page/OpenStack%20and%20PowerVM%20-%20Mitaka%20Update
4. Nova config for os_ironic: https://review.openstack.org/#/c/293315
5. Neutron Open vSwitch Agent support: https://review.openstack.org/#/c/298765/

6. Support Ubuntu 16.04: https://blueprints.launchpad.net/openstack-ansible/+spec/multi-platform-host

10.9 Additional Role for Gnocchi Deployment

date
2016-01-20 11:20

tags
gnocchi, openstack-ansible

The purpose of this spec is to add support for the OpenStack Gnocchi program to OpenStack-Ansible. This would allow the deployment of Gnocchi along with the core OpenStack components using OpenStack-Ansible.

Blueprint - Gnocchi deployment on OpenStack-Ansible:

https://blueprints.launchpad.net/openstack-ansible/+spec/role-gnocchi

10.9.1 Problem description

Presently, while deploying OpenStack using OpenStack-Ansible only the core OpenStack components get deployed. The deployment of other components (eg: Gnocchi) via Ansible playbooks is not supported yet and to use this components services, they need to be deployed manually.

Gnocchi[1] is a multi-tenant timeseries, metrics, and resources database. It is designed to store metrics at a very large scale and to allow the retrieval of those metrics quickly and efficiently, each through an HTTP REST interface. Additionally, Gnocchi is designed to stand as a replacement storage engine for metrics processed through Ceilometer, relying on a more performant storage format.

10.9.2 Proposed change

This spec proposes to allow deployment and management of this service with a versatile configuration capable of scaling in a way that conforms to both the best practices from the Gnocchi and Telemetry communities as well as with those of the OpenStack-Ansible community.

This involves adding support for the Gnocchi services, and the Gnocchi client[2] into the appropriate hosts and containers. It also involves the optional configuration of Ceilometer to use Gnocchi in lieu of the currently supported MongoDB storage solution.

The proposed changes include:

- Creation of an openstack-ansible-os_gnocchi repository and Ansible role to support the deployment and management of Gnocchi services.
- Tests to verify the new Ansible role and the integration with OpenStack services.
- Documentation to support the roles operation and common deploy configurations.

Alternatives

None

Playbook/Role impact

Test playbooks will be placed in the openstack-ansible-os_gnocchi repository for functional testing purposes. Some changes are anticipated within the openstack-ansible-os_ceilometer role to configure deployment options needed for an optional integration where Ceilometer uses Gnocchi as its storage engine. Further a playbook, necessary group_vars and env and conf profiles would be provided for the openstack-ansible repository to complete the integration.

Upgrade impact

None. While an operator who had previously deployed Ceilometer might be interested in deploying Gnocchi, there is no migration model for exporting data from Ceilometer internal storage to Gnocchi storage. The Gnocchi role and any supporting changes to the Ceilometer role would make the transition as simple as running the associated playbooks.

Security impact

None.

Performance impact

The underlying storage used for Gnocchi would experience higher traffic, which might require a deployer to account for that additional traffic through additional tuning.

No other performance impacts are expected.

End user impact

OpenStack users would be able to make use of Gnocchi as a multi-tenant high volume timeseries data store when deployers use the role and associated playbook to deploy Gnocchi.

Deployer impact

This work provides an optional role for use in the OpenStack-Ansible tooling for use in their environments.

Developer impact

Some conditionals may be introduced into the Ceilometer role to facilitate the clean deployment of Gnocchi as a storage engine for Ceilometer. All other changes would be self-contained or limited to the introduction of new variables and a playbook which should present little to no additional cognitive load for developers.

Dependencies

Only a MySQL-compatible RDBMS is proposed for the Gnocchi index. Only filesystem, Ceph, and Swift storage engines are proposed. All of these are currently available within OpenStack-Ansible.

No support is proposed at this time for use of Graffana as a dashboard or the use of the statsd service endpoints for Gnocchi. In this way we avoid introducing any new dependencies.

10.9.3 Implementation

Assignee(s)

Primary assignee:
Steve Lewis (IRC: stevelle)

Other contributors:
None

Work items

1. Ask for the new repository, openstack-ansible-os_gnocchi, to be created
2. Create the role for Gnocchi support:

- Add support for running Gnocchi services (api and metricd) with basic convergence testing.
- Add an Ansible module to leverage gnocchiclient, which is the command line interface tool for using and managing Gnocchi.
- Introduce a playbook for deploying Gnocchi with OpenStack-Ansible.
- Add support for Gnocchi as an optional storage engine for Ceilometer.
- Add a full scenario test (described below) to ensure successful integration of Gnocchi.
- Add documentation to the role, and possibly general documentation to the install guide for deploying Gnocchi with each of the various supported storage engines.

10.9.4 Testing

In an environment where the role is integration tested through the openstack-ansible repository in one monolithic stack, the additional effort of deploying this additional project could add as much as a few minutes to gate testing. That is not desirable.

To preclude the need for that additional step in the main gate, a longer scenario test is proposed for inclusion in the openstack-ansible-os_gnocchi role, to integrate with Keystone, Nova, Cinder, Glance, Neutron, Ceilometer, and Gnocchi with metrics collection enabled and with Nova being exercised to ensure metering data propagates through the OpenStack environment.

This can then be verified through the use of the Ansible module for gnocchiclient by querying for the expected measures.

10.9.5 Documentation impact

Role-specific documentation describing the configuration of Gnocchi will be required.

10.9.6 References

- [1] Gnocchi: http://gnocchi.xyz/
- [2] Gnocchi client: http://git.openstack.org/cgit/openstack/python-gnocchiclient/

10.10 Additional Role for Tacker Service Deployment

date
: 2016-10-19 12:30

tags
: tacker, openstack-ansible

The purpose of this spec is to add support for the OpenStack Tacker service to OpenStack-Ansible. This would allow the deployment of Tacker along with the core OpenStack components using OpenStack-Ansible.

Blueprint - Tacker deployment on OpenStack-Ansible:

https://blueprints.launchpad.net/openstack-ansible/+spec/role-tacker

10.10.1 Problem description

Presently, while deploying OpenStack using OpenStack-Ansible only the core OpenStack components get deployed. The deployment of other components (eg: Tacker) on playbooks is not supported yet and to use other components services, they need to be deployed manually.

10.10.2 Proposed change

This change involves adding support for the Tacker server, Tacker client, and Tacker Horizon dashboard interface.

The proposed changes include:

- Creation of an openstack-ansible-tacker repository and Ansible role to support the deployment of Tacker.
- Tests to verify the new Ansible role.
- Deployment of Tacker client
- Deployment of Tacker Horizon

Alternatives

None

Playbook/Role impact

Test playbooks will be placed in the openstack-ansible-tacker repository for functional testing purposes, with no initially proposed changes to OpenStack-Ansible playbooks.

In the future, once the Tacker role has reached a muture state, a future spec will address the integration of the Tacker role with the main OpenStack-Ansible repository.

Upgrade impact

None

Security impact

None.

Performance impact

None.

End user impact

Deployers will be able to deploy Tacker service through OpenStack-Ansible framework for VNF management and orchestration purposes.

Deployer impact

When support for the new Tacker role is added to the parent repository, new Tacker specific configuration options will be made available. This will provide an optional role for use in the OpenStack-Ansible toolbox for the deployers.

Developer impact

As this change is self-contained initially, no impact on other developers is expected.

Dependencies

None

10.10.3 Implementation

Assignee(s)

Primary assignee:
Jeff Rametta (IRC: jcrst)

Other contributors:
None

Work items

1. Ask for the new repository, openstack-ansible-tacker, to be created
2. Create the role for Tacker support
   - Add support for running tacker-sever
   - Add support for python tacker client
   - Add support for Tacker Horizon dashboard
   - Add documentation and install guide for the role

10.10.4 Testing

The usual gate checks can be used for these changes. Also, each individual commit can be functionally tested individually.

10.10.5 Documentation impact

Adding support to the user guide on how to enable Tacker support will be required.

10.10.6 References

- Tacker server: https://git.openstack.org/cgit/openstack/tacker/
- Tacker client: https://git.openstack.org/cgit/openstack/python-tackerclient
- Tacker Horizon: https://git.openstack.org/cgit/openstack/tacker-horizon

10.11 Apply RHEL 7 STIG hardening standard

date
2016-08-11 00:00

tags
security

The Security Technical Implementation Guide (STIG) for Red Hat Enterprise Linux (RHEL) 7 is in the final stages of release. The security hardening role needs to be updated to apply these new requirements to Ubuntu 16.04, CentOS 7 and RHEL 7.

- https://blueprints.launchpad.net/openstack-ansible/+spec/security-rhel7-stig

10.11.1 Problem description

Today, the openstack-ansible-security role uses the RHEL 6 STIG as the basis for all of the security configurations applied to Ubuntu 14.04, Ubuntu 16.04, CentOS 7, and RHEL 7. However, the new RHEL 7 STIG is in the final stages of its release and the new security configurations provide a stronger security posture for all systemd-based distributions, including:

- Ubuntu 16.04
- CentOS 7
- RHEL 7

There are some challenges with a wholesale change to the RHEL 7 STIG:

- It doesnt apply well to Ubuntu 14.04
- It uses a new numbering scheme which doesnt match with the RHEL 6 STIG
- Many tasks have no overlap with the RHEL 6 STIG

10.11.2 Proposed change

The current role structure is flat and the differences between the distributions are handled within each task YAML file. The proposed new layout would look something like this:

```
/main.yml
/rhel6stig/main.yml
/rhel6stig/auth.yml
/rhel6stig/boot.yml
/rhel6stig/...
/rhel7stig/main.yml
/rhel7stig/auth.yml
/rhel7stig/boot.yml
/rhel7stig/...
```

The root `main.yml` would have a `when:` that would include the correct `main.yml` from the STIG version subdirectories. This comes with some nice benefits:

1. This would ensure that the functionality for Ubuntu 14.04 is unchanged.
2. When support for Ubuntu 14.04 is no longer needed, it could easily be removed later by simply removing the `rheli6stig` directory and the corresponding `include:` from the root `main.yml` file.
3. Some of the existing clutter in the role could be removed since Ubuntu 16.04, CentOS 7 and RHEL 7 are closely aligned (because they all use systemd).

Alternatives

Switch the entire role to the RHEL 7 and drop Ubuntu 14.04 support.
This could be upsetting for existing users of the role on 14.04.

Interleave the RHEL6/RHEL7 STIG configurations in the existing role
This could lead to lots of clutter and could add difficulty when Ubuntu 14.04 support needs to be removed.

Create a new role
Deployers could be confused by a new role and it would require changes to OpenStack-Ansibles integrated build.

Playbook/Role impact

See the **Proposed change** section above for details.

Upgrade impact

If a deployer is running the Newton release of the role on Ubuntu 16.04, CentOS 7, or RHEL 7, they will notice lots of additional security configurations being applied by the role per the requirements of the RHEL 7 STIG. Backing out security configurations from the previous versions of the role shouldnt be necessary.

Security impact

This change will improve the roles capability to secure new systemd-based distributions, such as Ubuntu 16.04, CentOS 7, and RHEL 7.

Performance impact

As with the previous versions of the role, the updates to the role from the RHEL 7 STIG should not cause performance impacts or downtime on the system.

End user impact

End users should not notice a difference when these changes are made.

Deployer impact

Deployers will apply the role using the same commands as they do now. However, they will see some new changes:

- New configurations being applied that werent being applied previously
- New variables for controlling the security configurations in the RHEL 7 STIG

Developer impact

Developers must ensure that RHEL 7 STIG content is kept separate from RHEL 6 content. This will be documented within the tasks themselves as well as in the formal role documentation.

Dependencies

This change has no dependencies.

10.11.3 Implementation

Assignee(s)

Primary assignee:
Major Hayden (LP: rackerhacker, IRC: mhayden)

Work items

1. Create a directory for the RHEL 7 STIG content and begin adding tasks there to apply security configurations.
2. Update documentation to reflect the new configurations and any new variables which exist to configure the roles actions.
3. When the RHEL 7 STIG content is working well on Ubuntu 16.04, CentOS 7, and RHEL 7, the root `main.yml` should include the tasks from the RHEL 7 STIG directory.
4. At a later date, Ubuntu 14.04 support could be removed by deleting the RHEL 6 directory, removing unneeded variables, and removing unneeded documentation.

10.11.4 Testing

The OpenStack CI environment would test the security role in the same way that it does now. Testing could be adjusted during the first phase of RHEL 7 STIG development so that both pathways (RHEL 6 STIG and RHEL 7 STIG) are tested on Ubuntu 16.04 and CentOS 7.

RHEL 7 testing will need to be manual since OpenStack CI has no RHEL image.

10.11.5 Documentation impact

New documentation will be needed for the RHEL 7 STIG security configurations as well as any new variables that are introduced. This will need to be done carefully (perhaps in a draft directory) until the RHEL 7 STIG content is ready to be applied to Ubuntu 16.04 and CentOS 7.

10.11.6 References

- DISA STIGs: http://iase.disa.mil/stigs/os/unix-linux/Pages/index.aspx
- openstack-dev mail: http://lists.openstack.org/pipermail/openstack-dev/2016-August/100883.html

10.12 standalone-swift

date
2015-07-07 22:00

tags
swift, aio, tests

This spec exists to allow for testing a diferent deployment methodogy, namely swift deployments. The problem is the openstack_user_config.yml.aio file defines hosts that are not needed for an AIO deployment.

- https://blueprints.launchpad.net/openstack-ansible/+spec/standalone-swift

10.12.1 Problem description

Deploying aio for testing deploys all Openstack services only swift is desired. We are not testing this deployment type.

10.12.2 Proposed change

- add openstack_user_config.yml.aio.swift for swift only deployments.
- add/modify the deployment scripts to add a switch for swift only deployments.
- modify tests to allow for swift only deployments.

Alternatives

N/A

Playbook impact

Minimal to no impact to the actual playbooks.

Upgrade impact

N/A

Security impact

N/A

Performance impact

N/A

End user impact

Allows the end user to use the openstack_user_config.yml.aio.swift file as a template to base their own swift deployments.

Deployer impact

The playbooks would remain unchanged, only deployers using the scripts may need to change, this does not alter default behavior.

Developer impact

This would allow testing of standalone swift deployments.

Dependencies

N/A

10.12.3 Implementation

Assignee(s)

Primary assignee:
prometheanfire

Work items

- create aio file
- add/alter scripts to allow for standalone swift testing (tempest changes)
- add test to project_config
- enable test in openstack-ansible

10.12.4 Testing

This will add a test/vote to openstack-ansible

10.12.5 Documentation impact

Possibly pointing out the openstack_user_config.yml.aio.swift file as a template for larger deployments and documenting the new environment variables.

10.12.6 References

N/A

10.13 Add support for multiple RabbitMQ clusters

date
2016-07-11 21:00

tags
rabbitmq, messaging, notifications, scalability

Larger deployments may wish to provision multiple RabbitMQ clusters such that each cluster is deployed on its own set of hosts.

Such functionality would allow a deployer to configure one or more additional component and container skeletons to add inventory groups to be used for the deployment of additional clusters.

10.13.1 Problem description

The current playbook and roles assume a single inventory group: `rabbitmq_all` that is deployed on the `shared-infra_hosts` infrastructure. The inventory group name is hardcoded throughout and the playbook makes the assumption that only one cluster will ever be needed.

10.13.2 Proposed change

- Modify the rabbitmq_server role to be more configurable with respect to the inventory group(s) that it operates upon
- Modify the rabbitmq-install play to be more configurable with respect to the inventory group it operates upon

Alternatives

Im not aware of alternative ways for the project to address this need.

Playbook/Role impact

Initial impacts will be to the rabbitmq_server role and the rabbitmq-install play. However, I expect that additional impacts may exist within other roles such that they would need to change to be more configurable with respect to the inventory group they expect to use for rabbit hosts, or the variables they use to identify which rabbit hosts they should connect to.

Upgrade impact

Unclear on how upgrades would be impacted. To my knowledge, custom inventory extensions are not currently handled in the upgrade automation.

Security impact

No unique security impacts. The existing RabbitMQ security posture will be maintained, though additional secrets may be required.

Performance impact

None expected/anticipated.

End user impact

End users will have increased flexibility in defining their deployment architecture.

Deployer impact

The goal is for the deployer impact to be negligible due to the opt-in nature of the changes discussed.

Developer impact

This change will add some additional complexity for developers, but it should be minimal.

Dependencies

None

10.13.3 Implementation

Assignee(s)

Primary assignee:
travis-truman (automagically)

Work items

- rabbitmq_server role modifications for inventory group configurability
- rabbitmq-install play modifications for inventory group configurability
- Documentation explaining how to create additional RabbitMQ cluster groups
- Other role modifications to support cluster connectivity configurability

10.13.4 Testing

This should be able to be tested within the rabbitmq_server role functional tests given some changes to the test inventory.

10.13.5 Documentation impact

An appendix should be added that explains to deployers how to configure their environment for RabbitMQ multiple cluster support.

10.13.6 References

None

10.14 Support Xen Virt Driver

date
2016-06-03 11:17

tags
ansible, xen

The purpose of this spec is to add support for the Xen Hypervisor to OpenStack-Ansible. This will allow the use of Xen as an option on OpenStack compute nodes.

https://blueprints.launchpad.net/openstack-ansible/+spec/xen-virt-driver

10.14.1 Problem description

Xen is a tested and supported hypervisor in OpenStack. It is used in some of the largest public clouds today and would make a good addition to OpenStack-Ansible. Support for Xen exists in the OpenStack Libvirt Driver today so implementation should not be difficult.

10.14.2 Proposed change

The primary change is to add support in OpenStack-Ansible for Xen on CentOS 7, Ubuntu 16.04, and Ubuntu 14.04 (by using UCA repos). The necessary changes for Xen to work with OpenStack are in Xen 4.5.1 and Libvirt 1.2.15. This blueprint covers deploying the nova-xen compute driver with the standard networking agents that OpenStack-Ansible supports.

The proposed changes include:

- Add support for installing/configuring the Xen virt driver and dependencies
- Documentation for how to configure a compute to run the Xen virt driver
- Tests to verify changes to the os_nova role required for Xen support

Alternatives

- Maintain independent Xen Ansible playbooks - This requires reinvention of base function and does not meet operator requirements.

Playbook/Role impact

See the Work Items for the playback/role impact. References to nova_virt_type will be updated to reflect a xen option.

Upgrade impact

None. The xen driver is new for OpenStack-Ansible, and as such has no upgrade impact.

Security impact

None.

Performance impact

None.

End user impact

End users will be able to deploy compute nodes using the Xen virt driver.

Deployer impact

Xen specific configuration options will be added to the openstack-ansible-os_nova role.

When support for Xen as a virt driver is added these config options will be available for use; however it is expected that Xen support will be disabled by default, requiring that deployers explicitly enable Xen support and configure hosts for OpenStack-Ansible to use.

Documentation of these new configuration items will be provided and a set of defaults will also be provided. The Xen virt driver has limited its configuration to be minimal, so the operators should only have a few required options to set when Xen is selected as the virt driver.

Developer impact

The existing development team will be asked for reviews and approvals of the change sets. The Xen driver team will do the necessary implementation and support of this function.

Dependencies

- Multi-platform Host OS Enablement - Needed to support Ubuntu 16.04 and CentOS 7

10.14.3 Implementation

Assignee(s)

Primary assignee:
Antony Messerli - antonym on IRC and Launchpad

Other contributors:

Work items

Multiple changes would be needed:

- Add xen to the nova_virt_types structure alongside the necessary variable requirements for driver configuration, matching the other compute types.
- As required, add nova.conf templating for xen-specific configuration options that is conditionally included when nova_virt_type is xen.
- Create a new nova_compute_xen.yml in the openstack-ansible-os_nova project. This will contain the tasks needed to ensure the xen driver is installed and configured on the system.
- Update the existing nova_compute.yml to include the nova_compute_xen.yml and add the appropriate conditionals for that import.
- Create a new nova_compute_xen_install.yml, which will be included by nova_compute_xen.yml. It will ensure that the necessary configuration and dependencies for running the Xen driver are in place.
- Update documentation and comments indicating the new Xen nova_virt_type and how to configure OpenStack-Ansible for the Xen driver.
- Automated unit test (see Testing)

10.14.4 Testing

A new test-install-nova-xen.yml will be created for validating the new xen playbooks within the openstack-ansible-os_nova project.

10.14.5 Documentation impact

Documentation covering how to enable and configure Xen support will be added to the user guide.

10.14.6 References

Xen and OpenStack required versions: http://wiki.xenproject.org/wiki/OpenStack_CI_Loop_for_Xen-Libvirt

Multi-platform Host OS Enablement: https://blueprints.launchpad.net/openstack-ansible/+spec/multi-platform-host

CHAPTER

ELEVEN

MITAKA SPECIFICATIONS

11.1 Build Facts Archive

date
2015-04-23

tags
archive, deployment, information

Create a script to archive all valuable information about a deployment. This information includes but is not limited to the following: kernel version of all physical host, version of OSAD that is currently installed, all installed packages and their versions, all running containers and their installed packages, latest tempest test run, all relevant OSAD configuration files (openstack_user_config, etc), host networking configuration, host disk configuration.

- https://blueprints.launchpad.net/openstack-ansible/+spec/build-facts-archive

11.1.1 Problem description

Currently there is no simple way to get information about a deployment. The current process requires a log into the deployment host and then knowledge of ansible and the openstack_inventory.json file and its groups to correctly structure a ansible query to gather information.

It is also challenging to create a tool outside of OSAD to do some automation around aggregation of deployment information as you need to parse the inventory file or know exactly which host or container you need to access to get information.

11.1.2 Proposed change

A simple script that the user can run to gather all predetermined important information to give a solid top down view of a deployed cluster.

Alternatives

This script could live in the rpc-extras repository instead of OSAD. This would not be ideal as it would only help the users who are using rpc-extras. If it resides in OSAD then all users get a simple way of getting a quick top down view of what their current cluster has.

Playbook impact

There will need to most likely be a playbook added to accomplish the task of gathering valuable information from each host / container based on their role / group. This playbook will not be deployment impacting.

Upgrade impact

None

Security impact

This could potentially touch all containers to gather secure information such as: configuration files which may contain passwords, information about keystone users (names, roles, etc). This is a minimal risk as the user would have to export the output off the host. If someone is running this script they already have access to this information as they are logged onto the deployment host.

Performance impact

None

End user impact

None

Deployer impact

This change will give the deployer an easy way to gather current information about their cloud. This could help troubleshoot config problems as well as allow them a quick insight into their latest test results. This will even allow them to see package discrepencies and help them prepare for an upgrade.

Developer impact

None

Dependencies

None

11.1.3 Implementation

Assignee(s)

Open to all

Primary assignee:
None

Other contributors:
None

Work items

Create a script to:

- create the archive directory
- gather all relevant deployment information
- tarball archive directory
- move tarball to well known location
- remove archive directory

It would be up to the end user / deployer what they do with the tarball, but it should be placed in a resonable spot on the deployment host that would be easy to find / access for the deployer.

11.1.4 Testing

This should add a task to gating/commit/nightly to run this script and return the captured archive tarball as a jenkins artifact. Tempest results could also be sent to jenkins so that the results.xml can be displayed. This should help developers / qe see testing trends and allow the users of jenkins to more accurately find bugs in a more timely manner.

11.1.5 Documentation impact

A simple reference to this script in the user guide would be all that is needed if it is determined that it warrents it.

11.1.6 References

If you look at the current scripts located in the scripts directory

- https://github.com/openstack/openstack-ansible/blob/master/scripts/scripts-library.sh#L226-L279

a lot of this information is already gathered about the host that the script is run on. This proposal should use the information that is gathered as a blueprint to some of the information that should be gathered about all hosts.

11.2 Convert AIO bootstrap to Ansible

date
2015-10-16 00:00

tags
aio, bootstrap, ansible

The process for an AIO installation of openstack-ansible involves a bash script to do the initial bootstrapping of the AIO host. This script works well, but it becomes difficult to update over time and a conversion to Ansible would make future updates, such as multi-platform-host blueprint, a little easier.

Blueprint - Convert AIO bootstrap to Ansible:

- https://blueprints.launchpad.net/openstack-ansible/+spec/convert-aio-bootstrap-to-ansible

11.2.1 Problem description

The `bootstrap-aio.sh` script works well, but it can be difficult to read in a few places. Deployers who are familiar with Ansible, but not bash, may have challenges with updating the script as well.

11.2.2 Proposed change

At this time, the AIO installation has four steps:

- Configuration *(optional)*
- Bootstrap the AIO build
- Bootstrap Ansible
- Run the openstack-ansible playbooks

This spec proposes the following steps to replace the existing ones:

- Configuration *(optional)*
- Bootstrap Ansible
- Run AIO playbook *(if an AIO deployment is desired)*
- Run the openstack-ansible playbooks

The current AIO boostrap script is **heavily** used by various deployers as well as other downstream projects, so changes must be made carefully. The proposed work for this spec would proceed as follows:

- Build out the Ansible role for bootstrapping an AIO build
- Update documentation to allow for early testing
- Change the `bootstrap-aio.sh` script to call the new AIO bootstrap playbook
- Update the documentation to reflect the new bootstrap script changes
- Remove the `bootstrap-aio.sh` script at a later date (if needed)

Alternatives

The current `bootstrap-aio.sh` script could remain as it is now, or it could be simplified to make it easier to read and update.

Playbook/Role impact

The openstack-ansible playbooks themselves shouldnt change as a result of this update. The AIO bootstrap is a prerequisite step in the deployment right now and that wont change after the AIO Ansible playbook is available for use.

Upgrade impact

This change would only affect greenfield deployments of AIO builds. If a deployer has an existing AIO build deployed, they would not need to run the AIO bootstrap playbook again, even with upgrades.

Security impact

There are no known security impacts of this change.

Performance impact

There are no known performance impacts of this change. The Ansible AIO playbook may be slightly slower than the bash script, but the difference should be negligible.

End user impact

An end user would not notice this change since it would only affect deployers.

Deployer impact

If deployers are doing greenfield AIO deployments, they will need to follow new steps and ensure they bootstrap Ansible prior to running the new AIO Ansible playbook. Documentation for AIO builds will require updates.

If deployers are doing deployments to multiple servers (non-AIO), their steps for deploying openstack-ansible will not change.

Developer impact

Developers will need to make any future AIO bootstrap changes within the Ansible playbook instead of the bash script.

Dependencies

This spec doesnt depend on any other blueprint or spec at this time.

11.2.3 Implementation

Assignee(s)

Primary assignee:

- Major Hayden (Launchpad: rackerhacker, IRC: mhayden)

Work items

The last bulleted list in *Proposed Changes* above details out the work items.

11.2.4 Testing

These changes will impact gating since the gating jobs run an AIO build. However, if the bootstrap-aio.sh script is changed to call the AIO bootstrap Ansible playbook, the gating job itself will not need to be changed.

No additional resources should be required during gating to run the Ansible AIO playbook.

11.2.5 Documentation impact

The documentation for AIO deployments would need to be updated with the new steps for bootstrapping an AIO build. The changes in the steps are in the *Proposed Changes* section at the top of this spec.

Also, deployers would need to note which environment variables and/or Ansible variables to set to control various parts of the deployment, such as whether or not to deploy certain OpenStack services in their environment.

11.2.6 References

No references at this time.

11.3 Independent Role Repositories

date
2015-08-17 14:00

tags
roles

In order to improve the ability to independently consume the roles produced by openstack-ansible in different reference use-case deployments and allow independent development of each role by different projects, this specification proposes that:

1. New roles be registered in separate repositories named `openstack/openstack-ansible-<role>`.
2. Existing roles can, through an independent blueprint/spec process, be split into their own repositories.

- https://blueprints.launchpad.net/openstack-ansible/+spec/independent-role-repositories

This provides the following benefits:

- Other projects (eg: DevStack, Kolla, Compass, RPC, etc) will be able to consume the roles using their own playbooks. This increases the opportunity for other projects to collaborate with openstack-ansible.
- The roles will be more easily consumed in different reference architectures. Currently the playbooks and roles are specifically geared for deployment in LXC containers. Making the roles independent entities will allow them to be consumed for completely different architectures. eg: a deployment without containers, a deployment with VMs instead of containers, a deployment with a different container technology.
- Each role can be developed at its own pace and versioned independently. This will make openstack-ansible more like the OpenStack Big Tent.
- Each role can be independently gate checked with checks that are specific and relevant to the role. This will provide quicker developer feedback and allow a quicker turnaround for development.
- The roles can be registered in Ansible Galaxy. This will provide greater awareness of the roles and may attract more contributors. This is especially useful for the infrastructure roles which may have a broader application than just for use in an OpenStack deployment. (eg: haproxy, MariaDB, etc)
- Role separation will simplify the mission of openstack-ansibles playbooks and scripts to be for the purpose of providing examples of how to consume the roles and to implement gate testing for integrated deployment verification of the use-cases that matter to the community. This prevents the situation where the playbooks have to cater for every possible combination use-case that may be thrown at them.

11.3.1 Problem description

A detailed description of the problem:

- Currently the openstack-ansible roles are tightly coupled with the playbooks that consume them. While the roles can technically be consumed using different playbooks, this is not immediately obvious to downstream consumers.
- When consumers to try to consume the roles with different playbooks for a different architecture, they are forced to implement many workarounds for the tight coupling that we have. Even when it is possible to do, it is hard for a deployer to see how to do it due to the deluge of variables that need to be set and code that needs to be read.
- There has been some interest in creating roles for other services (eg: rally, congress, etc). Implementing the strategy of a repo per role allows these fledgling roles to get into the open and get collaborated on far more quickly. Once theyre at the point where theyre ready to be integrated into an integrated use-case with integration gate tests, then they can tag a version and implement the openstack-ansible playbooks and scripts to test the appropriate use-case.
- The infrastructure roles implemented as part of the project are not getting much attention as they are more like a second-class citizen within the role structure. This results in the configurations deployed often not lining up to best-practices.
- Any small changes to roles require the execution of a full integration gate test which is slow and prone to error. It is difficult to isolate the error in the current monolithic stack.

11.3.2 Proposed change

The following documentation should be developed:

1. The primary use-cases/implementations tested by the project.
2. How to apply to add a new use-cases for integration testing.
3. How to register a new role within openstack-ansibles umbrella.
4. First steps for building a new role.
5. Update the README for each role to describe how to use it independently, whether using a static inventory or the dynamic inventory. It should also cover what options are available for its use, whether it relies on any other roles, how upgrades are handled, any known issues, etc.

The following process should be followed for registering new roles:

1. A blueprint must be registered and a spec for the implementation provided for review, with specific attention paid to any changes that would need to be made in the current openstack-ansible playbooks and roles to integrate the new role.
2. Once the spec is approved, the review to register the new repository should be registered upstream by the openstack-ansible PTL or a nominated openstack-ansible-core team member.
3. Once the new repository has been created, work can commence on the new role.
4. Before the first tag is set for the role, comprehensive testing for the role must already be in place.

The following process should be followed for breaking out existing roles:

1. For each role targeted for breaking out, a separate blueprint must be registered and a spec provided for the high level changes that would be required in the current openstack-ansible playbooks and roles to accommodate this change.
2. Once the spec is approved, the review to register the new repository should be registered upstream by the openstack-ansible PTL or a nominated openstack-ansible-core team member.
3. Once the new repository has been created, work can commence on extracting the role as planned in the spec for the roles migration.
4. Once independent gate checks on the role repository confirm that it is in working order and the work is done to prepare the role for usage in the integrated use-cases, tag the initial version of the role and implement the openstack-ansible playbook, script and role-requirements changes to consume the new role. The changes in openstack-ansible will need to pass the integrated gate checks before they can merge.

Alternatives

Leave everything as it is and continue to merge any new roles proposed into the same monolithic repository.

Playbook/Role impact

The impact to playbooks should be fairly incremental, but will need to be determined on a role by role basis. This must be described in the spec on a per role basis.

It is clear that the libraries, filters and plugins will need to be broken out into its own repository and each role that consumes them will need to use a submodule reference in the role. This ensures that all the roles use a common set of libraries, filters and plugins.

Upgrade impact

There are two aspects of upgrade impact to be considered:

1. The ability for the role itself to handle upgrades from previous interations of itself.
2. The ability for the integrated build use-cases that consume the role to be upgradable.

Each role should be developed with upgradability in mind and conform to the upgrade framework which is being developed for the Kilo -> Liberty upgrade process.

Security impact

n/a

Performance impact

n/a

End user impact

End users will not know of any differences.

Deployer impact

The structure and placement of roles on the deployment host will be different. The changes will need to be documented for support purposes. However, the configuration and execution of the existing playbooks for downstream consumers should be targeted to be exactly the same to minimise disruption.

Any role-specific impacts will need to be defined on a per-role basis.

Developer impact

The biggest negative developer impact will be the difficulty working between the two different structures - master being split, with liberty and kilo being consolidated. It may be worth considering ways to make them all work the same way once the conversions are done for master.

The positive impact has been outlined in the introduction.

Dependencies

This should only be implemented after liberty has been released.

11.3.3 Implementation

Assignee(s)

Primary assignee:
https://launchpad.net/~jesse-pretorius `odyssey4me` https://launchpad.net/~kevin-carter `cloudnull`

Other contributors:
TBD

Work items

See proposed change section.

11.3.4 Testing

The testing impact will need to be described on a per-role basis.

As a standard, the following tests are expected to be implemented as a standard for each role:

- bashate tests for all shell scripts
- pep8 tests for all python files
- ansible syntax checks for all ansible task files
- ansible-lint tests for all ansible task files
- functional tests for the service

Integration tests are expected to be implemented in the openstack-ansible repository and executed whenever the role versions are incremented. This ensures that a role tag increment is only accepted for an integrated release once it passes a full set of integration tests.

11.3.5 Documentation impact

While the placement of the role files on the deployment host will be different, the configuration and execution of the deployment should remain the same, resulting in minimal documentation impact.

See the proposed change section for developer reference documentation to be developed.

11.3.6 References

n/a

11.4 Installation Guide

date
2015-11-02 22:00

tags
install, config, architecture

https://blueprints.launchpad.net/openstack-ansible/+spec/install-guide

Improve the installation guide to appeal to more potential deployers.

11.4.1 Problem description

The current installation guide mainly supports only one rather complex deployment architecture that limits the apparent flexibility and appeal of the project to potential deployers.

11.4.2 Proposed change

Improve the installation guide to offer several useful deployment architectures ranging from simple to complex.

Alternatives

Continue using the existing content that contains significant technical debt from decisions made prior to entry into the Stackforge and later OpenStack namespaces.

Playbook/Role impact

None.

Upgrade impact

None, although a separate specification should address development of upgrade documentation referencing the deployment architectures in the installation guide as necessary.

Security impact

None, although the deployment architectures should implement security measures as necessary.

Performance impact

None, although more complex deployment architectures could perform poorly on hardware that disregards minimum requirements.

End user impact

None.

Deployer impact

A variety of different deployment architectures ranging from simple to complex highlight the flexibility of this project and increase appeal to potential deployers.

Developer impact

Developers should understand these deployment architectures and adjust them as necessary to account for new services, changes to existing services, changes to infrastructure requirements, etc.

Dependencies

None.

11.4.3 Implementation

Assignee(s)

Primary assignee:
None

Other contributors:
None

Work items

- Develop several deployment architectures that range from simple to complex and attempt to minimize opinions regarding OpenStack service configuration and operation. For example:
  - A simple architecture may include a minimum of two infrastructure nodes and one compute node using three networks with minimal physical network redundancy and deploy only core OpenStack services.
  - A complex architecture may include a minimum of three infrastructure nodes, one compute node, and three storage nodes using four networks with reasonable network redundancy and deploy all OpenStack services.
- Potentially restructure the installation guide to implement these deployment architectures in the most useful fashion.

11.4.4 Testing

- Verify operation of each deployment architecture prior to each major release.

11.4.5 Documentation impact

- Renovating the installation guide.

11.4.6 References

None.

11.5 IRR - APT package Pinning

date
2015-11-01

tags
independent-role-repositories, apt_package_pinning

Split out the apt package pinning role into its own repository.

11.5.1 Problem description

Roles are all contained within a single monolithic repository making it impossible/difficult to consume the OSA roles outside of deploying the entire stack.

11.5.2 Proposed change

To ensure that the OSA project is consumable by other stacks using different architectures, deployment methods, and capabilities the role apt_package_pinning need to be moved from the monolithic stack and into the its own role repository.

Alternatives

Leave everything the way it is. However doing that will hurt general OSA adoption.

Playbook/Role impact

- No impact to the playbooks.
- The role will be removed from the main stack. The plugins, filters, and libraries may need to be locally updated.

Upgrade impact

While the change will impact the placement of the role it will not impact upgrade-ability of the stack. The general workflow will need to be updated to ensure that users are updating roles on upgrade using the Ansible galaxy interface however generally speaking this is already being done for the deployer when running the `bootstrap-ansible.sh` script.

Security impact

n/a

Performance impact

Moving the role to an external repository will cause an impact in time to role resolution however that impact should be minimal.

End user impact

n/a

Deployer impact

Deployers will need to be aware of the new role locations and how to update existing roles however this should be minimal considering the tooling for updating existing roles already exists

Developer impact

Developers will need focus work within the roles which will exist within separate repositories.

Dependencies

n/a

11.5.3 Implementation

Assignee(s)

Primary assignee:
https://launchpad.net/~kevin-carter (IRC: cloudnull)

Work items

- With the role moved, tests will be created within the role via OpenStack CI to ensure that the role is performing the actions that its supposed to.
- Updated documentation via the README.rst will be created to show how the role can be used standalone.
- Example local inventory will be created to show how the role can be used. The local only inventory will also be used for testing the role.

11.5.4 Testing

- The test cases will deploy the role into a regular DSVM image
- The role will execute itself locally
- Once the role has completed an Ansible test play will run through several assert tasks to ensure the role functioned as intended.

11.5.5 Documentation impact

The base README.rst file will be updated to explain how the role can be used as a standalone role.

11.5.6 References

n/a

11.6 IRR - Galera

date
2015-11-01

tags
independent-role-repositories, galera

Split out the galera_server and galera_client roles into its own repository.

11.6.1 Problem description

Roles are all contained within a single monolithic repository making it impossible/difficult to consume the OSA roles outside of deploying the entire stack.

11.6.2 Proposed change

To ensure that the OSA project is consumable by other stacks using different architectures, deployment methods, and capabilities the role galera_server and galera_client need to be moved from the monolithic stack and into the its own role repository.

Alternatives

Leave everything the way it is. However doing that will hurt general OSA adoption.

Playbook/Role impact

- No impact to the playbooks.
- The role will be removed from the main stack. The plugins, filters, and libraries may need to be locally updated.

Upgrade impact

While the change will impact the placement of the role it will not impact upgrade-ability of the stack. The general workflow will need to be updated to ensure that users are updating roles on upgrade using the Ansible galaxy interface however generally speaking this is already being done for the deployer when running the `bootstrap-ansible.sh` script.

Security impact

n/a

Performance impact

Moving the role to an external repository will cause an impact in time to role resolution however that impact should be minimal.

End user impact

n/a

Deployer impact

Deployers will need to be aware of the new role locations and how to update existing roles however this should be minimal considering the tooling for updating existing roles already exists

Developer impact

Developers will need focus work within the roles which will exist within separate repositories.

Dependencies

n/a

11.6.3 Implementation

Assignee(s)

Primary assignee:
https://launchpad.net/~kevin-carter (IRC: cloudnull)

Work items

- With the role moved, tests will be created within the role via OpenStack CI to ensure that the role is performing the actions that its supposed to.
- Updated documentation via the README.rst will be created to show how the role can be used standalone.
- Example local inventory will be created to show how the role can be used. The local only inventory will also be used for testing the role.

11.6.4 Testing

- The test cases will deploy the role into a regular DSVM image
- The role will execute itself locally
- Once the role has completed an Ansible test play will run through several assert tasks to ensure the role functioned as intended.

11.6.5 Documentation impact

The base README.rst file will be updated to explain how the role can be used as a standalone role.

11.6.6 References

n/a

11.7 IRR - LXC Container Create

date
2015-11-01

tags
independent-role-repositories, lxc_container_create

Split out the lxc container create role into its own repository.

11.7.1 Problem description

Roles are all contained within a single monolithic repository making it impossible/difficult to consume the OSA roles outside of deploying the entire stack.

11.7.2 Proposed change

To ensure that the OSA project is consumable by other stacks using different architectures, deployment methods, and capabilities the role lxc_container_create need to be moved from the monolithic stack and into the its own role repository.

Alternatives

Leave everything the way it is. However doing that will hurt general OSA adoption.

Playbook/Role impact

- No impact to the playbooks.
- The role will be removed from the main stack. The plugins, filters, and libraries may need to be locally updated.

Upgrade impact

While the change will impact the placement of the role it will not impact upgrade-ability of the stack. The general workflow will need to be updated to ensure that users are updating roles on upgrade using the Ansible galaxy interface however generally speaking this is already being done for the deployer when running the `bootstrap-ansible.sh` script.

Security impact

n/a

Performance impact

Moving the role to an external repository will cause an impact in time to role resolution however that impact should be minimal.

End user impact

n/a

Deployer impact

Deployers will need to be aware of the new role locations and how to update existing roles however this should be minimal considering the tooling for updating existing roles already exists

Developer impact

Developers will need focus work within the roles which will exist within separate repositories.

Dependencies

The implementation of this spec relies on the following spec(s):

- https://review.openstack.org/#/c/240965
- https://review.openstack.org/#/c/241159

11.7.3 Implementation

Assignee(s)

Primary assignee:
https://launchpad.net/~kevin-carter (IRC: cloudnull)

Work items

- With the role moved, tests will be created within the role via OpenStack CI to ensure that the role is performing the actions that its supposed to.
- Updated documentation via the README.rst will be created to show how the role can be used standalone.
- Example local inventory will be created to show how the role can be used. The local only inventory will also be used for testing the role.

11.7.4 Testing

- The test cases will deploy the role into a regular DSVM image
- The role will execute itself locally
- Once the role has completed an Ansible test play will run through several assert tasks to ensure the role functioned as intended.

11.7.5 Documentation impact

The base README.rst file will be updated to explain how the role can be used as a standalone role.

11.7.6 References

n/a

11.8 IRR - LXC Host

date
2015-11-01

tags
independent-role-repositories, lxc_host

Split out the lxc host role into its own repository.

11.8.1 Problem description

Roles are all contained within a single monolithic repository making it impossible/difficult to consume the OSA roles outside of deploying the entire stack.

11.8.2 Proposed change

To ensure that the OSA project is consumable by other stacks using different architectures, deployment methods, and capabilities the role lxc_host need to be moved from the monolithic stack and into the its own role repository.

Alternatives

Leave everything the way it is. However doing that will hurt general OSA adoption.

Playbook/Role impact

- No impact to the playbooks.
- The role will be removed from the main stack. The plugins, filters, and libraries may need to be locally updated.

Upgrade impact

While the change will impact the placement of the role it will not impact upgrade-ability of the stack. The general workflow will need to be updated to ensure that users are updating roles on upgrade using the Ansible galaxy interface however generally speaking this is already being done for the deployer when running the `bootstrap-ansible.sh` script.

Security impact

n/a

Performance impact

Moving the role to an external repository will cause an impact in time to role resolution however that impact should be minimal.

End user impact

n/a

Deployer impact

Deployers will need to be aware of the new role locations and how to update existing roles however this should be minimal considering the tooling for updating existing roles already exists

Developer impact

Developers will need focus work within the roles which will exist within separate repositories.

Dependencies

The implementation of this spec relies on the following spec(s):

- https://review.openstack.org/#/c/240965

11.8.3 Implementation

Assignee(s)

Primary assignee:
https://launchpad.net/~kevin-carter (IRC: cloudnull)

Work items

- With the role moved, tests will be created within the role via OpenStack CI to ensure that the role is performing the actions that its supposed to.
- Updated documentation via the README.rst will be created to show how the role can be used standalone.
- Example local inventory will be created to show how the role can be used. The local only inventory will also be used for testing the role.

11.8.4 Testing

- The test cases will deploy the role into a regular DSVM image
- The role will execute itself locally
- Once the role has completed an Ansible test play will run through several assert tasks to ensure the role functioned as intended.

11.8.5 Documentation impact

The base README.rst file will be updated to explain how the role can be used as a standalone role.

11.8.6 References

n/a

11.9 IRR - Memcached Server

date
: 2015-11-01

tags
: independent-role-repositories, memcached_server

Split out the memcached_server role into its own repository.

11.9.1 Problem description

Roles are all contained within a single monolithic repository making it impossible/difficult to consume the OSA roles outside of deploying the entire stack.

11.9.2 Proposed change

To ensure that the OSA project is consumable by other stacks using different architectures, deployment methods, and capabilities the role memcached_server need to be moved from the monolithic stack and into the its own role repository.

Alternatives

Leave everything the way it is. However doing that will hurt general OSA adoption.

Playbook/Role impact

- No impact to the playbooks.
- The role will be removed from the main stack. The plugins, filters, and libraries may need to be locally updated.

Upgrade impact

While the change will impact the placement of the role it will not impact upgrade-ability of the stack. The general workflow will need to be updated to ensure that users are updating roles on upgrade using the Ansible galaxy interface however generally speaking this is already being done for the deployer when running the `bootstrap-ansible.sh` script.

Security impact

n/a

Performance impact

Moving the role to an external repository will cause an impact in time to role resolution however that impact should be minimal.

End user impact

n/a

Deployer impact

Deployers will need to be aware of the new role locations and how to update existing roles however this should be minimal considering the tooling for updating existing roles already exists

Developer impact

Developers will need focus work within the roles which will exist within separate repositories.

Dependencies

n/a

11.9.3 Implementation

Assignee(s)

Primary assignee:
https://launchpad.net/~kevin-carter (IRC: cloudnull)

Work items

- With the role moved, tests will be created within the role via OpenStack CI to ensure that the role is performing the actions that its supposed to.
- Updated documentation via the README.rst will be created to show how the role can be used standalone.
- Example local inventory will be created to show how the role can be used. The local only inventory will also be used for testing the role.

11.9.4 Testing

- The test cases will deploy the role into a regular DSVM image
- The role will execute itself locally
- Once the role has completed an Ansible test play will run through several assert tasks to ensure the role functioned as intended.

11.9.5 Documentation impact

The base README.rst file will be updated to explain how the role can be used as a standalone role.

11.9.6 References

n/a

11.10 IRR - OpenStack Hosts

date
2015-11-01

tags
independent-role-repositories, openstack_hosts

Split out the OpenStack hosts role into its own repository.

11.10.1 Problem description

Roles are all contained within a single monolithic repository making it impossible/difficult to consume the OSA roles outside of deploying the entire stack.

11.10.2 Proposed change

To ensure that the OSA project is consumable by other stacks using different architectures, deployment methods, and capabilities the role openstack_hosts need to be moved from the monolithic stack and into the its own role repository.

Alternatives

Leave everything the way it is. However doing that will hurt general OSA adoption.

Playbook/Role impact

- No impact to the playbooks.
- The role will be removed from the main stack. The plugins, filters, and libraries may need to be locally updated.

Upgrade impact

While the change will impact the placement of the role it will not impact upgrade-ability of the stack. The general workflow will need to be updated to ensure that users are updating roles on upgrade using the Ansible galaxy interface however generally speaking this is already being done for the deployer when running the `bootstrap-ansible.sh` script.

Security impact

n/a

Performance impact

Moving the role to an external repository will cause an impact in time to role resolution however that impact should be minimal.

End user impact

n/a

Deployer impact

Deployers will need to be aware of the new role locations and how to update existing roles however this should be minimal considering the tooling for updating existing roles already exists

Developer impact

Developers will need focus work within the roles which will exist within separate repositories.

Dependencies

The implementation of this spec relies on the following spec(s):

- https://review.openstack.org/#/c/240965

11.10.3 Implementation

Assignee(s)

Primary assignee:
https://launchpad.net/~kevin-carter (IRC: cloudnull)

Work items

- With the role moved, tests will be created within the role via OpenStack CI to ensure that the role is performing the actions that its supposed to.
- Updated documentation via the README.rst will be created to show how the role can be used standalone.
- Example local inventory will be created to show how the role can be used. The local only inventory will also be used for testing the role.

11.10.4 Testing

- The test cases will deploy the role into a regular DSVM image
- The role will execute itself locally
- Once the role has completed an Ansible test play will run through several assert tasks to ensure the role functioned as intended.

11.10.5 Documentation impact

The base README.rst file will be updated to explain how the role can be used as a standalone role.

11.10.6 References

n/a

11.11 IRR - pip install

date
2015-11-01

tags
independent-role-repositories, pip_install

Split out the pip install role into its own repository.

11.11.1 Problem description

Roles are all contained within a single monolithic repository making it impossible/difficult to consume the OSA roles outside of deploying the entire stack.

11.11.2 Proposed change

To ensure that the OSA project is consumable by other stacks using different architectures, deployment methods, and capabilities the role pip_install need to be moved from the monolithic stack and into the its own role repository.

Alternatives

Leave everything the way it is. However doing that will hurt general OSA adoption.

Playbook/Role impact

- No impact to the playbooks.
- The role will be removed from the main stack. The plugins, filters, and libraries may need to be locally updated.

Upgrade impact

While the change will impact the placement of the role it will not impact upgrade-ability of the stack. The general workflow will need to be updated to ensure that users are updating roles on upgrade using the Ansible galaxy interface however generally speaking this is already being done for the deployer when running the `bootstrap-ansible.sh` script.

Security impact

n/a

Performance impact

Moving the role to an external repository will cause an impact in time to role resolution however that impact should be minimal.

End user impact

n/a

Deployer impact

Deployers will need to be aware of the new role locations and how to update existing roles however this should be minimal considering the tooling for updating existing roles already exists

Developer impact

Developers will need focus work within the roles which will exist within separate repositories.

Dependencies

n/a

11.11.3 Implementation

Assignee(s)

Primary assignee:
https://launchpad.net/~kevin-carter (IRC: cloudnull)

Work items

- With the role moved, tests will be created within the role via OpenStack CI to ensure that the role is performing the actions that its supposed to.
- Updated documentation via the README.rst will be created to show how the role can be used standalone.
- Example local inventory will be created to show how the role can be used. The local only inventory will also be used for testing the role.

11.11.4 Testing

- The test cases will deploy the role into a regular DSVM image
- The role will execute itself locally
- Once the role has completed an Ansible test play will run through several assert tasks to ensure the role functioned as intended.

11.11.5 Documentation impact

The base README.rst file will be updated to explain how the role can be used as a standalone role.

11.11.6 References

n/a

11.12 IRR - pip_lock_down

date
2015-11-01

tags
independent-role-repositories, pip_lock_down

Split out the pip_lock_down role into its own repository.

11.12.1 Problem description

Roles are all contained within a single monolithic repository making it impossible/difficult to consume the OSA roles outside of deploying the entire stack.

11.12.2 Proposed change

To ensure that the OSA project is consumable by other stacks using different architectures, deployment methods, and capabilities the role pip_lock_down need to be moved from the monolithic stack and into the its own role repository.

Alternatives

Leave everything the way it is. However doing that will hurt general OSA adoption.

Playbook/Role impact

- No impact to the playbooks.
- The role will be removed from the main stack. The plugins, filters, and libraries may need to be locally updated.

Upgrade impact

While the change will impact the placement of the role it will not impact upgrade-ability of the stack. The general workflow will need to be updated to ensure that users are updating roles on upgrade using the Ansible galaxy interface however generally speaking this is already being done for the deployer when running the `bootstrap-ansible.sh` script.

Security impact

n/a

Performance impact

Moving the role to an external repository will cause an impact in time to role resolution however that impact should be minimal.

End user impact

n/a

Deployer impact

Deployers will need to be aware of the new role locations and how to update existing roles however this should be minimal considering the tooling for updating existing roles already exists

Developer impact

Developers will need focus work within the roles which will exist within separate repositories.

Dependencies

n/a

11.12.3 Implementation

Assignee(s)

Primary assignee:
https://launchpad.net/~kevin-carter (IRC: cloudnull)

Work items

- With the role moved, tests will be created within the role via OpenStack CI to ensure that the role is performing the actions that its supposed to.
- Updated documentation via the README.rst will be created to show how the role can be used standalone.
- Example local inventory will be created to show how the role can be used. The local only inventory will also be used for testing the role.

11.12.4 Testing

- The test cases will deploy the role into a regular DSVM image
- The role will execute itself locally
- Once the role has completed an Ansible test play will run through several assert tasks to ensure the role functioned as intended.

11.12.5 Documentation impact

The base README.rst file will be updated to explain how the role can be used as a standalone role.

11.12.6 References

n/a

11.13 IRR - RabbitMQ server

date
2015-11-01

tags
independent-role-repositories, rabbitmq_server

Split out the rabbitmq_server role into its own repository.

11.13.1 Problem description

Roles are all contained within a single monolithic repository making it impossible/difficult to consume the OSA roles outside of deploying the entire stack.

11.13.2 Proposed change

To ensure that the OSA project is consumable by other stacks using different architectures, deployment methods, and capabilities the role rabbitmq_server need to be moved from the monolithic stack and into the its own role repository.

Alternatives

Leave everything the way it is. However doing that will hurt general OSA adoption.

Playbook/Role impact

- No impact to the playbooks.
- The role will be removed from the main stack. The plugins, filters, and libraries may need to be locally updated.

Upgrade impact

While the change will impact the placement of the role it will not impact upgrade-ability of the stack. The general workflow will need to be updated to ensure that users are updating roles on upgrade using the Ansible galaxy interface however generally speaking this is already being done for the deployer when running the `bootstrap-ansible.sh` script.

Security impact

n/a

Performance impact

Moving the role to an external repository will cause an impact in time to role resolution however that impact should be minimal.

End user impact

n/a

Deployer impact

Deployers will need to be aware of the new role locations and how to update existing roles however this should be minimal considering the tooling for updating existing roles already exists

Developer impact

Developers will need focus work within the roles which will exist within separate repositories.

Dependencies

n/a

11.13.3 Implementation

Assignee(s)

Primary assignee:
https://launchpad.net/~kevin-carter (IRC: cloudnull)

Work items

- With the role moved, tests will be created within the role via OpenStack CI to ensure that the role is performing the actions that its supposed to.
- Updated documentation via the README.rst will be created to show how the role can be used standalone.
- Example local inventory will be created to show how the role can be used. The local only inventory will also be used for testing the role.

11.13.4 Testing

- The test cases will deploy the role into a regular DSVM image
- The role will execute itself locally
- Once the role has completed an Ansible test play will run through several assert tasks to ensure the role functioned as intended.

11.13.5 Documentation impact

The base README.rst file will be updated to explain how the role can be used as a standalone role.

11.13.6 References

n/a

11.14 IRR - Repo Server

date
2015-11-01

tags
independent-role-repositories, repo_server

Split out the repo server role into its own repository.

11.14.1 Problem description

Roles are all contained within a single monolithic repository making it impossible/difficult to consume the OSA roles outside of deploying the entire stack.

11.14.2 Proposed change

To ensure that the OSA project is consumable by other stacks using different architectures, deployment methods, and capabilities the role repo_server need to be moved from the monolithic stack and into the its own role repository.

Alternatives

Leave everything the way it is. However doing that will hurt general OSA adoption.

Playbook/Role impact

- No impact to the playbooks.
- The role will be removed from the main stack. The plugins, filters, and libraries may need to be locally updated.

Upgrade impact

While the change will impact the placement of the role it will not impact upgrade-ability of the stack. The general workflow will need to be updated to ensure that users are updating roles on upgrade using the Ansible galaxy interface however generally speaking this is already being done for the deployer when running the `bootstrap-ansible.sh` script.

Security impact

n/a

Performance impact

Moving the role to an external repository will cause an impact in time to role resolution however that impact should be minimal.

End user impact

n/a

Deployer impact

Deployers will need to be aware of the new role locations and how to update existing roles however this should be minimal considering the tooling for updating existing roles already exists

Developer impact

Developers will need focus work within the roles which will exist within separate repositories.

Dependencies

n/a

11.14.3 Implementation

Assignee(s)

Primary assignee:
https://launchpad.net/~kevin-carter (IRC: cloudnull)

Work items

- With the role moved, tests will be created within the role via OpenStack CI to ensure that the role is performing the actions that its supposed to.
- Updated documentation via the README.rst will be created to show how the role can be used standalone.
- Example local inventory will be created to show how the role can be used. The local only inventory will also be used for testing the role.

11.14.4 Testing

- The test cases will deploy the role into a regular DSVM image
- The role will execute itself locally
- Once the role has completed an Ansible test play will run through several assert tasks to ensure the role functioned as intended.

11.14.5 Documentation impact

The base README.rst file will be updated to explain how the role can be used as a standalone role.

11.14.6 References

n/a

11.15 IRR - rsyslog client

date
2015-11-01

tags
independent-role-repositories, rsyslog_client

Split out the rsyslog client role into its own repository.

11.15.1 Problem description

Roles are all contained within a single monolithic repository making it impossible/difficult to consume the OSA roles outside of deploying the entire stack.

11.15.2 Proposed change

To ensure that the OSA project is consumable by other stacks using different architectures, deployment methods, and capabilities the role rsyslog_client need to be moved from the monolithic stack and into the its own role repository.

Alternatives

Leave everything the way it is. However doing that will hurt general OSA adoption.

Playbook/Role impact

- No impact to the playbooks.
- The role will be removed from the main stack. The plugins, filters, and libraries may need to be locally updated.

Upgrade impact

While the change will impact the placement of the role it will not impact upgrade-ability of the stack. The general workflow will need to be updated to ensure that users are updating roles on upgrade using the Ansible galaxy interface however generally speaking this is already being done for the deployer when running the `bootstrap-ansible.sh` script.

Security impact

n/a

Performance impact

Moving the role to an external repository will cause an impact in time to role resolution however that impact should be minimal.

End user impact

n/a

Deployer impact

Deployers will need to be aware of the new role locations and how to update existing roles however this should be minimal considering the tooling for updating existing roles already exists

Developer impact

Developers will need focus work within the roles which will exist within separate repositories.

Dependencies

n/a

11.15.3 Implementation

Assignee(s)

Primary assignee:
https://launchpad.net/~kevin-carter (IRC: cloudnull)

Work items

- With the role moved, tests will be created within the role via OpenStack CI to ensure that the role is performing the actions that its supposed to.
- Updated documentation via the README.rst will be created to show how the role can be used standalone.
- Example local inventory will be created to show how the role can be used. The local only inventory will also be used for testing the role.

11.15.4 Testing

- The test cases will deploy the role into a regular DSVM image
- The role will execute itself locally
- Once the role has completed an Ansible test play will run through several assert tasks to ensure the role functioned as intended.

11.15.5 Documentation impact

The base README.rst file will be updated to explain how the role can be used as a standalone role.

11.15.6 References

n/a

11.16 IRR - rsyslog_server

date
2015-11-01

tags
independent-role-repositories, rsyslog_server

Split out the rsyslog_server role into its own repository.

11.16.1 Problem description

Roles are all contained within a single monolithic repository making it impossible/difficult to consume the OSA roles outside of deploying the entire stack.

11.16.2 Proposed change

To ensure that the OSA project is consumable by other stacks using different architectures, deployment methods, and capabilities the role rsyslog_server need to be moved from the monolithic stack and into the its own role repository.

Alternatives

Leave everything the way it is. However doing that will hurt general OSA adoption.

Playbook/Role impact

- No impact to the playbooks.
- The role will be removed from the main stack. The plugins, filters, and libraries may need to be locally updated.

Upgrade impact

While the change will impact the placement of the role it will not impact upgrade-ability of the stack. The general workflow will need to be updated to ensure that users are updating roles on upgrade using the Ansible galaxy interface however generally speaking this is already being done for the deployer when running the `bootstrap-ansible.sh` script.

Security impact

n/a

Performance impact

Moving the role to an external repository will cause an impact in time to role resolution however that impact should be minimal.

End user impact

n/a

Deployer impact

Deployers will need to be aware of the new role locations and how to update existing roles however this should be minimal considering the tooling for updating existing roles already exists

Developer impact

Developers will need focus work within the roles which will exist within separate repositories.

Dependencies

n/a

11.16.3 Implementation

Assignee(s)

Primary assignee:
https://launchpad.net/~kevin-carter (IRC: cloudnull)

Work items

- With the role moved, tests will be created within the role via OpenStack CI to ensure that the role is performing the actions that its supposed to.
- Updated documentation via the README.rst will be created to show how the role can be used standalone.
- Example local inventory will be created to show how the role can be used. The local only inventory will also be used for testing the role.

11.16.4 Testing

- The test cases will deploy the role into a regular DSVM image
- The role will execute itself locally
- Once the role has completed an Ansible test play will run through several assert tasks to ensure the role functioned as intended.

11.16.5 Documentation impact

The base README.rst file will be updated to explain how the role can be used as a standalone role.

11.16.6 References

n/a

11.17 IRR - Utility

date
2015-11-01

tags
independent-role-repositories, utility

Split out the utility role into its own repository.

11.17.1 Problem description

Roles are all contained within a single monolithic repository making it impossible/difficult to consume the OSA roles outside of deploying the entire stack.

11.17.2 Proposed change

To ensure that the OSA project is consumable by other stacks using different architectures, deployment methods, and capabilities the role utility need to be moved from the monolithic stack and into the its own role repository.

Alternatives

Leave everything the way it is. However doing that will hurt general OSA adoption.

Playbook/Role impact

- No impact to the playbooks.
- The role will be removed from the main stack. The plugins, filters, and libraries may need to be locally updated.

Upgrade impact

While the change will impact the placement of the role it will not impact upgrade-ability of the stack. The general workflow will need to be updated to ensure that users are updating roles on upgrade using the Ansible galaxy interface however generally speaking this is already being done for the deployer when running the `bootstrap-ansible.sh` script.

Security impact

n/a

Performance impact

Moving the role to an external repository will cause an impact in time to role resolution however that impact should be minimal.

End user impact

n/a

Deployer impact

Deployers will need to be aware of the new role locations and how to update existing roles however this should be minimal considering the tooling for updating existing roles already exists

Developer impact

Developers will need focus work within the roles which will exist within separate repositories.

Dependencies

n/a

11.17.3 Implementation

Assignee(s)

Primary assignee:
https://launchpad.net/~kevin-carter (IRC: cloudnull)

Work items

- With the role moved, tests will be created within the role via OpenStack CI to ensure that the role is performing the actions that its supposed to.
- Updated documentation via the README.rst will be created to show how the role can be used standalone.
- Example local inventory will be created to show how the role can be used. The local only inventory will also be used for testing the role.

11.17.4 Testing

- The test cases will deploy the role into a regular DSVM image
- The role will execute itself locally
- Once the role has completed an Ansible test play will run through several assert tasks to ensure the role functioned as intended.

11.17.5 Documentation impact

The base README.rst file will be updated to explain how the role can be used as a standalone role.

11.17.6 References

n/a

11.18 Load Balancers v2 (LBaaSv2 & octavia)

date
2016-01-28 00:00

tags
lbaasv2, octavia, load balancing, neutron

Blueprint: Load Balancers v2 (LBaaSv2 & octavia)

- https://blueprints.launchpad.net/openstack-ansible/+spec/lbaasv2

OpenStack-Ansible currently offers LBaaSv1, but it is deprecated in Liberty and it is scheduled to be removed in Newton. LBaaSv2 became stable in Liberty and it provides several improvements over LBaaSv1:

- **Multiple TCP ports per load balancer:** This helps when hosting websites which must serve visitors on ports 80 and 443. It can also reduce the number of floating IP address that are required for deployment.
- **Failover support:** Deployers have the option to deploy a single load balancer node or an active/passive pair.
- **Load balancers inside guests:** The load balancer runs inside virtual machines rather than alongside other neutron agents.
- **Housekeeping:** If a load balancer goes offline or haproxy is down, the health manager will remove the faulty load balancer and build a new one in its place.
- **TLS Termination:** TLS can be terminated at the load balancer so that individual virtual machines arent required to encrypt/decrypt traffic.

However, LBaaSv2 does have some limitations:

- **Horizon support:** Horizon support for LBaaSv2 is *planned* for Mitaka and work is currently in progress, but the panels are not yet available at the time of writing of this spec.
- **Cannot run alongside v1:** It is not possible for both versions of LBaaS to run at the same time, and there is not a migration path available today for users who want to migrate their v1 load balancers to v2.

11.18.1 Problem description

Although LBaaSv1 support exists in OpenStack-Ansible already, it is deprecated in Liberty and it is scheduled for removal in Newton. It also has a limitation of one listening port per load balancer, which limits a users ability to host an application on more than one port (think HTTP and HTTPS) on a single load balancer.

LBaaSv2 replaces LBaaSv1 and will be supported by OpenStack developers going forward.

11.18.2 Proposed change

There will be several changes needed to deploy LBaaSv2:

1. **Add octavia to existing neutron virtualenv:** The octavia project will need to be included with the neutron virtualenv that is deployed within the neutron-server container.
2. **Manage four octavia daemons:** Four daemons will need to run in the neutron-server container: octavia-api, octavia-housekeeping, octavia-worker, and octavia-health-monitor.
3. **Deploy octavias configuration file:** The `/etc/octavia/octavia.conf` will need to be deployed and managed.
4. **RabbitMQ/MariaDB credentials:** Octavia will require its own database, database credentials, and RabbitMQ credentials.
5. **Neutron configuration changes:** Changes will be needed in `neutron.conf` to add the LBaaSv2 service plugin and driver.
6. **Update documentation:** Deployers would need to know the difference between both load balancer implementations and how to deploy each. Deployers using LBaaSv1 would also need some advice on how to handle a change to LBaaSv2 in their OpenStack environments.

Alternatives

We could choose to stay with v1 through the Mitaka release, but then we might be forced to remove it for the Newton release in favor of v2.

Playbook/Role impact

The vast majority of the changes should take place in the neutron role. LBaaSv1 is currently enabled by adding a service_plugin entry, and the same could be done for LBaaSv2. When the LBaaSv2 service_plugin entry is present, the neutron role would ensure that octavia is running and ready to receive requests.

Upgrade impact

The upgrade impact depends on whether a deployer is currently using LBaaSv1.

If a deployer is already using LBaaSv1, they would need to carefully consider their migration path to v2 since both implementations cannot run concurrently. However, if a deployer is already using v1, they could upgrade to Liberty or Mitaka without making any adjustments to how they use load balancers. Upgrading to Newton would require changes since LBaaSv1 is expected to be removed in that release.

If a deployer is not using LBaaSv1 at this time, then they would simply be gaining functionality that they did not have previously when they upgrade to Liberty, Mitaka, or Newton.

Security impact

There are no sigificant performance impacts within LBaaSv2, but it could allow deployers to deploy HTTPS websites on the same IP address as their HTTP site.

Octavia also integrates with the Anchor and Barbican projects. Anchor allows users to obtain certificates from a pre-configured certificate authority (CA) within their OpenStack environment and Barbican can be used to store private keys for SSL/TLS connections.

Performance impact

LBaaSv2 should scale better than LBaaSv1 since it runs within virtual machines rather than inside the neutron-agents container with multiple haproxy-agents.

End user impact

If an end user is not currently using LBaaSv1, then they would be gaining a new feature that they could consume via an API. Horizon panels are planned for Mitaka.

If an end user is currently using LBaaSv1, they would lose load balancer functionality when LBaaSv2 is deployed until they configure their load balancers within the LBaaSv2 API. Deployers must work closely with end users to determine the best path forward.

Deployer impact

If a deployer is not currently using LBaaSv1, they could easily enable LBaaSv2 by adding a service_plugin within their `openstack_user_config.yml` for LBaaSv2, just as they do for LBaaSv1 today.

If a deployer is currently using LBaaSv1, they could continue using it without interruption (until the Newton release). If they choose to move to LBaaSv2, they would need to coordinate the changes with their end users to avoid lengthy service interruptions.

Developer impact

The LBaaSv2 deployment would be part of the neutron deployment, much like the current deployment process for LBaaSv1. No additional roles or playbooks are required.

Dependencies

This spec does not depend on any other development work in OpenStack-Ansible.

11.18.3 Implementation

Assignee(s)

Primary assignee:
Major Hayden (IRC: mhayden, LP: rackerhacker)

Other contributors:
None

Work items

See the details in the *Proposed Changes* section above.

11.18.4 Testing

Tempest testing exists for the LBaaSv2 API but tempest tests for the octavia API are still in progress.

11.18.5 Documentation impact

Some topics are mentioned above in the *Proposed Changes* section. The following topics must be documented:

- What is different between LBaaSv1/2?
- What do I do if I already deployed LBaaSv1?
- How do I migrate from v1 to v2?
- How do I deploy/configure LBaaSv2?
- How do I troubleshoot LBaaSv2 issues?

11.18.6 References

Mailing list discussion:

- LBaaSv2 / Octavia support

Software:

- https://github.com/openstack/octavia
- https://github.com/openstack/neutron-lbaas

Documentation:

- LBaaSv2 in Devstack
- Load Balancing as a Service v2.0 - Liberty and Beyond (PDF)
- Networking API v2.0 extensions

11.19 Limit Mysql Config Distribution

date
: 2015-07-20

tags
: mysql, galera

- https://blueprints.launchpad.net/openstack-ansible/+spec/limit-mysql-config-distribution

11.19.1 Problem description

The distribution of the `.my.cnf` file should be limited to API nodes and the utility container.

11.19.2 Proposed change

- Add a variable to the galera_client role to limit the distribution of the `.my.cnf` file.

Alternatives

Leave everything the way it is.

Playbook/Role impact

This will change the galera_client and os_* roles to ensure that the `.my.cnf` files are only distributed to a limited set of hosts.

Upgrade impact

n/a

Security impact

By limiting the distribution of the `.my.cnf` file we should be able to improve general system security.

Performance impact

n/a

End user impact

n/a

Deployer impact

n/a

Developer impact

n/a

Dependencies

n/a

11.19.3 Implementation

Assignee(s)

Primary assignee: (unassigned)

Work items

- Add a variable to the galera_client role to disable the task Drop local .my.cnf file
- Change the meta entries where the **galera_client** roles is used use the new variable where appropriate.

11.19.4 Testing

This will be tested within every gate check for functionality.

11.19.5 Documentation impact

n/a

11.19.6 References

Bug reference for the change:

- https://bugs.launchpad.net/openstack-ansible/trunk/+bug/1412393

11.20 Modularize configuration files

date
2015-05-08 00:00

tags
config, configuration, modularize, modular

Modularize deployment configuration files to simplify the configuration process.

Blueprint:

https://blueprints.launchpad.net/openstack-ansible/+spec/modularize-config

11.20.1 Problem description

Deployment configuration primarily occurs in a rather monolithic `openstack_user_config.yml` file. Although adding documentation to this file eases understanding of the various levels and options, it also increases the size and apparant complexity, especially for larger deployments. With the addition of swift, the configuration structure already supports configuration files in the conf.d directory. Splitting the main monolithic configuration file into smaller files containing similar components helps overall organization, especially for larger deployments.

11.20.2 Proposed change

Similar to swift, modularize similar sections of configuration files, particularly `openstack_user_config.yml`, into the following separate files in the conf.d directory.

- hosts.yml
  - Includes configuration for target hosts with simple options. For example, `repo_hosts` typically contains only a list of hosts. In comparison, `storage_hosts` requires significantly more options and should therefore use a separate file.
  - Contains the following levels:
    * `shared-infra_hosts`

    * repo_hosts
    * os-infra_hosts
    * identity_hosts
    * network_hosts
    * compute_hosts
    * storage-infra_hosts
    * swift_proxy-hosts
    * log_hosts

Note: For consistency, consider changing swift_proxy-hosts to swift-proxy_hosts and swift_hosts to swift-storage_hosts.

- networking.yml
  - Includes host networks, IP address blacklist for inventory generator, load balancer options, and provider networks.
  - Contains the following levels:
    * cidr_networks
    * used_ips
    * provider_networks (from global_overrides)
    * internal_lb_vip_address, external_lb_vip_address, management_bridge, and tunnel_bridge (from global_overrides)
- cinder_storage_hosts.yml
  - Includes configuration for cinder storage target hosts with complex options for backends.
  - Contains the following level: * storage_hosts
- swift_storage_hosts.yml
  - Includes configuration for swift storage target hosts with complex options.
  - Contains the following levels: * swift (from global_overrides) * swift_hosts

Alternatives

Use a different strategy to modularize the configuration files or keep the existing monolithic structure.

Playbook impact

None.

Upgrade impact

Optionally, modularize configuration files according to this specification before or after upgrading to a version that supports it.

Security impact

None.

Performance impact

None.

End user impact

None.

Deployer impact

Simplify the configuration process.

Developer impact

Developers should consider the modular configuration when adding or changing configuration items.

Dependencies

None.

11.20.3 Implementation

Assignee(s)

Primary assignee:
<ionosphere80> **Sam-I-Am**

Work items

- As necessary, break existing monolithic configuration files into smaller files that contain groups of similar items and reside in a `.d` directory within the configuration file structure.

11.20.4 Testing

- Verify changes do not break gating process. The AIO script for gating can continue to use a monolithic file or modular files with the `.aio` extension.

11.20.5 Documentation impact

- Change documentation that references monolithic configuration files to reference modular configuration files.

11.20.6 References

None.

11.21 Policy Files Distribution to Horizon

date
2015-11-24 15:00

tags
cross-project, cross-role, json, policy, distribution, Horizon

OpenStack Horizon can use policy.json files to filter the actions available on its webinterface. For that, Horizon consumes the policy.json files of each openstack project (like cinder/nova/glance/), it doesnt distribute its own.

Therefore, if the deployer wants to have a consistent policy through the apis and the webinterface, the deployer has to upload its policies to Horizon.

Currently, its not done within openstack-ansible.

https://blueprints.launchpad.net/openstack-ansible/+spec/policy-files-distribution

11.21.1 Problem description

Currently every deployer that needs policy files is doing the same work. Lets try to avoid that in the future: They create policy files for the openstack project thanks to openstack-ansible but then need to upload the policy files to Horizon manually with their own role.

This should fix that, and propose a solution to the policy files deployment

11.21.2 Proposed change

First, there should be a generic cross-role switch (`policy_file_distribution_enabled`) defaulted to False, unless the deployer has set a `_policy_overrides` for a component. Of course, a deployer can prevent this policy file distribution by setting it to False.

Then, we should handle the policy distribution in two steps:

1. Download each deployed policy.json file (from the first host of each group) during the `horizon-install` playbook into the `policy_files_distribution_folder` (by default `/etc/openstack_deploy/`) on the deploy node.
2. Having the Horizon role could consume these files on the deploy host and upload the json files to the Horizon nodes. This would require connecting on multiple hosts and will lengthen deployments time (on the first run, if enabled)

Alternatives

- Not implementing this, and let the deployer do the work himself
- Rely on Horizon distributing its own policy mapping in the future
- Include each projects (i.e. nova,neutron,etc.) default policy file from their git source in the Horizon role and use the *config_template* to upload/override the final `nova_policy.json`, `glance_policy.json,...` files on Horizon. This would require us to track OpenStack project policy changes in both Horizon and the respective project roles.
- Download each project policy.json file from their git source repository (i.e. glance, nova,etc.) to the deployment node before running the os_horizon role. Then use the *config_template* to upload/override the final json files on Horizon. This would require us to track OpenStack projects policy files URL changes.
- Last alternative would be to distribute using another mechanism (like memcache/swift/file sync).

Playbook/Role impact

Small changes in playbooks/role.

Upgrade impact

No upgrade impact.

Security impact

None

Performance impact

Slightly longer deployment time if enabled for the first time. Implementation would redownload if a file exists, unless explicitly told by a variable: `policy_file_distribution_force_refresh`.

End user impact

The end-user will not have inconsistent behaviour of having one button that doesnt work because the policy prevents it in the component api but not in Horizon.

Deployer impact

A few new variables:

- policy_file_distribution_enabled
- policy_file_distribution_force_refresh
- policy_files_distribution_files

NB: Their name could be adapted later (cf. implementation)

Developer impact

Nothing should change.

Dependencies

None.

11.21.3 Implementation

Assignee(s)

None yet

Work items

1. group_var to define auto download
2. playbook edition to download policies
3. role changes to upload json files

11.21.4 Testing

- Does this change impact how gating is done?

No

- Can this change be tested on a **per-commit** basis?

Yes

- Given the instance size restrictions, as found in OpenStack Infra (8GB Ram, vCPUs <= 8), can the test be run in a resource constrained environment?

Yes.

- Is this untestable given current limitations (specific hardware / software configurations available)? If so, are there mitigation plans for this change to be tested within 3rd party testing, gate enhancements, etc?

This change is testable.

- If the service is not OpenStack specific how can we test the change?

Its openstack specific

11.21.5 Documentation impact

Well need to update the documentation to mention how to edit the policies and how to enable the policy distribution to Horizon.

11.21.6 References

Policy files url:

- Horizon: http://docs.openstack.org/developer/horizon/topics/policy.html#policy-files
- keystone: https://github.com/openstack/keystone/blob/master/etc/policy.json
- Glance: https://github.com/openstack/glance/blob/master/etc/policy.json
- Nova: https://github.com/openstack/nova/blob/master/etc/nova/policy.json
- Neutron: https://github.com/openstack/neutron/blob/master/etc/policy.json
- Cinder: https://github.com/openstack/cinder/blob/master/etc/cinder/policy.json

11.22 Additional Role for Designate Deployment

date
: 2015-12-08 12:00

tags
: designate, openstack-ansible

The purpose of this spec is to add support for the OpenStack Designate program to OpenStack-Ansible. This would allow the deployment of Designate along with the core OpenStack components using OpenStack-Ansible.

Blueprint - Designate deployment on OpenStack-Ansible:

https://blueprints.launchpad.net/openstack-ansible/+spec/role-designate

11.22.1 Problem description

Presently, while deploying OpenStack using OpenStack-Ansible only the core OpenStack components get deployed. The deployment of other components (eg: Designate, Trove) on playbooks is not supported yet and to use other components services, they need to be deployed manually.

11.22.2 Proposed change

The Designate program encompasses a number of projects, but this spec and this proposed series of changes covers the initial implementation of support for Designate. This will involve adding support for the Designate server[1] and Designate client[2].

The proposed changes include:

- Creation of an openstack-ansible-designate repository and Ansible role to support the deployment of Designate.
- Tests to verify the new Ansible role.

Alternatives

None

Playbook/Role impact

Test playbooks will be placed in the openstack-ansible-designate repository for functional testing purposes, with no initially proposed changes to OpenStack-Ansible playbooks.

In the future, once the Designate role is found to be useful and acceptable, a future spec will address the integration of the Designate role with the main OpenStack-Ansible repository.

Upgrade impact

None

Security impact

None.

Performance impact

None.

End user impact

Deployers will be able to deploy Designate and use DNSaaS through OpenStack-Ansible.

Deployer impact

When support for the new Designate role is added to the parent repository, new Designate specific configuration options will be made available. This will provide an optional role for use in the OpenStack-Ansible toolbox for the deployers.

Developer impact

As this change is self-contained initially, no impact on other developers is expected.

Dependencies

None

11.22.3 Implementation

Assignee(s)

Primary assignee:
Swati Sharma (IRC: Swati)

Other contributors:
None

Work items

1. Ask for the new repository, openstack-ansible-designate, to be created
2. Create the role for Designate support
   - Add support for running designate-api, designate-central, designate-pool_manager, designate-sink, designate-mdns
   - Add support for including python-designateclient, which is the operator tool for supporting Designate.

11.22.4 Testing

The usual gate checks can be used for these changes. Also, each individual commit can be functionally tested individually.

11.22.5 Documentation impact

Adding support to the user guide on how to enable Designate support will be required.

11.22.6 References

- [1] The Designate server: http://git.openstack.org/cgit/openstack/designate/
- [2] The Designate client: http://git.openstack.org/cgit/openstack/python-designateclient/

11.23 Role Ironic

date
2015-10-12 16:30

tags
ansible, ironic

The purpose of this spec is to add support for the OpenStack Ironic program to OpenStack Ansible, allowing the provisioning of compute nodes to bare metal machines.

https://blueprints.launchpad.net/openstack-ansible/+spec/role-ironic

11.23.1 Problem description

Openstack Ansible currently does not support the provisioning of bare metal compute hosts, but this is functionality that operators and users are likely to want.

11.23.2 Proposed change

The Ironic program encompasses a number of projects, but this spec and this proposed series of changes covers the initial implementation of support for Ironic. This will involve adding support for the Ironic server[1] and Ironic client[2].

Future specs may be raised to cover the addition of ironic-inspector, or to support alternate deployment mechanisms, or to support different deployment drivers. The specific detail for these will be added in future specs.

This work will build upon the experiences learnt in developing bifrost[3] (which is a set of ansible playbooks for deploying Ironic standalone, without other OpenStack components).

The changes that are proposed as part of this spec are:

- Creation of an openstack-ansible-ironic repository and ansible role to support the initial implementation of Ironic. This will allow openstack-ansible to deploy compute nodes to bare metal hosts,

via the nova API. Initially, support will be limited to bare metal hosts that support IPMI for power control, and PXE for boot.

- Tests to verify the new ansible role

Alternatives

None, really. Supporting bare metal hosts in OpenStack is done via using Ironic.

Playbook/Role impact

Test playbooks will be placed in the openstack-ansible-ironic repository for functional testing purposes, with no initially proposed changes to openstack-ansible playbooks.

In the future, once the ironic role is deemed useful and acceptible, a future spec will address the integration of the ironic role with the main openstack-ansible repository.

Upgrade impact

None

Security impact

None.

Performance impact

None.

End user impact

Deployers will be able to deploy compute nodes to bare metal hosts.

Deployer impact

Ironic specific configuration options will be added to the new repository. When support for the new Ironic role is added to the parent repository new config options will be made available, however it is expected that Ironic support will initially be disabled, requiring that deployers explicitly enable Ironic support, and to enrol hosts for openstack-ansible to use.

Developer impact

As this change is self-contained initially, no impact on other developers is expected.

Dependencies

None

11.23.3 Implementation

Assignee(s)

Primary assignee:
Michael Davies - mrda on Launchpad and on IRC

Other contributors:
None

Work items

1. Ask for the new repository, openstack-ansible-ironic, to be created
2. Create the role for ironic support
   - Add support for running ironic-api
   - Add support for running ironic-conductor
   - Add support for including python-ironicclient, which is the operator tool for supporting Ironic.
   - Add configuration to make configuring bare metal deployment easy
3. Add support for enrolling bare metal nodes
4. Add support for configuring Nova to use Ironic. Initially this will be in the form of documentation until the parent openstack-ansible repository is updated to use openstack-ansible-ironic

11.23.4 Testing

As this is testing deploying to hardware, this is challenging :)

Develop a test playbook to deploy to hardware that can exercise the new role. Develop tests that verify the roles behaviour independent of actually requiring hardware to test the roles functionality.

11.23.5 Documentation impact

Adding support to the user guide on how to enable Ironic support will be required.

11.23.6 References

- [1] The Ironic server: http://git.openstack.org/cgit/openstack/ironic/
- [2] The Ironic client: http://git.openstack.org/cgit/openstack/python-ironicclient/
- [3] The Bifrost project, standalone Ironic installation: http://git.openstack.org/cgit/openstack/bifrost

11.24 Additional Role for Zaqar Deployment

date
2016-01-20 11:20

tags
zaqar, openstack-ansible

The purpose of this spec is to add support for the OpenStack Zaqar program to OpenStack-Ansible. This would allow the deployment of Zaqar along with the core OpenStack components using OpenStack-Ansible.

Blueprint - Zaqar deployment on OpenStack-Ansible:

https://blueprints.launchpad.net/openstack-ansible/+spec/role-zaqar

11.24.1 Problem description

Presently, while deploying OpenStack using OpenStack-Ansible only the core OpenStack components get deployed. The deployment of other components (eg: Zaqar) on playbooks is not supported yet and to use other components services, they need to be deployed manually.

11.24.2 Proposed change

The Zaqar program encompasses a number of projects, but this spec and this proposed series of changes covers the initial implementation of support for Zaqar. This will involve adding support for the Zaqar server[1] and Zaqar client[2].

The proposed changes include:

- Creation of an openstack-ansible-zaqar repository and Ansible role to support the deployment of Zaqar.
- Tests to verify the new Ansible role.

Alternatives

None

Playbook/Role impact

Test playbooks will be placed in the openstack-ansible-zaqar repository for functional testing purposes, with no initially proposed changes to OpenStack-Ansible playbooks.

In the future, once the Zaqar role is found to be useful and acceptable, a future spec will address the integration of the Zaqar role with the main OpenStack-Ansible repository.

Upgrade impact

None

Security impact

None.

Performance impact

None.

End user impact

Deployers will be able to deploy Zaqar and use messaging service through OpenStack-Ansible.

Deployer impact

When support for the new Zaqar role is added to the parent repository, new Zaqar specific configuration options will be made available. This will provide an optional role for use in the OpenStack-Ansible toolbox for the deployers.

Developer impact

As this change is self-contained initially, no impact on other developers is expected.

Dependencies

None

11.24.3 Implementation

Assignee(s)

Primary assignee:
Fei Long Wang (IRC: flwang)

Other contributors:
None

Work items

1. Ask for the new repository, openstack-ansible-zaqar, to be created
2. Create the role for Zaqar support
   - Add support for running zaqar-sever
   - Add support for including python-zaqarclient, which is the operator tool for supporting Zaqar.

11.24.4 Testing

The usual gate checks can be used for these changes. Also, each individual commit can be functionally tested individually.

11.24.5 Documentation impact

Adding support to the user guide on how to enable Zaqar support will be required.

11.24.6 References

- [1] The Zaqar server: http://git.openstack.org/cgit/openstack/zaqar/
- [2] The Zaqar client: http://git.openstack.org/cgit/openstack/python-zaqarclient/

11.25 Security Hardening for Hosts

date
2015-09-10 00:00

tags
security

The goal of this spec is to apply hardening standards to openstack-ansible so that users can build environments that meet the requirements of various compliance programs, such as the Payment Card Industry Data Security Standard (PCI-DSS). These changes wont make a particular environment PCI compliant, but they should bring the environment in compliance with Requirement 2.2 from PCI-DSS. That requirement states that deployments must follow an industry-accepted hardening standard.

Blueprint - Security Hardening for OSAD Hosts:

- https://blueprints.launchpad.net/openstack-ansible/+spec/security-hardening

11.25.1 Problem description

Compliance programs, such as PCI-DSS, often have a requirement for using industry-accepted hardening standards for all deployments. At the moment, deployments done on Ubuntu 14.04 with openstack-ansible meet many, but not all, security hardening standards that are approved within PCI-DSS.

PCI-DSS 3.1 Requirement 2.2 states that deployments that handle credit card data must be secured with industry-accepted hardening standards. The test of the requirement is as follows:

> 2.2 Develop configuration standards for all system components. Assure that these standards address all known security vulnerabilities and are consistent with industry-accepted system hardening standards.

The United States Defense Information Systems Agency (DISA) publishes sets of security hardening guides called Security Technical Implementation Guides (STIGs). Theyre comprehensive and they provide mechanisms for checking secured systems for compliance with the standards. In addition, they are in the public domain.

11.25.2 Proposed change

The proposed changes include:

1. Create a new role in a new repo to hold the security tasks
   - See *Work items* below for specifics about the role.
2. Write documentation about the hardening standards applied
   - Is this standard already deployed by default in Ubuntu 14.04 or by OSAD already?
   - If a standard is applied, what does a deployer gain from it?
   - If a standard is skipped, why was it skipped and what does the deployer lose?
3. Submit patches that actually apply those hardening standards
   - Start by making a bug for each with a description of what will be changed and why
   - Determine whether the patch belongs in openstack-ansible or within a new security-hardening role that can be pulled into openstack-ansible during deployments
4. Create an automated way to test that the security changes are applied and they dont cause negative impacts on openstack-ansible deployments
   - This could be done via OpenSCAP or via CIS Java-based checker
   - Needs to be checked via gate check jobs
5. Make it easy for deployers to import the security hardening role into openstack-ansible

- Should be easily pulled into an openstack-ansible deployment if a deployer chooses

Here are several examples of security improvements recommended by the RHEL 6 STIG which apply well to Ubuntu:

- V-38497: The system must not have accounts configured with blank or null passwords.
- V-38476: Vendor-provided cryptographic certificates must be installed to verify the integrity of system software.
- V-38607: The SSH daemon must be configured to use only the SSHv2 protocol.
- V-38614: The SSH daemon must not allow authentication using an empty password.
- V-38673: The operating system must ensure unauthorized, security-relevant configuration changes detected are tracked.
- V-38632: The operating system must produce audit records containing sufficient information to establish what type of events occurred.

Alternatives

No known alternatives.

Playbook/Role impact

Depending on the nature of the change and the usefulness to deployers, the changes may be applied directly to existing roles in openstack-ansible or they may be applied to security hardening role that is optionally pulled in during openstack-ansible deployments

Any changes which could affect the performance, stability, or functionality of a production deployment would be disabled by default and heavily documented. Deployers could then make an educated decision on whether or not they want that security hardening standard enabled.

Upgrade impact

If security features are added via feature flags and disabled by default, the effect on upgrades would be very minimal if theyre even noticed at all. All configuration changes should be examined individually to determine if they will have an impact on upgrades.

Security impact

The entire goal of this spec is to have a positive security impact without becoming an operational burden.

Performance impact

Its possible that some security changes could impact the performance of a running OpenStack system. As noted in *Upgrade impact* above, these configuration changes would need to be examined individually to determine the balance between security and performance impacts.

End user impact

End users shouldnt notice the majority of the security changes. They will still interact with API endpoints and virtual machines as they do today. Theres a chance that some security improvements could impact an end user, but deployers will have full control of how those improvements are applied.

Deployer impact

Deployers could potentially be able to build OpenStack systems that are more secure by default. However, if these security features are disabled by default, we need solid documentation that tells users how to enable these features and what the impact of enabling those features might be.

Deployers would need to explicitly include the security hardening role within their openstack-ansible deployments.

Developer impact

Developers would need to include the security hardening role within their deployments if they wanted to test openstack-ansible with additional security enhancements.

Dependencies

This spec has no dependencies.

11.25.3 Implementation

Assignee(s)

Who is leading the writing of the code? Or is this a blueprint where youre throwing it out there to see who picks it up?

If more than one person is working on the implementation, please designate the primary author and contact.

Primary assignee:
Major Hayden (LP: rackerhacker, IRC: mhayden)

Other contributors:
Cody Bunch (LP: cody-bunch, IRC: e-vad)

Work items

The security hardening role should be in a separate repository titled `openstack-ansible-security`. Once the role has content and is well-tested against openstack-ansible, it could be added as an optional dependency within openstack-ansible. Documentation for the new role could be added into the existing openstack-ansible documentation to make it easier for openstack-ansible users to reference it.

The other work items are in the *Proposed change* section above in a numbered list. Each configuration change should come with documentation about the change.

11.25.4 Testing

The usual gate checks can be used for these changes. Also, each individual commit can be tested individually.

11.25.5 Documentation impact

Documentation is a critical piece of this spec, and its the first step in the process. It would be helpful to get the documentation team to weigh in on some of the documentation changes to ensure it makes sense for deployers.

11.25.6 References

Mailing list thread:

- http://lists.openstack.org/pipermail/openstack-dev/2015-September/074104.html

IRC discussion:

- http://bit.ly/1F1wBgB

DISA STIGs:

- https://en.wikipedia.org/wiki/Payment_Card_Industry_Data_Security_Standard

CHAPTER

TWELVE

LIBERTY SPECIFICATIONS

12.1 Compartmentalize RabbitMQ

date
2015-07-14

tags
rabbitmq

The purpose of this spec is to adjust our current RabbitMQ setup to better use the available system resources by creating a vhost and user per-consumer service within RabbitMQ.

Include the URL of your launchpad blueprint:

- https://blueprints.launchpad.net/openstack-ansible/+spec/compartmentalize-rabbitmq

12.1.1 Problem description

Presently all services use the single root virtual host within RabbitMQ and while this is OK for small to mid sized deployments however it would be better to divide services into logical resource groups within RabbitMQ which will bring with it additional security.

12.1.2 Proposed change

All services that utilize RabbitMQ should have their own virtual host, user, and password.

Overview:

- Each role would use the upstream Ansible RabbitMQ user module to create a new user. The username will be customizable with a default being the same as the name of the service.
- Each role will use the upstream Ansible RabbitMQ vhost module to create a new virtual host per service. The vhost will be customizable with a default being the same as the name of the service.
- A Password entry will be created within the `user_secrets.yml` file for each RabbitMQ service user.
- The oslo config section of each service will be updated to use the new vhost name, username, and password.

Alternatives

Leave RabbitMQ the way it is.

Playbook impact

The playbooks will have no impact. The changes being proposed are being done within roles. Ideally this would be a simple default addition, two new tasks, and a simple change within the oslo_messaging section in the service configuration files.

Upgrade impact

There will be an upgrade impact as the user will need to add the new secret entries to the `user_secrets.yml` file. If this was to be accepted as a backport to kilo this would have to be targeted to a major version.

Security impact

Serpentining the services into different vhosts with different users and passwords should improve security. And brings our project more inline with what is described in the OpenStack Messaging Security documentation.

- http://docs.openstack.org/security-guide/content/messaging-security.html

Performance impact

The separation of service into logical vhosts has been not been reported to have any noticeable performance impact.

- http://stackoverflow.com/questions/12518685/ performance-penalty-of-multiple-vhosts-in-rabbitmq
- http://lists.rabbitmq.com/pipermail/rabbitmq-discuss/2012-September/ 022618.html

End user impact

n/a

Deployer impact

The deployer will need to ensure they have passwords entries set within the `user_secrets.yml` file. This should not impact greenfield deployments however it will need to be something covered in an upgrade.

Developer impact

n/a

Dependencies

n/a

12.1.3 Implementation

Assignee(s)

Primary assignee:
https://launchpad.net/~kevin-carter `cloudnull`

Work items

- Add new RabbitMQ users for all services.
- Add new RabbitMQ vhosts for all services.
- Update all service configuration files to use the new vhost, user, and password.

12.1.4 Testing

The testing of this change is a convergence test. The gate job will utilize the the changes on every commit.

12.1.5 Documentation impact

Docs will need to be updated in terms of upgrades to add the new variables.

12.1.6 References

n/a

12.2 Allow os_* services to use a venv

date
2015-05-08 00:00

tags
python, venv, deployment

Enable the ability for a role to deploy OpenStack python code inside a venv

Blueprint:
https://blueprints.launchpad.net/openstack-ansible/+spec/enable-venv-support-within-the-roles

12.2.1 Problem description

There are two problems that we need to start anticipating:

- Some OpenStack services are running on physical hosts in the root namespace. This creates a situation where its possible for a service to have conflicting requirements with what is already on the host installed through the host package manager. In these situation weve found some instabilities that needed workarounds to ensure there are no stability or usage issues with the service.
- OpenStack services have started moving toward a non-integrated release which will allow projects to change their release cycle / cadence which will effect versions of services that we deploy. Additionally, these projects may choose to use dependencies outside of what is set in Global requirements.

The use of on metal services, the change in release cycles / cadence, and the likelihood of projects using requirements that conflict with one another requires more separation between the installed projects which lends itself to using a virtual environment for installed OpenStack Python code.

12.2.2 Proposed change

- Each os_* role will be modified to support a service running in a virtual environment. This will mean a few new variables to the defaults per-role to determine where the venv will live, change in pip package requirements as the virtualenv package will need to be installed first, changes to the init scripts to support a virtual environment, and a change to the sudoers file to allow the virtual environment bin path to be saved when executing a rootwrap command.
- The roles will support the option to deploy in a venv or not. This will be disabled `false` by default.
- The playbooks will have an option within them to enable or disable venv support at run time.
- Each venv will be named and tagged such that its unique as it pertains to the deployment. This will allow for package upgrade and downgrades on long lived deployements to take place without manual intervention or messing with the hosts packages which may have been installed as part of the base kick and using the operating system package manager.

Alternatives

Leave things unchanged or further pursue re-containerizing services that have been moved to the host. If we decide to go the route of re-containerizing projects that have been moved to the hosts namespace we will need to invest in kernel development to fix several issues we encountered which forced the move to running is_metal in the first place.

Playbook/Role impact

See *Proposed change*.

Upgrade impact

The use of venvs within an environment will not effect an existing deployment nor have any adverse effects on upgrades. Upgrading a service that hadnt used venvs in the past will be taken care of automatically as init scripts, sudoers files, and rootwrap configs will be changed to support the new venv install.

The benefit of running a service in a venv is apparent when dealing with downgrading a package requirement. This issue has been seen a few times where an upstream OpenStack project has downgraded a python package requirement in the middle of a release. In the current deployment system an administrator is required to manually intervene to resolve package downgrade issues. If the system was using a venv and was tagged based on a given deployment upgrading from one to release to another is as simple as re-running the role from the new released version. The result will be a new venv created for the service and the version. This has an upgrade side effect that will allow for Kilo to Liberty upgrades without having to deal with a epoch wheel build or munging of the wheels repo further simplifying an upgrade in terms of what will be required by the end user.

Security impact

N/A.

Performance impact

While not directly related to the implementation of this spec it would be possible for us to extend the virtualenv implementation to allow for building and redistribution of pre-built virtualenvs as a means of speeding up and maintaining reproducibility within an environment.

End user impact

N/A.

Deployer impact

When working within a container access to the service management utilities (nova-manage, cinder-manager, etc) the deployer or administrator working on an environment will need to sourced/activated the virtualenv before running the tools. While this is an extra step there are no other changes that will need to be addressed in the typical deployer workflow.

Developer impact

N/A.

Dependencies

N/A.

12.2.3 Implementation

Assignee(s)

Primary assignee:
<cloudnull> - https://launchpad.net/~kevin-carter

Secondary assignee:
Anyone who wants to help

Work items

- Update all roles to support venvs
- Add a variable to the OSA playbooks to enable venv support within the roles.

12.2.4 Testing

- Testing this will rely on the gate as a convergence test.
- This is implemented in Liberty we can create a simple periodic job in OpenStack infra to test upgrades. The upgrade testing will report back to the OpenStack QA mailing list and key of their periodic job queue.

12.2.5 Documentation impact

- Documentation will need to be written to acknowledge the venv based deployment and how deployers are to interact with the management tools as provided by the service.

12.2.6 References

Related Bug:

- https://bugs.launchpad.net/openstack-ansible/+bug/1488315

12.3 Liberty Release

date
2015-09-08 08:30

tags
liberty, update

Update the various playbooks and roles in the master branch with the changes necessary to implement a fully functional and updated Liberty deployment.

- https://blueprints.launchpad.net/openstack-ansible/+spec/liberty-release

Initial work will be based on the Liberty RC tags in each of the OpenStack projects since Liberty is not yet officially released.

12.3.1 Problem description

While the master branch has been tracking Liberty code for some time, none of the configuration files have been updated to match the upstream changes in order to handle deprecation, different default values, etc.

12.3.2 Proposed change

1. Each template/file carried in-tree will need to be reviewed and revised according to the new defaults and other adjustments in each OpenStack project.
2. Each service will need to be inspected for changes in how deployments and upgrades are handled, and the role tasks adjusted accordingly.
3. Any other changes to each service will need to be inspected and adjustments to the roles must be made accordingly.
4. In a final test, fatal_deprecations should be set to True for all the services to validate that all deprecated configurations have been removed or replaced.

The approach to dealing with differences (eg changed defaults for a particular setting) will be to use the Liberty value where possible. Deployers who are upgrading from Kilo may use the config_overrides to implement overrides for any configurations that they wish to keep at the previous values.

Examples of configs impacted (these will differ depending on the service being worked on):

```
/etc/<servicename>/<servicename>.conf
/etc/<servicename>/<servicename>-api-paste.ini
/etc/<servicename>/policy.json
/etc/<servicename>/<servicename>-<agentname>.ini
```

Alternatives

We could, wherever needed, preserve Kilo settings rather than taking forward the Liberty settings. This is potentially easier on users in an upgrade scenario, but does mean that new users deploying Liberty would get an already out of date deployment. It also means that we miss an opportunity to implement best practices deployments, instead sticking on old, less relevant, values.

Playbook impact

There will be no impact on the playbooks. These changes are on the dependency and role level which only impact the configuration files and role options.

Upgrade impact

This change will impact upgrades, but upgrades are specifically out of scope and will be addressed separately in https://blueprints.launchpad.net/openstack-ansible/+spec/liberty-upgrade-path

The focus for this spec will be for new deployments only.

Security impact

Security testing and improvements are specifically out of scope. Testing for security changes and improvements can be done after the release and implemented in subsequent patches.

Performance impact

Performance testing and improvements are specifically out of scope. Testing for performance changes and improvements can be done after the release and implemented in subsequent patches.

End user impact

N/A

Deployer impact

Impacts must be noted in the commit messages for each change.

Developer impact

This change is to allow development of a production grade Liberty deployment

Dependencies

1. There are several feature blueprints which are expected to merge into the master branch alongside these changes. These features are to facilitate future improvements in the Mitaka development timeframe and may be backported to Liberty. The feature blueprints are marked as dependent in Launchpad.
2. The final release of OpenStack-Ansibles Liberty release is entirely dependent on OpenStacks Liberty release.

12.3.3 Implementation

Assignee(s)

- Ceilometer: https://launchpad.net/~miguel-cantu `alextricity`
- Cinder: https://launchpad.net/~jesse-pretorius `odyssey4me`
- Glance: https://launchpad.net/~jesse-pretorius `odyssey4me`
- Heat: https://launchpad.net/~jesse-pretorius `odyssey4me`
- Horizon: https://launchpad.net/~steve-lewis `stevelle`
- Keystone: https://launchpad.net/~jesse-pretorius `odyssey4me`
- Neutron: https://launchpad.net/~jesse-pretorius `odyssey4me`
- Nova: https://launchpad.net/~jesse-pretorius `odyssey4me`
- Swift: https://launchpad.net/~jesse-pretorius `odyssey4me`
- Tempest: https://launchpad.net/~jesse-pretorius `odyssey4me`

Work items

See Assignees.

12.3.4 Testing

No changes to the current testing and or gating framework will be made. Each change that is made to a service to bring forward new configs and settings will be required to pass the same gate tests as are required by our production systems.

12.3.5 Documentation impact

All changes made will require DocImpact tags in the commit messages in order to track the changes required for documentation.

12.3.6 References

- https://etherpad.openstack.org/p/liberty-config-changes

12.4 Modularizing Neutron plays for agents and non ml2 plugin support

date
2015-09-09 18:00

tags
neutron, plugins, agents

This spec is propsed to enhance the current neutron playbooks that take a static approach to plugin and agent insertion. Where ml2 and a few agents are used by default.

- https://blueprints.launchpad.net/openstack-ansible/+spec/modularize-neutron-liberty

12.4.1 Problem Description

Presently a straightforward approach does not exist to add new plugins and add / remove agents to the neutron setup. A deployer either has to perform these changes after the whole setup is complete or make their own changes in the playbooks locally. This feature has already been implemented in juno branch.

12.4.2 Proposed Change

The files in playbooks/roles/os_neutron/tasks will be modified, particularly neutron_pre_install.yml and neutron_post_install.yml. Addition of new parameters will be made to playbooks/roles/os_neutron/defaults/main.yml

Playbook Impact

The following playbooks are expected to be modified to support this feature:

- playbooks/roles/os_neutron/defaults/main.yml
- playbooks/roles/os_neutron/tasks/main.yml
- playbooks/roles/os_neutron/tasks/neutron_pre_install.yml
- playbooks/roles/os_neutron/tasks/neutron_post_install.yml

Upgrade Impact

None

Alternatives

Using the current architecture, prospective new core plugins which are not ml2 will have to take an overwriting the default configuration, after its done, approach to insert their own changes.

Security Impact

None known at this time.

Performance Impact

This change is not expected to impact performance. Installing the default set of agents and plugins as done now, will take the same amount of effort.

End User Impact

This is not expected to impact end users as it deals with the deployment aspect only.

Deployer Impact

This will introduce a more modular architecture for deployers to select neutron plugins/agents from, allowing a wider use case for the OSAD playbooks.

Developer Impact

Using the default values will require no new developer effort, only those interested in changing the neutron config will be effected.

Dependencies

N/A

12.4.3 Implementation

Assignee(s)

Primary assignee:
https://launchpad.net/~javeria-ak `javeriak`

Work items

This change will include modifying the existing os_neutron role to pick up what plugin to install along with what agents. The names and configs for individual plugins will be created as new variables in playbooks/roles/os_neutron/defaults/main.yml

Dependencies

N/A

12.4.4 Testing

There are no additional changes required to test this in the current testing and or gating framework.

12.4.5 Documentation Impact

A bit of additional documentation describing how to insert new plugins/agents will be required. This will be deployer documentation.

12.4.6 References

- https://blueprints.launchpad.net/openstack-ansible/+spec/modularize-neutron-plays

12.5 Named veths

date
2015-08-31 22:00

tags
lxc, veth, troubleshooting

This spec aims to make troubleshooting openstack-ansible issues a more efficient process by using container names to build names for veth interfaces.

Link to blueprint:

- https://blueprints.launchpad.net/openstack-ansible/+spec/named-veths

12.5.1 Problem description

All veth interfaces on the host are named using randomly generated names, such as *vethK070G4*. This can make troubleshooting container networking issues more challenging since its difficult to trace a veth name to a particular network interface within the container.

12.5.2 Proposed change

Names of veth interfaces should be unique and easily correlated to their containers. However, names of network interfaces have restrictions which must be handled carefully:

- 16 characters maximum
- Certain characters, like dashes (-) arent allowed

The random characters on the end of the container hostname could be used along with the interface name to form a veth name. As an example, a container called *aio1_utility_container-a9ef9551* could have two named veth interfaces:

- a9ef9551_eth0
- a9ef9551_eth1

Alternatives

Leave veth names as randomly generated by LXC.

Playbook/Role impact

The veth names will only be adjusted on the host within the LXC configuration files themselves. Containers wont be affected. The playbooks dont use the veth names on the host for any actions.

If veths are not cleaned up properly when a container stops (this is sometimes called dangling veths), theres a chance that the container wont start until the dangling veth is manually removed with *ip link del <veth>*.

Upgrade impact

Upgrades should be unaffected. This change only adjusts the LXC container configuration files and doesnt change the running configuration of any of the containers.

If a container is running and its LXC configuration file is adjusted to use named veths, it will only utilize those adjustments when it is restarted. If an upgrade happens to restart only a subset of the containers on the host, then only those containers will use named veths after they restart.

Security impact

This change shouldnt affect security.

Performance impact

This change shouldnt affect performance.

End user impact

This change shouldnt affect end users.

Deployer impact

Users who deploy OpenStack should be able to troubleshoot network issues more efficiently.

For example, if a user was having trouble reaching the nova API container, they could quickly see which veths were associated with the container. This would allow users to diagnose network problems with various tools, like ethtool and tcpdump, without digging into interface indexes or writing scripts.

If a deployer wants to begin using named veth pairs immediately, then all containers must be restarted. This is because the LXC configuration files are adjusted on disk but running containers arent adjusted.

Developer impact

Much like the deployer impact above, this change could help developers diagnose issues within different containers more efficiently.

Dependencies

This spec has no known dependencies.

12.5.3 Implementation

Assignee(s)

Primary assignee:
https://launchpad.net/~rackerhacker `mhayden`

Work items

- Update ansible playbooks to specify *lxc.network.veth.pair* in the main LXC configuration files as well as the interface .ini files

12.5.4 Testing

- Do greenfield deployment and verify named veths
- Do an upgrade between releases and verify named veths
- Verify that both tests have no impact on running containers

12.5.5 Documentation impact

Documentation would be beneficial, especially around how this helps with troubleshooting issues.

12.5.6 References

N/A

12.6 Add PLUMgrid plugin to neutron playbooks

date
2015-09-09 19:30

tags
neutron, plugins, networking

This spec is propsed to insert the capability of using the PLUMgrid OpenStack Neutron Plugin through the OSAD neutron playbooks.

- https://blueprints.launchpad.net/openstack-ansible/+spec/plumgrid-support-liberty

12.6.1 Problem Description

PLUMgrid is a core neutron networking plugin that has been a part of OpenStack neutron since Grizzly. It offers a Network Virtualization Platform that uses direct communication with the Hypervisor layer to provide all the networking functionality requested through Neutron APIs. The PLUMgrid Neutron Plugin implements Neutron v2 APIs and helps configure L2/L3 virtual networks created through the PLUMgrid Platform. It also implements External Networks and Port Binding Extensions.

APIs supported by the PLUMgrid plugin:

- Networks
- Subnets
- Ports
- External Networks
- Routers
- Security Groups
- Quotas
- Port Binding

- Provider Networks

12.6.2 Proposed Change

This change is proposed to add the PLUMgrid plugin as a core plugin option alongside ml2, which will be the default. This configurability should already be achieved by the BP: modularize-neutron-liberty.

The rest of the installation for PLUMgrid that requires PLUMgrid Controller and Compute components, that enable management of the plugin, is maintained in a public plumgrid-ansible repository.

The changes described below assume the previously mentioned BP modularization changes in place.

This feature is proposed for the master branch leading to liberty. Once implemented it will be backported to kilo.

The parameters relevant to the PLUMgrid plugin installation will be added to a new dictionary item in neutron_plugins in playbooks/roles/os_neutron/defaults/main.yml. This will allow setting the neutron_plugin_type to plumgrid if desired.

Playbook Impact

These files are expected to be modified:

- playbooks/roles/os_neutron/defaults/main.yml

New templates will be added in the os_neutron role:

- playbooks/roles/os_neutron/templates/plugins/plumgrid/plumgrid.ini
- playbooks/roles/os_neutron/templates/plugins/plumgrid/plumlib.ini
- playbooks/roles/os_neutron/files/rootwrap.d/plumlib.filters

Upgrade impact

None

Alternatives

To continue using the default ml2 and linuxbridge-agent neutron deployment with no possibility of other core neutron plugins.

Security Impact

N/A

Performance Impact

This change is not expected to impact performance. A typical PLUMgrid plugin installation, will furthermore disable neutron agent installations. Hence the overall performance is expected to remain the same.

End User Impact

End users will be able to leverage the enhanced scale and operational capabilities provided by the PLUMgrid plugin when choosing to install this plugin. Further details can be found in the References section below.

Deployer Impact

This will provide Deployers with the option to use PLUMgrid as the neutron plugin. Upgrading from a previous release to use this new feature will only be possible through a re-run of the neutron playbooks as well. This change does not effect running instances within the cloud.

Developer Impact

This change adds further installable options and as such does not effect the default flow of the playbooks.

Dependencies

None

12.6.3 Implementation

Assignee(s)

Primary assignee:
https://launchpad.net/~javeria-ak `javeriak`

Work items

This change will use the modularized neutron playbooks to provide PLUMgrid as a plugin option. A set of three new template files will be added to the neutron plays to support plumgrid.

Dependencies

Dependent on:

- https://blueprints.launchpad.net/openstack-ansible/+spec/modularize-neutron-liberty

12.6.4 Testing

There are no additional changes required to test this in the current testing and or gating framework that also covers the neutron components.

12.6.5 Documentation Impact

Documentation describing how to modify the configuration parameters to install PLUMgrid will be required. This will be deployer documentation.

12.6.6 References

- https://www.vmware.com/products/nsx.html
- https://wiki.openstack.org/wiki/PLUMgrid-Neutron
- https://github.com/plumgrid/plumgrid-ansible

12.7 Remove upstream repo dependency

date
: 2015-07-19

tags
: repo, repo-servers, repo-clone, pip-wheel

The purpose of this spec is to remove the repo-clone play from OSAD.

- https://blueprints.launchpad.net/openstack-ansible/+spec/Remove-upstream-repo-dependency

12.7.1 Problem description

Presently the repo-clone-mirror play is responsible for cloning the upstream repository that Rackspace maintains into the repo containers. While this process is simple enough it does bring with it a reliance on an upstream deployer/vendor. OSAD already has the ability to build its own python packages which is the process used to do all gate check testing so it should also be the default means to deploy an OSAD environment.

12.7.2 Proposed change

- Remove the repo-clone-mirror.yml play
- Change repo-install.yml to use repo-build.yml as its included method.
- Modify the pip install role to remove the install requirement using the upstream mirror.

Alternatives

Leave everything the way it is.

Playbook impact

Changes the repo create process to always build. This will only impact deployers that are using the repo-servers and will ensure that the system is always building the correct packages.

When bootstrapping a new environment the pip install role is used throughout the stack. This would modify that role to always pull upstream pip unless otherwise instructed, through the use of *user_vars*, to go elsewhere.

Upgrade impact

n/a

Security impact

n/a

Performance impact

Repo clone was intended to be a faster means of delivering packages to the deployment infrastructure however in testing repo clone and repo build operate at roughly the same speed.

End user impact

n/a

Deployer impact

This change will be unnoticeable to the deployer.

Developer impact

n/a

Dependencies

n/a

12.7.3 Implementation

Assignee(s)

Primary assignee:
https://launchpad.net/~kevin-carter `cloudnull`

Work items

- Delete the repo-clone-mirror.yml play
- Change the include in repo-install.yml s/repo-clone-mirror.yml/repo-build.yml/

12.7.4 Testing

This is already being tested on every build within upstream OSAD.

12.7.5 Documentation impact

n/a

12.7.6 References

n/a

12.8 HAProxy improvements

date
2015-09-04 14:00

tags
haproxy, production use

HA Proxy can be improved by adding a few changes:

- Making it really HA
- Allowing configuration interface to easily adapt load
- Deploying only the configuration for the services deployed within the inventory.

- Improving backends configuration, for example galera or adapting the timer values to be more efficient

https://blueprints.launchpad.net/openstack-ansible/+spec/role-haproxy-v2

12.8.1 Problem description

There are a few features already asked by the community:

- HA for haproxy
- Enable statistics and improve manageability of haproxy
- Limiting the unnecessary checks of haproxy

12.8.2 Proposed change

- Implement keepalived for haproxy
- Change the standard haproxy role to add administrative tools (admin level on socket and stats)
- Remove the large haproxy variable in vars/ folder
- Give this information component by component (in the group_vars), and make it possible to have user overrides (user_variables or component by component). Then `delegate` the configuration to haproxy hosts.
- Introduce a skip variable, if you want to deploy haproxy on some components but not some others

Alternatives

Wait for ansible2 to have variable merging/cleanup for dicts on a per task/playbook basis.

Playbook/Role impact

The playbook `haproxy-install.yml` will be completely overwritten.

haproxy playbook run will be longer, due to the `delegate to`.

Upgrade impact

None.

Security impact

No change

Performance impact

Improved performance by:

- Doing less unnecessary checks to backends
- Adding an easy way to set customer values for the backends timers.

End user impact

No change

Deployer impact

- No change in default configuration
- The deployer can overwrite the `haproxy_service_configs` per component

Developer impact

No impact at first sight.

Dependencies

None

12.8.3 Implementation

Assignee(s)

None

Work items

- Keepalived: https://review.openstack.org/#/c/217517/
- Easy administration: https://review.openstack.org/#/c/215019/ and https://review.openstack.org/#/c/214110/
- Default configuration less static:
  - rewrite haproxy-install with the delegate_to and with a when haproxy_component_skip (if you want to deploy haproxy on some components but not some others)
  - create a file per component with default variables under group_vars
- Default timer value changes.

12.8.4 Testing

- Does this change impact how gating is done?

There will be a change to haproxy-install playbook if merged.

- Can this change be tested on a **per-commit** basis?

Yes

- Given the instance size restrictions, as found in OpenStack Infra (8GB Ram, vCPUs <= 8), can the test be run in a resource constrained environment?

No change

- Is this untestable given current limitations (specific hardware / software configurations available)? If so, are there mitigation plans for this change to be tested within 3rd party testing, gate enhancements, etc?

No

- If the service is not OpenStack specific how can we test the change?

Running the new playbooks

12.8.5 Documentation impact

For those who change the default configuration of haproxy (currently not documented), this change would modify their current configuration, so it needs to be documented. Explanation of the skip variable and component by component override should be good to add in the doc too.

12.8.6 References

None

12.9 Tunable OpenStack Configuration

date
2015-08-26

tags
openstack, configuration, tuning

Instead of implementing a specific variable for each possible/desired configuration entry in every role, there is a more general way that this could be done which would enable the deployer to implement any desired valid configuration entry for a given role. OpenStack Ansible will then only need to implement deviations from the standard OpenStack default settings. Examples of these settings would be minimum settings to make the service work and best practice settings. Changes to templates would allow the project the ability to ship flat, or otherwise minimally dynamic, configuration files in an effort to limit the sheer size and scope of the number of variables we have if a given role.

- https://blueprints.launchpad.net/openstack-ansible/+spec/ tunable-openstack-configuration

This implementation is intended to:

- Reduce the cruft of variables the project needs to carry just to enable customizations requested from deployers.
- Reduce the amount of configuration file settings the project needs to work through for every major upgrade of the underlying OpenStack environment.
- Decrease the turnaround time for deployers to implement additional configuration items in OpenStack configuration files.

12.9.1 Problem description

The OpenStack Ansible project currently carries a lot of variables in roles which are simply there to enable the ability to override OpenStack default settings. This is unnecessary cruft which has to be reviewed for deprecated options whenever the project starts working with an updated version of OpenStack.

It also introduces an unnecessary propose/develop/test/release cycle for simple changes to configuration files which could easily be done by a deployer if theyre enabled to do so.

12.9.2 Proposed change

- A new Ansible *action plugin* will be created which will facilitate the ability for templates to be updated dynamically. This change will build off of the existing Ansible template functionality and but allow for changes to be applied through a simple dictionary data structure. Updates are applied to the template while in transit allowing us to carry minimal code while leveraging all of the already built in Ansible functionality.
- The action plugin named config_template will add two new input types *config_data* and *config_type*. The new input options will be optional allowing us to simply replace our current template tasks with the *config_template* module.
- New defaults will be created as empty dictionaries as base entry points for deployers to override items in config.

Code wise the change to a templed task will look something this:

```
- name: run config template ini
  config_template:
    src: templates/test.ini.j2
    dest: /tmp/test.ini
    config_data: {'data': 'things'}
    config_type: ini

- name: run config template json
  config_template:
    src: templates/test.json.j2
    dest: /tmp/test.json
    config_data: {'data': 'things'}
    config_type: json

- name: run config template yaml
  config_template:
    src: templates/test.yaml.j2
```

(continues on next page)

(continued from previous page)

```
    dest: /tmp/test.yaml
    config_data: {'data': 'things'}
    config_type: yaml
```

To note:

A dictionary used to update or override items within a configuration template. The dictionary data structure may be nested. If the target config file is an ini file the nested keys in the `config_data` will be used as section headers.

Alternatives

Continue to use the current paradigm of adding an ansible variable per configuration configurable file entry.

Playbook impact

The existing roles would need to be adjusted to support the new config_data entry point and to have the relevant template tasks updated to the config_template module.

Upgrade impact

This change will not impact current upgrades as everything in all of the templates can remain the same. The proposed module changes simply make it possible for deployers to update templates as needed without having to make changes in tree. In future releases we can deprecate variables were presently carrying as needed or wanted while still maintaining the ability to override options as needed through the use of the config_template module.

Security impact

None.

Performance impact

None.

End user impact

None

Deployer impact

Deployers will be able to dynamically update configuration options based on their need without making changes in tree. This will allow deployers a greater ability to tailor deployments as needed.

Developer impact

This would reduce the need for developers to get involved with small patches that implement basic configuration file entries which deployers wish to use.

Dependencies

None.

12.9.3 Implementation

Assignee(s)

Primary assignee:
https://launchpad.net/~kevin-carter IRC: cloudnull

Other contributors:
None at this time.

Work items

Implement tunable configurations for all configuration files that fall under the following formats: [yaml, json, ini]

- Develop Ansible Action Plugin to enable the ability to make in flight config changes to an existing template.
- Change all template tasks within the roles that drop configuration files to use the new config_template module.
- Replace the *copy_update* module with the *config_template* module.

12.9.4 Testing

In the current gate testing we can add a basic template test to override a few options / add a few options and assert that the changes from the base template took place. This can be accomplished using items from the example tasks and a simple json, ini, and yaml data structure. We could also set overrides with the gate that we know we want to run within our deployments such that were exercising the OpenStack code paths that were attempting to enable via the gate. In this way we might be able to cut out some of our gate script variables as well.

12.9.5 Documentation impact

While the Action Plugin has documentation within it, per the normal Ansible module documentation process, we can also update our general install documentation to reference the existence of the new module and how it works. Id like to refrain from documenting every override entry point as the authoritative source for those types of items will be the role defaults themselves.

12.9.6 References

None

12.10 MariaDB upgrade to v10

date
2015-07-19

tags
mysql, galera

The purpose of this spec is to upgrade MariaDB from v5.5 to v10.0

https://blueprints.launchpad.net/openstack-ansible/+spec/MariaDB-upgrade-to-v10

12.10.1 Problem description

MariaDB + Galera is presently using v5.5 which is old and should be upgraded. Additionally, we are using xtrabackup v1 which was deprecated in favor of xtrabackup v2 as such that should be changed as we upgrade to v10 so that we can take advantage of the performance and security enhancement available in the new releases.

12.10.2 Proposed change

- Upgrade MariaDB - this is a package change as well as upstream mariadb repo change
- Change xtrabackup to xtrabackup-v2 - This will add a configuration section in the default `my.cnf` for the xtrabackup client(s).

Alternatives

Leave everything the way it is.

Playbook/Role impact

There will be no playbook impact however the The galera_server and galera_client roles will change to support the new packages for xtrabackup-v2 and mariadb+galera v10.

Upgrade impact

n/a

Security impact

Upgrading to MariaDB v10 w/ xtrabackup v2 will result OSAD being able to take advantage of better security options in the future if we so choose.

Performance impact

Upgrading to MariaDB v10 w/ xtrabackup v2 will result in greater performance.

End user impact

n/a

Deployer impact

The deployer will need to be aware that mariadb v5.5 is being upgraded however all of the post upgrade processes should be handled automatically.

Developer impact

n/a

Dependencies

- SPEC/Limit the distribution of .my.cnf - https://review.openstack.org/#/c/203754/

12.10.3 Implementation

Assignee(s)

Primary assignee: (unassigned)

Work items

- Change the package for MariaDB10 w/ Galera
- Add repo for new versions of XtraBackup
- Update the my.cnf for use with MariaDB10 (revise it for anything that may need to be removed)
- Update the cluster.cnf for use with MariaDB10 (revise it for anything that may need to be removed)

12.10.4 Testing

The testing for this change will be automatic in upstream as everybuild will change to using this by default.

12.10.5 Documentation impact

n/a

12.10.6 References

n/a

CHAPTER

THIRTEEN

KILO SPECIFICATIONS

13.1 Add PLUMgrid plugin to neutron playbooks

date
2015-06-2 14:30

tags
neutron, plugins, networking

This spec is propsed to insert the capability of using the PLUMgrid OpenStack Neutron Plugin through the os-ansible neutron playbooks.

- https://blueprints.launchpad.net/openstack-ansible/+spec/add-plumgrid-neutron-plugin

13.1.1 Problem Description

PLUMgrid is a core neutron networking plugin that has been a part of OpenStack neutron since Grizzly. It offers a Network Virtualization Platform that uses direct communication with the Hypervisor layer to provide all the networking functionality requested through Neutron APIs. The PLUMgrid Neutron Plugin implements Neutron v2 APIs and helps configure L2/L3 virtual networks created through the PLUMgrid Platform. It also implements External Networks and Port Binding Extensions.

APIs supported by the PLUMgrid plugin:

- Networks
- Subnets
- Ports
- External Networks
- Routers
- Security Groups
- Quotas
- Port Binding
- Provider Networks

13.1.2 Proposed Change

This change is proposed to add the PLUMgrid plugin as a core plugin option alongside ml2, which will be the default. This configurability should already be achieved by the BP: modularize-neutron-plays.

The rest of the installation for PLUMgrid that requires PLUMgrid Controller and Compute components, that enable management of the plugin, will be added externally through Ansible Galaxy roles.

The changes described below assume the previously mentioned BP modularization changes in place.

This feature is proposed for both kilo and juno branches, the juno change will be carried out first:

1. For juno, parameters relevant to the PLUMgrid plugin, namely the plumgrid core plugin and plugin config file, plumgrid.ini will be added to a new dictionary item in neutron_plugins in inventory/group_vars/neutron_all.yml This will allow setting the neutron_plugin_type = plumgrid if desired.
2. For kilo, parameters relevant to the PLUMgrid plugin will be added to a new dictionary item in neutron_plugins in playbooks/roles/os_neutron/defaults/main.yml. This will allow setting the neutron_plugin_type to plumgrid if desired.

Playbook Impact

1. In juno, the following files are expected to be modified:

- rpc_deployment/playbooks/openstack/inventory/group_vars/neutron_all.yml

The following templates will be created in neutron-common role:

- rpc_deployment/roles/neutron_common/templates/plugins/plumgrid/plumgrid.ini
- rpc_deployment/roles/neutron_common/templates/plugins/plumgrid/plumlib.ini
- rpc_deployment/roles/neutron_common/templates/rootwrap.d/plumlib.filters

2. In kilo, these files are expected to be modified:

- playbooks/roles/os_neutron/defaults/main.yml

New templates will be added in the os_neutron role:

- playbooks/roles/os_neutron/templates/plugins/plumgrid/plumgrid.ini
- playbooks/roles/os_neutron/templates/plugins/plumgrid/plumlib.ini
- playbooks/roles/os_neutron/files/rootwrap.d/plumlib.filters

Upgrade impact

None

Alternatives

To continue using the default ml2 and linuxbridge-agent neutron deployment with no possibility of other core neutron plugins.

Security Impact

N/A

Performance Impact

This change is not expected to impact performance. A typical PLUMgrid plugin installation, will furthermore disable neutron agent installations. Hence the overall performance is expected to remain the same.

End User Impact

End users will be able to leverage the enhanced scale and operational capabilities provided by the PLUMgrid plugin when choosing to install this plugin. Further details can be found in the References section below.

Deployer Impact

This will provide Deployers with the option to use PLUMgrid as the neutron plugin. Upgrading from a previous release to use this new feature will only be possible through a re-run of the neutron playbooks as well. This change does not effect running instances within the cloud.

Developer Impact

This change adds further installable options and as such does not effect the default flow of the playbooks.

Dependencies

None

13.1.3 Implementation

Assignee(s)

Primary assignee:
https://launchpad.net/~javeria-ak

Work items

This change will use the modularized neutron playbooks to provide PLUMgrid as a plugin option. A set of three new template files will be added to the neutron plays to support plumgrid.

Dependencies

Dependent on:

- https://review.openstack.org/184665
- https://blueprints.launchpad.net/openstack-ansible/+spec/modularize-neutron-plays

13.1.4 Testing

There are no additional changes required to test this in the current testing and or gating framework that also covers the neutron components.

13.1.5 Documentation Impact

Documentation describing how to modify the configuration parameters to install PLUMgrid will be required. This will be deployer documentation.

13.1.6 References

- http://www.plumgrid.com/
- https://wiki.openstack.org/wiki/PLUMgrid-Neutron

13.2 Ceph Block Devices

date
: 2015-07-23 12:00

tags
: storage, ceph

This spec is a proposal to add the ability to configure cinder, glance, and nova running in an openstack-ansible installation to use an existing Ceph cluster for the creation of volumes, images, and instances using Ceph block devices.

- https://blueprints.launchpad.net/openstack-ansible/+spec/ceph-block-devices

13.2.1 Problem description

This implementation should meet the following user requirements:

- Cinder Volume Creation: As a User I want to be able to allocate block storage volumes from the Ceph Storage Cluster to individual virtual machines.
- Cinder Boot from Volume: As a User I want to be able to create a virtual machine that boots from a block storage device hosted on the Ceph Storage Cluster.
- Cinder Snapshots: As a User I want to be able to create a snapshot of one or more Cinder Volumes.
- Cinder Backups: As a User I want to be able to use the Ceph Storage Cluster as a target for cinder backups.
- Glance Storage: As a User I want to be able to use the Ceph Storage Cluster as a backend for glance.
- Nova Ephemeral Storage: As a User I want to be able to allocate nova instance storage from the Ceph Storage Cluster.
- Live Migration Support: As an Admin I want to be able to live migrate virtual machines that depend upon (i.e. boots from/mounts) a block storage device hosted on the Ceph Storage Cluster. This is inclusive of both Boot from Volume and Nova ephemeral storage.

13.2.2 Proposed change

1. Create ceph_client role to handle installation of ceph packages and and configuration of ceph.conf file.
2. Update os_cinder role to conditionally allow cinder-volume to create volumes in Ceph by setting volume_driver to cinder.volume.drivers.rbd.RBDDriver.
3. Update os_nova role to conditionally allow nova to boot from cinder volumes stored in Ceph.
4. Update os_cinder role to conditionally allow cinder-backup to store backups in Ceph by setting backup_driver to cinder.backup.drivers.ceph.
5. Update os_glance role to conditionally allow glance to store images in Ceph by setting default_store to rbd.
6. Update os_nova role to conditionally allow nova to boot virtual machines directly into Ceph by setting libvirt_images_type to rbd.

Alternatives

None

Playbook impact

See Proposed change above.

Upgrade impact

No default configurations should be altered with the introduction of these changes, and therefore there should be no impact to the upgrade of an existing installation.

Security impact

OpenStack services require users and keys to interface with Ceph. This implementation should encourage the use of separate Ceph users for each OpenStack service and ensure that configuration files and keys can only be read by the intended users.

Performance impact

Enabling this functionality may result in performance increases or decreases across the OpenStack installation. This will highly depend on the hardware and software configuration of the attached Ceph cluster.

End user impact

Using Ceph block devices may introduce new features visible to the end user, such as the ability to live migrate an instance from one hypervisor to another. Additionally, as stated above, there may be visible performance increases or decreases depending on several different facters.

Deployer impact

A deployer will need to explicitly update their inventory and set Ansible variable overrides to a) enable this functionality and b) correctly interface with an existing Ceph cluster.

Developer impact

None

Dependencies

None known

13.2.3 Implementation

Assignee(s)

Primary assignee:
https://launchpad.net/~mattt416 (mattt)

Other contributors:
https://launchpad.net/~git-harry (git-harry) https://launchpad.net/~david-wilde-rackspace (d34dh0r53)

Work items

1. Create ceph_client role.
2. Update os_cinder role to conditionally allow cinder-volume to create volumes in Ceph.
3. Update os_nova role to conditionally allow nova to attach cinder volumes stored in Ceph.
4. Update os_cinder role to conditionally allow cinder-backup to store backups in Ceph.
5. Update os_glance role to conditionally allow glance to store images in Ceph.
6. Update os_nova role to conditionally allow nova to boot virtual machines directly into Ceph.

13.2.4 Testing

No gate-related adjustments will be made to openstack-ansible to support this change as no default configurations are being changed here. Additionally, that there are strict limitations on what can run in the all-in-one (AIO) gate instance.

13.2.5 Documentation impact

Documentation will need updating to include:

1. How to enable Ceph block devices for each cinder, glance, and nova services and what each newly introduced Ansible variable does.
2. What additional steps are required to be executed on the existing Ceph cluster to allow the OpenStack installation to interface with the Ceph cluster.

13.2.6 References

None

13.3 openstack-ansible overview documentation

date
2015-04-13 13:00

tags
documentation, docs, developer

Currently, the openstack-ansible repository does not have any cohesive developer documentation. This proposal aims to begin such documentation, providing a overview and reference for new contributors.

Documentation covered in this proposal is *not* intended to be exhaustive.

Also, documentation is a constantly shifting thing - this proposal is just intended to get the initial documents created; documentation maintenance falls outside of scope. Reviewers should help make sure patches adequately update associated documentation.

A blueprint for this proposal can be found at:

https://blueprints.launchpad.net/openstack-ansible/+spec/developer-docs

13.3.1 Problem description

Currently, when new contributors come to the repository, there isnt much to help them to understand the general structure of the project. Currently, there is the development-stack.rst file, which provides a very brief introduction to getting an all-in-one (AIO) install started, as well as tearing down an environment, but theres not much else. This can be intimidating for newcomers, as well as current contributors who might forget details of some portion of the large code base.

13.3.2 Proposed change

This proposal recommends making a new docs repo, which would contain Sphinx documentation on the following documentation:

- Overview of doing deployments using openstack-ansible
- Variable files, for knowing which variable files are used where in the process.
- Scripts. This section will cover using bootstrap, gating, and teardown scripts. It might also document some of the important variables/parameters for these scripts. The openstack-ansible wrapper would be nice to cover here and in the extending section.
- Playbooks should document the high-level playbooks that prepare physical hosts, create containers, and install OpenStack.
- Repository role/playbook. Since the repository is fairly unique to openstack-ansible, this should be probably be a bit more detailed than the rest of the of the documentation. The openstack_services.yml and openstack_other.yml files are of particular interest here.
- Inventory management. This section should discuss the dynamic_inventory.py file and the inventory_management.py files.
- Extending openstack-ansible. This would cover using the conf.d and env.d directories, as well as user_extras.yml files. Changes to ansible.cfg necessary might be useful, too.

Also, the docs directory should be built by Sphinx on a regular basis, preferably at the gate.

Some topics are explicitly out of scope for this changes. In particular:

- Host networking setup. This is highly individualized per environment, and too broad to cover here.
- Individual roles. The individual roles should not be documented as part of this. There are too many roles and too many changes to be able to keep up with those changes at the openstack-ansible level.
- End user documentation. For this specification, the end user is a deployer or user of the deployed OpenStack system. The installation guide, operations guide, and user/admin guides are all out of scope.

Alternatives

We could not do this documentation, leaving the repository as is.

Playbook impact

There should be no impact on the playbooks; this change only adds files outside of the playbooks.

Upgrade impact

This documentation will have to be kept up-to-date with releases. Since documentation is an on-going process, it falls to reviewers to enforce documentation updates to playbook changes.

Security impact

Since this change is not to the playbooks or scripts, it should have no security impact.

Performance impact

This will not have a production performance impact. I do propose adding a docs build job to the gating process, which would extend gating job times by some unknown amount.

End user impact

Users of a deployed cloud would likely never see this change. Its largely targeted at developers of this project or deployers.

Deployer impact

There should be little to no deployer impact. This documentation will be for developers, mostly, but deployers may be able to use it as reference for running scripts and tools on their deployment.

Developer impact

This documentation will be targeted mostly at developers in the hope that it will be easier for new contributors to understand how the project works and where to start. This can also be useful as a reference for existing developers.

The Sphinx build process may add some overhead, since developers should build the documentation before pushing their changes.

Where this proposal differs from the CONTRIBUTING.txt is the focus - CONTRIBUTING.txt is largely about the process around getting changes into the codebase. In contrast, the docs directory should cover technical information about how to use the repository.

Dependencies

This change does not depend on any other blueprints or specs. It can be done largely in parallel with other projects and issues.

13.3.3 Implementation

As described earlier, this will be implemented with ReST files in a docs directory at the root of the repo. Also, there will be a dependency on Sphinx in the dev-requirements, and a script added to run the Sphinx docs build job a tthe gate.

Assignee(s)

Other contributors are welcome to work on the mentioned sections.

Primary assignee:
nolan-brubaker **palendae**

Other contributors:
<launchpad-id or None>

Work items

- Add the docs directory and some basic structure files, like an index page and a Sphinx configuration file.
- Add a file for each section to the docs directory, as well as to the index page.
- Add a Sphinx build job to the gating scripts that only runs if there was a change to the docs directory.

13.3.4 Testing

This change will add a Sphinx build job to the gating process. The Sphinx build job should not run on changes that have no affected docs files.

The Sphinx documentation build job should succeed for the change to merge.

13.3.5 Documentation impact

As mentioned above, this will create a new docs repo that the docs team can then build more detailed documentation in reference to.

13.3.6 References

N/A

13.4 Implement Ceilometer

date
2015-03-31 10:30

tags
kilo, ceilometer

This blueprint was created to add the Celiometer project to OSAD. It will lay out a possible solution that will hopefully in turn create discussion around how to properly implement ceilometer as an OPTIONAL component of OSAD.

- https://blueprints.launchpad.net/openstack-ansible/+spec/implement-ceilometer

13.4.1 Problem description

Currently, OSAD does not implement Ceilometer for various reasons. One being the unstable nature of the project in production environments - causing ever growing database tables resulting in extremely slow API quries. However, some may still want to deploy Ceilometer and may prefer to deal with the problem internally.

Furthermore, these issues should not discount the credibility of Ceilometer as an OpenStack project, and as such, it should be implemented into OSAD.

13.4.2 Proposed change

An additional role, os_ceilometer, would need to be added to handle the installation/configuration of the ceilometer services. Also, additional configuration directives would need to be added to other projects (such as cinder, nova, glance, etc..) so that they will generate notifications.

For the database, the /etc/openstack_deploy/conf.d/ceilometer.yml should give the user an option as to which database they want to connect to. This BP does not implement the deployment of an additional database for Ceilometer, but can be discussed as a potential addition.

Furthermore, new host groups will be added to the env.d/ceilometer.yml file so that the user may specify which infra nodes to install the central agents on , and which compute nodes to install the compute agent on. The proposed changes to the this file will look something like:

```
component_skel:
  ceilometer_agent_compute:
    belongs_to:
    - ceilometer_all
  ceilometer_agent_central:
    belongs_to:
    - ceilometer_all
  ceilometer_agent_notification:
    belongs_to:
    - ceilometer_all
  ceilometer_collector:
    belongs_to:
    - ceilometer_all
  ceilometer_alarm_evaluator:
    belongs_to:
    - ceilometer_all
  ceilometer_alarm_notifier:
    belongs_to:
    - ceilometer_all
  ceilometer_api:
    belongs_to:
    - ceilometer_all

container_skel:
  ceilometer_api_container:
    belongs_to:
      - metering-infra_containers
    contains:
      - ceilometer_agent_central
      - ceilometer_agent_notification
      - ceilometer_collector
      - ceilometer_alarm_evaluator
      - ceilometer_alarm_notifier
      - ceilometer_api
    properties:
      service_name: ceilometer
      container_release: trusty
  metering-compute_container:
    belongs_to:
    - metering-compute_containers
    contains:
    - ceilometer_agent_compute
    properties:
      service_name: ceilometer
      container_release: trusty
      is_metal: true
```

(continues on next page)

(continued from previous page)

```
physical_skel:
  metering-compute_containers:
    belongs_to:
    - all_containers
  metering-compute_hosts:
    belongs_to:
    - hosts
  metering-infra_containers:
    belongs_to:
    - all_containers
  metering-infra_hosts:
    belongs_to:
    - hosts
```

Notable changes:

- New ceilometer role
- New ceilometer playbooks
- A openstack_user_config.yml file in the openstack_deploy/conf.d/ directory specifically for ceilometer configurations.
- Added vars in appropriate files
- Haproxy config changes for ceilometer apis.

Alternatives

Dont do it.

Playbook impact

The ceilometer component should be OPTIONAL. And thus have no effect on other playbooks when chosen not to be run. However, when chosen to be run, other configurations across different projects can be changed to allow notifications.

Upgrade impact

None known

Security impact

Credentials for the Ceilometer database may need to be specified by the user, as this BP does not implement the deployment of the Ceilometer database at this moment.

Performance impact

When ceilometer is enabled, the horizon dashboard can potentially slow down due to large API responses from ceilometer. This is related to the problem stated in the Problem Description. We must let the user know about this known problem, and advise the ceilometer database be properly supervised. When ceilometer is disabled, it should have no performance impact.

End user impact

The user will now have the option to enable ceilometer on their private cloud.

Deployer impact

Additional configs must be specified in /etc/openstack_deploy/conf.d/ceilometer.yml only if the deployer wants ceilometer to be enabled.

The deployer needs to be aware of the database that is going to be used for ceilometer.

Developer impact

A new role will be created to handle the installation/configuration of ceilometer. This role will be run explicitly by the deployer.

Dependencies

N/A

13.4.3 Implementation

Assignee(s)

Primary Assignee(s)

Miguel Alejandro Cantu Sudarshan Acharya

Other contributors:

Work items

At the very least, the following need to get done:

- New ceilometer role
- New ceilometer playbooks
- A openstack_user_config.yml file in the openstack_deploy/conf.d/ directory specifically for ceilometer configurations.
- Added vars in the appropriate sections.
- Haproxy config changes for ceilometer apis.

The following files need to be modified:

- etc/openstack_deploy/env.d/openstack_environment.yml
- etc/openstack_deploy/user_secrets.yml
- etc/openstack_deploy/user_variables.yml
- playbooks/inventory/group_vars/all.yml
- playbooks/vars/configs/haproxy_config.yml
- playbooks/vars/repo_packages/openstack_services.yml
- playbooks/roles/os_cinder/defaults/main.yml
- playbooks/roles/os_cinder/templates/cinder.conf.j2
- playbooks/roles/os_glance/defaults/main.yml
- playbooks/roles/os_glance/templates/glance-api.conf.j2
- playbooks/roles/os_glance/tempaltes/glance-registry.conf.j2
- playbooks/roles/os_heat/defaults/main.yml
- playbooks/roles/os_heat/templates/heat.conf.j2
- playbooks/roles/os_nova/defaults/main.yml
- playbooks/roles/os_nova/templates/nova.conf.j2
- playbooks/roles/os_swift/defaults/main.yml
- playbooks/roles/os_swift/tasks/swift_service_setup.yml
- playbooks/roles/os_swift/templates/proxy-server.conf.j2

The following files need to added:

- etc/openstack/deploy/conf.d/ceilometer.yml.example
- playbooks/os-ceilometer-install.yml
- playbooks/roles/os_ceilometer/
  - CONTRIBUTING.rst
  - LICENSE

- README.rst
- defaults/
  - main.yml
- handlers/
  - main.yml
- meta/
  - main.yml
- templates/
  - ceilometer.conf.j2
  - api_paste.ini.j2
  - ceilometer-upstart-init.j2
  - event_definitions.yaml.j2
  - event_pipeline.yaml.j2
  - pipeline.yaml.j2
  - sudoers.j2
- tasks/
  - main.yml
  - ceilometer_install.yml
  - ceilometer_pre_install.yml
  - ceilometer_post_install.yml
  - ceilometer_service_setup.yml
  - ceilometer_service_add.yml
  - ceilometer_upstart_common_init.yml
  - ceilometer_upstart_init.yml

13.4.4 Testing

The gate scripts will need to be modified with the ceilometer configurations turned on. A variable of the like of DEPLOY_CEILOMETER will be added with the appropriate conditionals in place to deploy ceilometer with a mongodb server for testing.

The playbooks will point to a mongodb server deployed on the AIO. A gate script will be created to deploy a simple mongodb server on the AIO using an ansible galaxy role.

13.4.5 Documentation impact

Thorough documentation will need to be created explaining how to enable ceilometer.

13.4.6 References

- http://goo.gl/LA7o6N
- http://goo.gl/Xj7cqT
- https://www.rdoproject.org/CeilometerQuickStart
- http://docs.openstack.org/developer/ceilometer/install/manual.html

13.5 Keystone Federation

date
2015-06-22 08:00

tags
federation, scalability

This spec is to propose adding support for Keystone federation to openstack-ansible.

Launchpad blueprint: https://blueprints.launchpad.net/openstack-ansible/+spec/keystone-federation

Operators of private clouds often have the need for additional capacity or services found in other private and public clouds. OpenStack has accommodated this use case through Keystone Identity Federation, allowing identity credentials from one cloud to act as authorization in another.

The primary use case would be for a private cloud to act as an Identity Provider (IdP) to other clouds, typically Public Clouds. This allows users found in the private cloud databases to authenticate in order to consume resources provided by other Service Provider (SP) clouds.

The secondary use case is where a private clouds would work as a SP to another Private Cloud or external provider acting in the role of an IdP.

13.5.1 Problem description

- As a User, in order to utilize my Keystone identity to consume resources in other Keystone backed Service Providers, I should be able to effectively authenticate with those Service Providers using only my Keystone identity credentials via the Service Providers Command Line Interface (CLI).
- As an Administrator, in order to allow my users to utilize their Keystone identity with other Service Providers, I should be able to establish a trust relationship between my Keystone and a Service Provider Keystone via CLI.
- As an Administrator of multiple clouds, in order to provide identity federation between my multiple clouds, I should be able to establish a trust relationship between my Keystone Identity Provider Cloud and my Keystone Service Provider clouds.
- As an Administrator, in order to effectively map Identity Provider groups and users to Service Provider roles, I should be able to simply define mappings to Service Provider projects, domains and roles for given groups.

- As a Deployer, in order to prevent downtime or interruption, I should be able to setup my cloud as an Identity Provider or Service Provider with little or no interruption to the data plane.
- As a User, in order to understand the resources available to me, I should be able to retrieve a list of Service Providers which trust my Identity Provider as well as a service catalog for the services offered by those Service Providers.
- As an Administrator, in order to use Identity Federation between my secured network Private Cloud and other Public Service Providers, I should be able to easily establish a trust relationship between the two without compromising my network security.

13.5.2 Proposed change

1. Enable and configure Keystone Federation, implementing the IdP/SP configuration in a manner that is simple for deployers and requires little or no data plane downtime. The initial SP configuration will use saml-based authentication and Apache mod_shib. Later options to extend support to would include the saml-based Apache mod_auth_mellon, the OpenID-based Apache mod_auth_openidc, the kerberos-based Apache mod_auth_kerb/mod_auth_identity.
2. Improve the configuration of Keystone SSL endpoints to ensure that both the IdP and SP public interfaces can be served via SSL using a supplied server key, server certificate, Certificate Authority certificate and (optionally) an intermediary certificate.
3. Change the Keystone and Utility containers to use the python-openstackclient instead of the python-keystoneclient in order to ensure that the Keystone v3 API may be used. This is required for the administration of Federation IdP and SP configuration entities.
4. Change the Horizon configuration to allow it to consume the Keystone v3 API.
5. Automate the registration of a trusted IdP to an SP.
6. Automate the registration of a list of trusted SPs to an IdP.
7. Document and, if possible, automate the registration and mapping of external identities to specified domains, projects, roles and users.

Alternatives

None

Playbook/Role impact

1. The os_keystone role will require changes to both tasks and templates in order to facilitate the configuration of the IdP, SP, openstackclient and SSL.
2. The os_horizon configuration will require changes to the templates in order to facilitate the change to use the Keystone v3 API.
3. The openstack_openrc role may need to be changed in order to place a different openrc file into the keystone and utility containers.
4. The automation of registration and mapping of external identities to specific domains, projects, roles and users may be done in a new playbook/role or within the existing keystone playbook/role.

5. The os_keystone role will need to include the capability to replicate the same SP signing certificates from the first Keystone container to all the others.

Upgrade impact

Horizon will be reconfigured to use the Keystone v3 API.

Security impact

1. Keystone must have its public endpoint implemented with SSL in order to encrypt and protect authentication and identity information which is communicated between the IdP and the SP.
2. The SP signing key and certificate must be properly secured on every server that has it.

Performance impact

TBD

End user impact

1. Administration of keystone via the v3 API will mean switching from using the keystone CLI to using the openstack CLI.

Deployer impact

1. The deployer will be able to implement specified SSL certificates for the Keystone public endpoints.

Developer impact

1. The keystone Ansible module will be updated to make use of the keystone v3 API.

Dependencies

1. Keystone IdP requires the following: * xmlsec1: http://packages.ubuntu.com/search?keywords=xmlsec1 * python-openstackclient: https://pypi.python.org/pypi/python-openstackclient
2. Keystone SP requires the following: * xmlsec1: http://packages.ubuntu.com/search?keywords=xmlsec1 * libapache2-mod-shib2: http://packages.ubuntu.com/search?keywords=libapache2-mod-shib2 * python-openstackclient: https://pypi.python.org/pypi/python-openstackclient
3. Keystone mapping documentation: * https://review.openstack.org/192850
4. Keystone SP must use uuid tokens for now * https://bugs.launchpad.net/keystone/+bug/1471289

13.5.3 Implementation

Assignee(s)

Primary assignee:
https://launchpad.net/~miguelgrinberg (miguelgrinberg)

Other contributors:
https://launchpad.net/~hughsaunders (hughsaunders) https://launchpad.net/~icordasc (sigmavirus24) https://launchpad.net/~jesse-pretorius (odyssey4me)

Work items

1. Convert existing Keystone Ansible module to use v3 API
2. Add federation commands to Keystone Ansible Module
3. Keystone public endpoint SSL configuration
4. Keystone/Utility container implementation of python-openstackclient
5. Keystone IdP software deployment, configuration and SP registration
6. Keystone SP software deployment, configuration and IdP registration
7. Document and, if possible, automate the registration and mapping of external identities to specified domains, projects, roles and users.

13.5.4 Testing

Due to the nature of this feature requiring two independant installations there will be no specific gate testing for it.

All changes implemented in the roles/plays as a result of this work will need to be done in such a way that the existing gate checks continue to pass.

13.5.5 Documentation impact

1. The upgrade impact will need to be noted in the release notes.
2. The method of implementing the required user_variables for an IdP/SP will need to be described.
3. The specifics of registering and mapping external identities to domains, projects, roles and users will need to be documented.

13.5.6 References

- http://docs.openstack.org/developer/keystone/configure_federation.html
- http://docs.openstack.org/developer/keystone/extensions/federation.html
- http://docs.openstack.org/developer/keystone/extensions/shibboleth.html
- http://blog.rodrigods.com/it-is-time-to-play-with-keystone-to-keystone-federation-in-kilo/
- https://zenodo.org/record/11982/files/CERN_openlab_Luca_Tartarini.pdf

13.6 Keystone Service Provider with ADFS Identity Provider Deployment

date
2015-06-22 10:00

tags
federation, scalability

This spec is to propose adding support to openstack-ansible for Keystone federation using an Active Directory Federation Service (ADFS) Identity Provider.

Launchpad blueprint: https://blueprints.launchpad.net/openstack-ansible/+spec/keystone-sp-adfs-idp

OpenStack cloud deployers frequently utilize Microsoft Active Directory (AD) as a corporate identity provider. In this case, provisioning user credentials specifically for their OpenStack clouds, and managing/updating the corresponding permissions for those users is burdensome. Deployers would rather use Keystones Federation capabilities with ADFS to have AD act as an Identity Provider (IdP) to Keystone as a Service Provider (SP).

13.6.1 Problem description

- As a User, in order to utilise my AD identity to consume resources in my OpenStack Cloud, I should be able to authenticate to my OpenStack Cloud using my AD credentials via the Service Providers Horizon Dashboard and Command Line Interface (CLI).
- As an Administrator, in order to maintain one identity system, I should be able to create a trust relationship between my ADFS IdP and my OpenStack SPs.
- As an Administrator, in order to effectively map Identity Provider groups and users to Service Provider roles, I should be able to simply define mappings to Service Provider projects, domains and roles for given groups.

13.6.2 Proposed change

1. Enable and configure Keystone as a federated SP with SSL public endpoints. The initial SP configuration will use saml-based Apache mod_shib. Later options to extend support to would include Apache mod_auth_mellon.
2. Document the configuration of the ADFS IdP in order to support the Keystone SP.
3. Change the Horizon configuration to support Web Single-Sign-On (SSO), thereby providing support for end-users to authenticate using their AD credentials.
5. Automate the registration of the trusted ADFS IdP to the Keystone SP.
7. Document and, if possible, automate the registration and mapping of external identities to specified domains, projects, roles and users.

Alternatives

None

Playbook impact

1. The os_horizon configuration will require changes to the templates in order to facilitate the change to use WebSSO.

Upgrade impact

None

Security impact

There are security aspects, but they affect docs more than code:

- Security is to some extent delegated to the external IDP (AD). Therefore Deployers must be confident of the security of their AD before using it for federation.
- Deployers must take time to understand the mapping mechanisms in order to ensure that only the expected users/groups are granted access to OpenStack resources.

Performance impact

TBD

End user impact

1. If an external IdP is configured, Horizon will show multiple methods of authentication available via a drop-down list. The user will be able to choose between credentials and the available WebSSO sources.

Deployer impact

None

Developer impact

None

Dependencies

1. Keystone Federation Deployment Implementation: * https://blueprints.launchpad.net/openstack-ansible/+spec/keystone-federation
2. Horizon requires the following: * django-openstack-auth v1.2.0 or higher: https://pypi.python.org/pypi/django_openstack_auth

13.6.3 Implementation

Assignee(s)

Primary assignee:
https://launchpad.net/~jesse-pretorius (odyssey4me)

Other contributors:
https://launchpad.net/~hughsaunders (hughsaunders) https://launchpad.net/~icordasc (sigmavirus24) https://launchpad.net/~miguelgrinberg (miguelgrinberg)

Work items

1. Add the required ADFS configuration to the Keystone SP. * shibboleth2.xml * attribute-map.xml
2. Document the configuration of the ADFS IdP in order to support the Keystone SP.
3. Automate the registration of the trusted ADFS IdP to the Keystone SP.
4. Change the Horizon configuration to support Web Single-Sign-On (SSO), thereby providing support for end-users to authenticate using their AD credentials.
5. Document and, if possible, automate the registration and mapping of external identities to specified domains, projects, roles and users.

13.6.4 Testing

Due to the nature of this feature requiring an installation of ADFS (which is not possible in OpenStack-CI) there will be no specific gate testing for it.

All changes implemented in the roles/plays as a result of this work will need to be done in such a way that the existing gate checks continue to pass.

13.6.5 Documentation impact

1. The preparation of the ADFS IdP to support the Keystone SP will need to be described.
2. The method of implementing the required user_variables for the Keystone SP will need to be described.
3. The specifics of registering and mapping external identities to domains, projects, roles and users will need to be documented.

13.6.6 References

- http://docs.openstack.org/developer/keystone/extensions/websso.html
- http://specs.openstack.org/openstack/keystone-specs/specs/kilo/websso-portal.html
- https://wiki.shibboleth.net/confluence/display/SHIB2/NativeSPADFS
- https://zenodo.org/record/11982/files/CERN_openlab_Luca_Tartarini.pdf

13.7 Master Kilofication

date
: 2015-03-23 13:00

tags
: kilo, update,

Update the various openstack-ansible playbooks and roles in the master branch with the changes necessary to implement a fully functional and updated kilo deployment.

- https://blueprints.launchpad.net/openstack-ansible/+spec/master-kilofication

Initial work will be based on the k3 tags in each of the openstack projects since kilo is not yet officially released.

13.7.1 Problem description

Master is setup to deploy Juno at this time we want the master branch to begin tracking Kilo.

13.7.2 Proposed change

As opposed to the minimal-kilo blueprint which is focused on making the minumum fewest possible changes necessary to point at kilo and have a deployment that passes gating, this specification is targeted more at updating all config files and code to bring in the kilo versions of the configs for each service, parsing each file for differences and making informed decisions about what values to take to ensure we have a production grade deployment system.

The approach to dealing with differences (eg changed defaults for a particular setting) will be to use the kilo value where possible, adding an option to make any changed setting tunable if it was not already. This gives the option to users who are upgrading from juno to be able to reset a value back to the juno default if desired, but also means that greenfield deployments of kilo use the (hopefully better) kilo value.

Examples of configs impacted (these will differ depending on the service being worked on):

```
/etc/<servicename>/<servicename>.conf
/etc/<servicename>/<servicename>-api-paste.ini
/etc/<servicename>/policy.json
/etc/<servicename>/<servicename>-<agentname>.ini
```

Alternatives

We could, wherever needed, preserve juno settings rather than taking forward the kilo settings. This is potentially easier on users in an upgrade scenario, but does mean that new users deploying kilo would get an already out of date deployment. It also means that we miss an opportunity to implement best practices deployments, instead sticking on old, less relevant, values.

Playbook impact

There will be no impact on the playbooks. These changes are on the dependency and role level which only impact the configuration files and role options.

Upgrade impact

This change will impact upgrades, but upgrades are out of scope for this spec which will be addressed separately. Largely it addresses greenfield deployments of kilo.

Security impact

These changes will initially be based on BETA code (k3 and rc1 tags of kilo) which may have consequences regarding security, but work will be done to test against production kilo when it is released (and prior to the 11.0.0 release of openstack-ansible being tagged)

Performance impact

Because the Kilo code base is not tested and released, the performance of the stack will not be in scope at this time. As future work develops to finalize the roles used in Kilo, work will be done on a per role basis to ensure performance.

End user impact

N/A

Deployer impact

As stated previously, this change will initially introduce new BETA code. Deployers shouldnt be using master at this time.

Developer impact

This change is to allow development of a production grade kilo deployment

Dependencies

The spec will introduce a number of new dependencies. At this time not all are exactly known. However, we can safely say that all new clients will be used throughout the stack as well as various middlewares.

13.7.3 Implementation

Assignee(s)

Various

Work items

Unknown at this time

13.7.4 Testing

No changes to the current testing and or gating framework will be made. Each change that is made to a service to bring forward new configs and settings will be required to pass the same gate tests as are required by our production systems.

13.7.5 Documentation impact

This change will likely have documentation impact. Specifically when documenting changed values or deprecated config items.

13.7.6 References

N/A

13.8 Minimal Kilo

date
2015-03-17 21:34

tags
kilo, minimum, update,

Update master to point to the minimum configuration nessisary for a functional kilo stack.

- https://blueprints.launchpad.net/openstack-ansible/+spec/minimal-kilo

This spec is being created to track the work required to get a minimum viable deployment of kilo. Because the Kilo release of OpenStack has not yet been released the work done within this blueprint will pull from the head of master and stabilize on the a given sha for the time being.

13.8.1 Problem description

Master is setup to deploy Juno at this time we want the master branch to begin tracking Kilo.

13.8.2 Proposed change

In order to have a minimally functional Kilo stack there are several issues that need to be resolved which have been raised within Launchpad. Once the following issues are resolved Kilo should be a functional deployment from the stand point of gating. The point of this Spec is to introduce the least amount of changes into the stack in an effort to enable a Kilo code base. The changes should pass gating from the a commit basis. Once this spec is complete other work can follow to make Kilo a production ready product.

Alternatives

There are no alternatives to this approach. Without a bulk commit to address the minimal changes to get Kilo functional we will not be able to move forward with development.

Playbook impact

There will be no impact on the playbooks. These changes are on the dependency and role level which only impact the configuration files and role options.

Upgrade impact

This change will impact upgrades. The change will introduce new code which will allow the system to upgrade inplace. That said, this is a transitional spec which will translate into future work to make Kilo a production ready product. Upgrades are out of the scope of this spec and it is expected that Juno to Kilo upgrades will be broken at this point.

Security impact

These changes will introduce BETA code which will likely have consequences regarding security however the changes are not geared at production at this time and will be revised in a fast follow effort.

Performance impact

Because the Kilo code base is not tested and released the performance of the stack will not be in scope at this time. As future work develops to finalize the roles used in Kilo work will be done on a per role basis to ensure performance.

End user impact

N/A

Deployer impact

As stated previously, this change will introduce new BETA code. Deployers shouldnt be using master at this time.

Developer impact

This change is geared at enabling developers to begin working on Kilo.

Dependencies

The spec will introduce a number of new dependencies. At this time not all are exactly known. However, we can safely say that all new clients will be used throughout the stack as well as various middlewares.

13.8.3 Implementation

Assignee(s)

Primary assignee:
https://launchpad.net/~kevin-carter

Other contributors:
https://launchpad.net/~nolan-brubaker

IRC: cloudnull, palendae

Work items

In order to have a minimum viable installation of OpenStack Kilo the following issues will need to be addressed.

- #1428421 Keystone.py needs to be updated for kilo
- #1428431 OpenStack Clients need to be updated for Kilo
- #1428437 Update/Removal of pinned Oslo Messaging and Middleware for kilo
- #1428445 Neutron needs plugin references removed for kilo
- #1428451 Heat policy.json file needs to be updated for Kilo
- #1428469 Neutron rootwarp(s) need to be updated for Kilo
- #1428639 Nova requires python-libguestfs in Kilo

13.8.4 Testing

No changes to the current testing and or gating framework will be made. The minimum viable Kilo deployment will be required to pass the same gate tests as are required by our production systems.

13.8.5 Documentation impact

This change specifically does not have any documentation impact.

13.8.6 References

N/A

13.9 Modularizing Neutron plays for agents and non ml2 plugin support

date
2015-03-30 16:35

tags
neutron, plugins, agents

This spec is propsed to enhance the current neutron playbooks that take a static approach to plugin and agent insertion. Where ml2 and a few agents are used by default.

- https://blueprints.launchpad.net/openstack-ansible/+spec/modularize-neutron-plays

13.9.1 Problem Description

Presently a straightforward approach does not exist to add new plugins and add / remove agents to the neutron setup. A deployer either has to perform these changes after the whole setup is complete or make his own changes in the playbooks.

13.9.2 Proposed Change

This feature is proposed for both master and juno branches, the juno effort will be carried out first:

1. For juno, the openstack/roles/neutron_common.yml will be modified to install a configurable list of plugins and agents through new variables defined in inventory/group_vars/neutron_all. The default values to these new variables with be the current set of installed agents and plugins.

2. For master, the playbooks/roles/os_neutron/tasks files will be modified, particularly neutron_post_install.yml. Addition of new parameters will be made to playbooks/roles/os_neutron/defaults/main.yml

Playbook Impact

1. In juno, the following files are expected to be modified:
   - openstack/roles/neutron_common.yml
   - openstack/inventory/group_vars/neutron_all.yml
2. In master, these files will be modified:
   - playbooks/roles/os_neutron/tasks/neutron_post_install.yml
   - playbooks/roles/os_neutron/defaults/main.yml

Upgrade Impact

None

Alternatives

Using the current architecture, prospective new plugins which are not ml2 will have to take an overwriting the default configuration, after its done, approach to insert their own changes.

Security Impact

None known at this time.

Performance Impact

This change is not expected to impact performance. Installing the default set of agents and plugins as done now, will take the same amount of effort.

End User Impact

This is not expected to impact end users as it deals with the deployment aspect only.

Deployer Impact

This will introduce a more modular architecture for deployers to select neutron plugins/agents from, allowing a wider use case for these playbooks.

Developer Impact

Using the default values will require no new developer effort, only those interested in changing the neutron config will be effected.

Dependencies

N/A

13.9.3 Implementation

Assignee(s)

Primary assignee:
https://launchpad.net/~javeria-ak

Work items

This change will include modifying the existing neutron_common role to pick up what plugin to install along with what agents. The names and configs for individual plugins will be created as new variables in inventory/group_vars/neutron_all.yml

Dependencies

N/A

13.9.4 Testing

There are no additional changes required to test this in the current testing and or gating framework.

13.9.5 Documentation Impact

A bit of additional documentation describing how to insert new plugins/agents will be required. This will be deployer documentation.

13.9.6 References

N/A

13.10 Multi-Region Swift

date
2015-07-03 13:00

tags
kilo, swift

This blueprint was created to add Multi-Region Swift support to OSAD. It will lay out a series of use cases to define the requirements of Multi-Region Swift within OSAD.

- https://blueprints.launchpad.net/openstack-ansible/+spec/multi-region-swift

A Swift cluster can be deployed in such a way that the cluster spans multiple geographically dispersed data centers. This allows an end-user to ensure resiliency in the event of a data center failure, with one or more copies of an object stored in each data center location. This facilitates end-users being able to build out geographically dispersed infrastructure, enabling high availability.

13.10.1 Problem description

- As a User I want to be able to store or retrieve objects in any location of a multi-location object storage solution, using the same credentials.
- As a User, I want my default object storage/retrieval location to be as close to my location as possible - by specifying the closest endpoint and having that endpoint access the closest storage locations.
- As an Administrator, I want to be able to set a global storage policy that enables me to specify the number of copies of every object to be stored in each location of the multi-location object storage based on one of the following predefined scenarios:
  - With 3 or more storage deployments, each location is considered to be as important as any other location. The same number of copies should be kept in each location.
  - With 3 storage deployments we consider 1 to be the primary and the other 2 as geographically convenient locations for read purposes. There should be 2 copies in the primary location and 1 in each of the other locations.
  - With 2 storage deployments, we consider 1 to be the primary location and the other a backup location. There should be 2 copies in the primary location and 1 copy in the backup location.
- As a User, I want to be assured durability of my content under various circumstances, such as:
  - Initial upload to single location. i.e. in the case of uploading an object to location A in a 3 location solution where the global policy is 1 locally, 1 in each remote - until the 2 remote objects are confirmed, the local object storage cluster will have 3 copies.
  - Failure of 1 or more locations. i.e. In a 3 location solution where the global policy is 1 locally, 1 in each remote - if location A fails, either location B or C will generate a second copy of the missing objects to ensure that there are always 3 copies.
- As an Administrator, I want to be able to override the global storage policy at a container level in order to increase or reduce replication. For example:
  - One or more containers can have an alternate policy specified at the container level that overrides the global policy. If the global policy in a 3 location solution is 1 locally and 1 in each remote, one or more container can be configured to a 3 locally policy.
- As a User, where the global storage policy has been configured to replicate objects across locations, I want to be able to retrieve my objects from an alternate location in the event of a failure of my default location.
- As an Administrator, I want to be assured that when replicating data from the primary location to the remote locations the data is secured and not transmitted in the clear.

13.10.2 Proposed change

1. Enable the use of the read_affinity, write_affinity and write_affinity_node_count settings within Swift on a swift-proxy host basis. This will allow the prioritization of reads/writes based on region.
2. Enable some form of encryption to ensure the replication of objects across locations is secure.
3. Configure the management of the ring and keys required for communication between swift storage hosts across multiple locations and deployments.

4. Document the use of the read_affinity, write_affinity and write_affinity_node_count settings within openstack-ansible and provide guidance around how to implement the specified Use Cases.
5. Document the process of setting up a global swift cluster within openstack-ansible.
6. Adjust settings to allow the use of keystone v3 API for swift, in the swift configuration files.

Alternatives

We could not enable swift multi-region support.

Playbook impact

Whilst the Multi-Region component will be optional, we will need to implement the following changes without adjusting how the current default operates:

1. The Multi-Region component will require adjustments to ring/key management for swift hosts, as well as some changes to how the Swift inventory is managed within the user_config.yml (or conf.d/swift.yml) files.
2. Read_affinity, write_affinity and write_affinity_node_count will need to be added configuration for proxy-servers.
3. The synchronization of the swift-ring will needed to be handled across nodes in all locations/regions.

Upgrade impact

This should have no upgrade impact.

Security impact

Since swift does not handle encryption of objects this will need to be handled externally to swift.

Performance impact

N/A

End user impact

The user will now have the option to configure a Multi-Region Swift cluster. The default will remain the same, so it should not impact any users who do not wish to utilise a Multi-Region Swift cluster.

Deployer impact

A deployer will be able to adjust the inventory across multiple deploys to ensure a global swift cluster operating unformly for all deploys.

Developer impact

None

Dependencies

None known

13.10.3 Implementation

Assignee(s)

Primary assignee:
https://launchpad.net/~andrew-mccrae (andymccr)

Other contributors:
https://launchpad.net/~steve-lewis (stevelle) https://launchpad.net/~tom-cameron (rackertom) https://launchpad.net/~apsu-2 (Apsu) https://launchpad.net/~prometheanfire (prometheanfire)

Work items

1. Enable the use of the read_affinity, write_affinity and write_affinity_node_count settings within Swift on a swift-proxy host basis. This will allow the prioritization of reads/writes based on region.
2. Enable some form of encryption to ensure the replication of objects across locations is secure.
3. Configure the management of the ring and keys required for communication between swift storage hosts across multiple locations and deployments.
4. Document the use of the read_affinity, write_affinity and write_affinity_node_count settings within openstack-ansible and provide guidance around how to implement the specified Use Cases.
5. Document the process of setting up a global swift cluster within openstack-ansible.
6. Adjust settings to allow the use of keystone v3 API.

13.10.4 Testing

As this will require two independent installations of swift we wont add anything specific to the gate to automatically test this. However the changes should not adjust how current tests work and all changes will need to ensure that existing tests continue to pass.

13.10.5 Documentation impact

1. Use case implementation will need to be documented
2. Implementation of a global cluster and the settings required.
3. New network requirements will need to be documented.
4. Inventory management, and configuration options that are added as a result will need to be documented.

13.10.6 References

- http://docs.openstack.org/developer/swift/admin_guide.html#geographically-distributed-clusters
- https://swiftstack.com/blog/2012/09/16/globally-distributed-openstack-swift-cluster/
- search