Blazar Specs

Release 0.0.1.dev30

OpenStack Foundation

Feb 04, 2026

CONTENTS

CHAPTER

ONE

EXAMPLE SPEC - THE TITLE OF YOUR BLUEPRINT

Include the URL of your launchpad blueprint:

https://blueprints.launchpad.net/blazar/+spec/example

Introduction paragraph why are we doing anything? A single paragraph of prose that operators can understand. The title and this first paragraph should be used as the subject line and body of the commit message respectively.

Some notes about the blazar-spec and blueprint process:

- Not all blueprints need a spec.
- The aim of this document is first to define the problem we need to solve, and second agree the overall approach to solve that problem.
- This is not intended to be extensive documentation for a new feature. For example, there is no need to specify the exact configuration changes, nor the exact details of any DB model changes. But you should still define that such changes are required, and be clear on how that will affect upgrades.
- You should aim to get your spec approved before writing your code. While you are free to write prototypes and code before getting your spec approved, its possible that the outcome of the spec review process leads you towards a fundamentally different solution than you first envisaged.
- But, API changes are held to a much higher level of scrutiny. As soon as an API change merges, we must assume it could be in production somewhere, and as such, we then need to support that API change forever. To avoid getting that wrong, we do want lots of details about API changes upfront.

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/blazar/+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
- For help with syntax, see http://sphinx-doc.org/rest.html
- To test out your formatting, build the docs using tox and see the generated HTML file in doc/build/html/specs/<path_of_your_file>
- 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.

- If your specification proposes any changes to the Blazar REST API such as changing parameters which can be returned or accepted, or even the semantics of what happens when a client calls into the API, then you should add the APIImpact flag to the commit message. Specifications with the APIImpact flag can be found with the following query:

  https://review.opendev.org/#/q/project:openstack/blazar-specs+message:apiimpact,n,z

1.1 Problem description

A detailed description of the problem. What problem is this blueprint addressing?

1.1.1 Use Cases

What use cases does this address? What impact on actors does this change have? Ensure you are clear about the actors in each use case: Developer, End User, Deployer etc.

1.2 Proposed change

Here is where you cover the change you propose to make in detail. How do you propose to solve this problem?

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?

At this point, if you would like to just get feedback on if the problem and proposed change fit in blazar, you can stop here and post this for review to get preliminary feedback. If so please say: Posting to get preliminary feedback on the scope of this spec.

1.2.1 Alternatives

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.

1.2.2 Data model impact

Changes which require modifications to the data model often have a wider impact on the system. The community often has strong opinions on how the data model should be evolved, from both a functional and performance perspective. It is therefore important to capture and gain agreement as early as possible on any proposed changes to the data model.

Questions which need to be addressed by this section include:

- What new data objects and/or database schema changes is this going to require?
- What database migrations will accompany this change.
- How will the initial set of new data objects be generated, for example if you need to take into account existing instances, or modify other existing data describe how that will work.

1.2.3 REST API impact

Each API method which is either added or changed should have the following

- Specification for the method
  - A description of what the method does suitable for use in user documentation
  - Method type (POST/PUT/GET/DELETE)
  - Normal http response code(s)
  - Expected error http response code(s)
    - A description for each possible error code should be included describing semantic errors which can cause it such as inconsistent parameters supplied to the method, or when an instance is not in an appropriate state for the request to succeed. Errors caused by syntactic problems covered by the JSON schema definition do not need to be included.
  - URL for the resource
    - URL should not include underscores, and use hyphens instead.
  - Parameters which can be passed via the url
  - JSON schema definition for the request body data if allowed
    - Field names should use snake_case style, not CamelCase or MixedCase style.
  - JSON schema definition for the response body data if any
    - Field names should use snake_case style, not CamelCase or MixedCase style.
- Example use case including typical API samples for both data supplied by the caller and the response
- Discuss any policy changes, and discuss what things a deployer needs to think about when defining their policy.

Note that the schema should be defined as restrictively as possible. Parameters which are required should be marked as such and only under exceptional circumstances should additional parameters which are not defined in the schema be permitted (eg additionaProperties should be False).

Reuse of existing predefined parameter types such as regexps for passwords and user defined names is highly encouraged.

1.2.4 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 the 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.

- Can this change enable a resource exhaustion attack, such as allowing a single API interaction to consume significant server resources? Some examples of this include launching subprocesses for each connection, or entity expansion attacks in XML.

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.

1.2.5 Notifications impact

Please specify any changes to notifications. Be that an extra notification, changes to an existing notification, or removing a notification.

1.2.6 Other end user impact

Aside from the API, are there other ways a user will interact with this feature?

- Does this change have an impact on python-blazarclient? What does the user interface there look like?

1.2.7 Performance Impact

Describe any potential performance impact on the system, for example how often will new code be called, and is there a major change to the calling pattern of existing code.

Examples of things to consider here include:

- A small change in a utility function or a commonly used decorator can have a large impacts on performance.
- Calls which result in a database queries can have a profound impact on performance when called in critical sections of the code.
- Will the change include any locking, and if so what considerations are there on holding the lock?

1.2.8 Other deployer impact

Discuss things that will affect how you deploy and configure OpenStack that have not already been mentioned, such as:

- What config options are being added? Should they be more generic than proposed (for example a flag that other hypervisor drivers might want to implement as well)? 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 directory name that instances are stored in, how do we handle instance directories created before the change landed? Do we move them? Do we have a special case in the code? Do we assume that the operator will recreate all the instances in their cloud?

1.2.9 Developer impact

Discuss things that will affect other developers working on OpenStack, such as:

- If the blueprint proposes a change to the driver API, discussion of how other hypervisors would implement the feature is required.

1.2.10 Upgrade impact

Describe any potential upgrade impact on the system.

1.3 Implementation

1.3.1 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>

1.3.2 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.

Consider creating an ordering of patches, that allows gradually merging instead of the need to merge them all at once.

1.4 Dependencies

- Include specific references to specs and/or blueprints in blazar, or in other projects, that this one either depends on or is related to.
- If this requires functionality of another project that is not currently used by Blazar, document that fact.
- Does this feature require any new library dependencies or code otherwise not included in OpenStack? Or does it depend on a specific version of library?

1.5 Testing

Please discuss the important scenarios needed to test here, as well as specific edge cases we should be ensuring work correctly. For each scenario please specify if this requires specialized hardware, a full openstack environment, or can be simulated inside the Blazar tree.

Please discuss how the change will be tested. We especially want to know what tempest tests will be added. It is assumed that unit test coverage will be added so that doesnt need to be mentioned explicitly,

but discussion of why you think unit tests are sufficient and we dont need to add more tempest tests would need to be included.

Is this untestable in gate given current limitations (specific hardware / software configurations available)? If so, are there mitigation plans (3rd party testing, gate enhancements, etc).

1.6 Documentation Impact

Which audiences are affected most by this change, and which documentation titles on docs.openstack.org should be updated because of this change? Dont repeat details discussed above, but reference them here in the context of documentation for multiple audiences. For example, the Operations Guide targets cloud operators, and the End User Guide would need to be updated if the change offers a new feature available through the CLI or dashboard. If a config option changes or is deprecated, note here that the documentation needs to be updated to reflect this specifications change.

1.7 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 notes from a summit session
- Links to relevant research, if appropriate
- Related specifications as appropriate
- Anything else you feel it is worthwhile to refer to

1.8 History

Optional section intended to be used each time the spec is updated to describe new design, API or any database schema updated. Useful to let reader understand whats happened along the time.

Table 1: Revisions

Release Name	Description
Zed	Introduced

CHAPTER

TWO

EXAMPLE SPEC - THE TITLE OF YOUR BLUEPRINT

Include the URL of your launchpad blueprint:

https://blueprints.launchpad.net/blazar/+spec/example

Introduction paragraph why are we doing anything? A single paragraph of prose that operators can understand. The title and this first paragraph should be used as the subject line and body of the commit message respectively.

Some notes about the blazar-spec and blueprint process:

- Not all blueprints need a spec.
- The aim of this document is first to define the problem we need to solve, and second agree the overall approach to solve that problem.
- This is not intended to be extensive documentation for a new feature. For example, there is no need to specify the exact configuration changes, nor the exact details of any DB model changes. But you should still define that such changes are required, and be clear on how that will affect upgrades.
- You should aim to get your spec approved before writing your code. While you are free to write prototypes and code before getting your spec approved, its possible that the outcome of the spec review process leads you towards a fundamentally different solution than you first envisaged.
- But, API changes are held to a much higher level of scrutiny. As soon as an API change merges, we must assume it could be in production somewhere, and as such, we then need to support that API change forever. To avoid getting that wrong, we do want lots of details about API changes upfront.

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/blazar/+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
- For help with syntax, see http://sphinx-doc.org/rest.html
- To test out your formatting, build the docs using tox and see the generated HTML file in doc/build/html/specs/<path_of_your_file>
- 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.

- If your specification proposes any changes to the Blazar REST API such as changing parameters which can be returned or accepted, or even the semantics of what happens when a client calls into the API, then you should add the APIImpact flag to the commit message. Specifications with the APIImpact flag can be found with the following query:

  https://review.opendev.org/#/q/project:openstack/blazar-specs+message:apiimpact,n,z

2.1 Problem description

A detailed description of the problem. What problem is this blueprint addressing?

2.1.1 Use Cases

What use cases does this address? What impact on actors does this change have? Ensure you are clear about the actors in each use case: Developer, End User, Deployer etc.

2.2 Proposed change

Here is where you cover the change you propose to make in detail. How do you propose to solve this problem?

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?

At this point, if you would like to just get feedback on if the problem and proposed change fit in blazar, you can stop here and post this for review to get preliminary feedback. If so please say: Posting to get preliminary feedback on the scope of this spec.

2.2.1 Alternatives

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.

2.2.2 Data model impact

Changes which require modifications to the data model often have a wider impact on the system. The community often has strong opinions on how the data model should be evolved, from both a functional and performance perspective. It is therefore important to capture and gain agreement as early as possible on any proposed changes to the data model.

Questions which need to be addressed by this section include:

- What new data objects and/or database schema changes is this going to require?
- What database migrations will accompany this change.
- How will the initial set of new data objects be generated, for example if you need to take into account existing instances, or modify other existing data describe how that will work.

2.2.3 REST API impact

Each API method which is either added or changed should have the following

- Specification for the method
  - A description of what the method does suitable for use in user documentation
  - Method type (POST/PUT/GET/DELETE)
  - Normal http response code(s)
  - Expected error http response code(s)
    - A description for each possible error code should be included describing semantic errors which can cause it such as inconsistent parameters supplied to the method, or when an instance is not in an appropriate state for the request to succeed. Errors caused by syntactic problems covered by the JSON schema definition do not need to be included.
  - URL for the resource
    - URL should not include underscores, and use hyphens instead.
  - Parameters which can be passed via the url
  - JSON schema definition for the request body data if allowed
    - Field names should use snake_case style, not CamelCase or MixedCase style.
  - JSON schema definition for the response body data if any
    - Field names should use snake_case style, not CamelCase or MixedCase style.
- Example use case including typical API samples for both data supplied by the caller and the response
- Discuss any policy changes, and discuss what things a deployer needs to think about when defining their policy.

Note that the schema should be defined as restrictively as possible. Parameters which are required should be marked as such and only under exceptional circumstances should additional parameters which are not defined in the schema be permitted (eg additionaProperties should be False).

Reuse of existing predefined parameter types such as regexps for passwords and user defined names is highly encouraged.

2.2.4 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 the 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.

- Can this change enable a resource exhaustion attack, such as allowing a single API interaction to consume significant server resources? Some examples of this include launching subprocesses for each connection, or entity expansion attacks in XML.

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.

2.2.5 Notifications impact

Please specify any changes to notifications. Be that an extra notification, changes to an existing notification, or removing a notification.

2.2.6 Other end user impact

Aside from the API, are there other ways a user will interact with this feature?

- Does this change have an impact on python-blazarclient? What does the user interface there look like?

2.2.7 Performance Impact

Describe any potential performance impact on the system, for example how often will new code be called, and is there a major change to the calling pattern of existing code.

Examples of things to consider here include:

- A small change in a utility function or a commonly used decorator can have a large impacts on performance.
- Calls which result in a database queries can have a profound impact on performance when called in critical sections of the code.
- Will the change include any locking, and if so what considerations are there on holding the lock?

2.2.8 Other deployer impact

Discuss things that will affect how you deploy and configure OpenStack that have not already been mentioned, such as:

- What config options are being added? Should they be more generic than proposed (for example a flag that other hypervisor drivers might want to implement as well)? 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 directory name that instances are stored in, how do we handle instance directories created before the change landed? Do we move them? Do we have a special case in the code? Do we assume that the operator will recreate all the instances in their cloud?

2.2.9 Developer impact

Discuss things that will affect other developers working on OpenStack, such as:

- If the blueprint proposes a change to the driver API, discussion of how other hypervisors would implement the feature is required.

2.2.10 Upgrade impact

Describe any potential upgrade impact on the system.

2.3 Implementation

2.3.1 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>

2.3.2 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.

Consider creating an ordering of patches, that allows gradually merging instead of the need to merge them all at once.

2.4 Dependencies

- Include specific references to specs and/or blueprints in blazar, or in other projects, that this one either depends on or is related to.
- If this requires functionality of another project that is not currently used by Blazar, document that fact.
- Does this feature require any new library dependencies or code otherwise not included in OpenStack? Or does it depend on a specific version of library?

2.5 Testing

Please discuss the important scenarios needed to test here, as well as specific edge cases we should be ensuring work correctly. For each scenario please specify if this requires specialized hardware, a full openstack environment, or can be simulated inside the Blazar tree.

Please discuss how the change will be tested. We especially want to know what tempest tests will be added. It is assumed that unit test coverage will be added so that doesnt need to be mentioned explicitly,

but discussion of why you think unit tests are sufficient and we dont need to add more tempest tests would need to be included.

Is this untestable in gate given current limitations (specific hardware / software configurations available)? If so, are there mitigation plans (3rd party testing, gate enhancements, etc).

2.6 Documentation Impact

Which audiences are affected most by this change, and which documentation titles on docs.openstack.org should be updated because of this change? Dont repeat details discussed above, but reference them here in the context of documentation for multiple audiences. For example, the Operations Guide targets cloud operators, and the End User Guide would need to be updated if the change offers a new feature available through the CLI or dashboard. If a config option changes or is deprecated, note here that the documentation needs to be updated to reflect this specifications change.

2.7 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 notes from a summit session
- Links to relevant research, if appropriate
- Related specifications as appropriate
- Anything else you feel it is worthwhile to refer to

2.8 History

Optional section intended to be used each time the spec is updated to describe new design, API or any database schema updated. Useful to let reader understand whats happened along the time.

Table 1: Revisions

Release Name	Description
Yoga	Introduced

CHAPTER

THREE

EXAMPLE SPEC - THE TITLE OF YOUR BLUEPRINT

Include the URL of your launchpad blueprint:

https://blueprints.launchpad.net/blazar/+spec/example

Introduction paragraph why are we doing anything? A single paragraph of prose that operators can understand. The title and this first paragraph should be used as the subject line and body of the commit message respectively.

Some notes about the blazar-spec and blueprint process:

- Not all blueprints need a spec.
- The aim of this document is first to define the problem we need to solve, and second agree the overall approach to solve that problem.
- This is not intended to be extensive documentation for a new feature. For example, there is no need to specify the exact configuration changes, nor the exact details of any DB model changes. But you should still define that such changes are required, and be clear on how that will affect upgrades.
- You should aim to get your spec approved before writing your code. While you are free to write prototypes and code before getting your spec approved, its possible that the outcome of the spec review process leads you towards a fundamentally different solution than you first envisaged.
- But, API changes are held to a much higher level of scrutiny. As soon as an API change merges, we must assume it could be in production somewhere, and as such, we then need to support that API change forever. To avoid getting that wrong, we do want lots of details about API changes upfront.

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/blazar/+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
- For help with syntax, see http://sphinx-doc.org/rest.html
- To test out your formatting, build the docs using tox and see the generated HTML file in doc/build/html/specs/<path_of_your_file>
- 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.

- If your specification proposes any changes to the Blazar REST API such as changing parameters which can be returned or accepted, or even the semantics of what happens when a client calls into the API, then you should add the APIImpact flag to the commit message. Specifications with the APIImpact flag can be found with the following query:

  https://review.opendev.org/#/q/project:openstack/blazar-specs+message:apiimpact,n,z

3.1 Problem description

A detailed description of the problem. What problem is this blueprint addressing?

3.1.1 Use Cases

What use cases does this address? What impact on actors does this change have? Ensure you are clear about the actors in each use case: Developer, End User, Deployer etc.

3.2 Proposed change

Here is where you cover the change you propose to make in detail. How do you propose to solve this problem?

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?

At this point, if you would like to just get feedback on if the problem and proposed change fit in blazar, you can stop here and post this for review to get preliminary feedback. If so please say: Posting to get preliminary feedback on the scope of this spec.

3.2.1 Alternatives

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.

3.2.2 Data model impact

Changes which require modifications to the data model often have a wider impact on the system. The community often has strong opinions on how the data model should be evolved, from both a functional and performance perspective. It is therefore important to capture and gain agreement as early as possible on any proposed changes to the data model.

Questions which need to be addressed by this section include:

- What new data objects and/or database schema changes is this going to require?
- What database migrations will accompany this change.
- How will the initial set of new data objects be generated, for example if you need to take into account existing instances, or modify other existing data describe how that will work.

3.2.3 REST API impact

Each API method which is either added or changed should have the following

- Specification for the method
  - A description of what the method does suitable for use in user documentation
  - Method type (POST/PUT/GET/DELETE)
  - Normal http response code(s)
  - Expected error http response code(s)
    - A description for each possible error code should be included describing semantic errors which can cause it such as inconsistent parameters supplied to the method, or when an instance is not in an appropriate state for the request to succeed. Errors caused by syntactic problems covered by the JSON schema definition do not need to be included.
  - URL for the resource
    - URL should not include underscores, and use hyphens instead.
  - Parameters which can be passed via the url
  - JSON schema definition for the request body data if allowed
    - Field names should use snake_case style, not CamelCase or MixedCase style.
  - JSON schema definition for the response body data if any
    - Field names should use snake_case style, not CamelCase or MixedCase style.
- Example use case including typical API samples for both data supplied by the caller and the response
- Discuss any policy changes, and discuss what things a deployer needs to think about when defining their policy.

Note that the schema should be defined as restrictively as possible. Parameters which are required should be marked as such and only under exceptional circumstances should additional parameters which are not defined in the schema be permitted (eg additionaProperties should be False).

Reuse of existing predefined parameter types such as regexps for passwords and user defined names is highly encouraged.

3.2.4 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 the 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.

- Can this change enable a resource exhaustion attack, such as allowing a single API interaction to consume significant server resources? Some examples of this include launching subprocesses for each connection, or entity expansion attacks in XML.

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.

3.2.5 Notifications impact

Please specify any changes to notifications. Be that an extra notification, changes to an existing notification, or removing a notification.

3.2.6 Other end user impact

Aside from the API, are there other ways a user will interact with this feature?

- Does this change have an impact on python-blazarclient? What does the user interface there look like?

3.2.7 Performance Impact

Describe any potential performance impact on the system, for example how often will new code be called, and is there a major change to the calling pattern of existing code.

Examples of things to consider here include:

- A small change in a utility function or a commonly used decorator can have a large impacts on performance.
- Calls which result in a database queries can have a profound impact on performance when called in critical sections of the code.
- Will the change include any locking, and if so what considerations are there on holding the lock?

3.2.8 Other deployer impact

Discuss things that will affect how you deploy and configure OpenStack that have not already been mentioned, such as:

- What config options are being added? Should they be more generic than proposed (for example a flag that other hypervisor drivers might want to implement as well)? 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 directory name that instances are stored in, how do we handle instance directories created before the change landed? Do we move them? Do we have a special case in the code? Do we assume that the operator will recreate all the instances in their cloud?

3.2.9 Developer impact

Discuss things that will affect other developers working on OpenStack, such as:

- If the blueprint proposes a change to the driver API, discussion of how other hypervisors would implement the feature is required.

3.2.10 Upgrade impact

Describe any potential upgrade impact on the system.

3.3 Implementation

3.3.1 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>

3.3.2 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.

3.4 Dependencies

- Include specific references to specs and/or blueprints in blazar, or in other projects, that this one either depends on or is related to.
- If this requires functionality of another project that is not currently used by Blazar (such as the glance v2 API when we previously only required v1), document that fact.
- Does this feature require any new library dependencies or code otherwise not included in OpenStack? Or does it depend on a specific version of library?

3.5 Testing

Please discuss the important scenarios needed to test here, as well as specific edge cases we should be ensuring work correctly. For each scenario please specify if this requires specialized hardware, a full openstack environment, or can be simulated inside the Blazar tree.

Please discuss how the change will be tested. We especially want to know what tempest tests will be added. It is assumed that unit test coverage will be added so that doesnt need to be mentioned explicitly, but discussion of why you think unit tests are sufficient and we dont need to add more tempest tests would need to be included.

Is this untestable in gate given current limitations (specific hardware / software configurations available)? If so, are there mitigation plans (3rd party testing, gate enhancements, etc).

3.6 Documentation Impact

Which audiences are affected most by this change, and which documentation titles on docs.openstack.org should be updated because of this change? Dont repeat details discussed above, but reference them here in the context of documentation for multiple audiences. For example, the Operations Guide targets cloud operators, and the End User Guide would need to be updated if the change offers a new feature available through the CLI or dashboard. If a config option changes or is deprecated, note here that the documentation needs to be updated to reflect this specifications change.

3.7 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 notes from a summit session
- Links to relevant research, if appropriate
- Related specifications as appropriate (e.g. if its an EC2 thing, link the EC2 docs)
- Anything else you feel it is worthwhile to refer to

3.8 History

Optional section intended to be used each time the spec is updated to describe new design, API or any database schema updated. Useful to let reader understand whats happened along the time.

Table 1: Revisions

Release Name	Description
Xena	Introduced

CHAPTER

FOUR

EXAMPLE SPEC - THE TITLE OF YOUR BLUEPRINT

Include the URL of your launchpad blueprint:

https://blueprints.launchpad.net/blazar/+spec/example

Introduction paragraph why are we doing anything? A single paragraph of prose that operators can understand. The title and this first paragraph should be used as the subject line and body of the commit message respectively.

Some notes about the blazar-spec and blueprint process:

- Not all blueprints need a spec.
- The aim of this document is first to define the problem we need to solve, and second agree the overall approach to solve that problem.
- This is not intended to be extensive documentation for a new feature. For example, there is no need to specify the exact configuration changes, nor the exact details of any DB model changes. But you should still define that such changes are required, and be clear on how that will affect upgrades.
- You should aim to get your spec approved before writing your code. While you are free to write prototypes and code before getting your spec approved, its possible that the outcome of the spec review process leads you towards a fundamentally different solution than you first envisaged.
- But, API changes are held to a much higher level of scrutiny. As soon as an API change merges, we must assume it could be in production somewhere, and as such, we then need to support that API change forever. To avoid getting that wrong, we do want lots of details about API changes upfront.

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/blazar/+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
- For help with syntax, see http://sphinx-doc.org/rest.html
- To test out your formatting, build the docs using tox and see the generated HTML file in doc/build/html/specs/<path_of_your_file>
- 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.

- If your specification proposes any changes to the Blazar REST API such as changing parameters which can be returned or accepted, or even the semantics of what happens when a client calls into the API, then you should add the APIImpact flag to the commit message. Specifications with the APIImpact flag can be found with the following query:

  https://review.opendev.org/#/q/project:openstack/blazar-specs+message:apiimpact,n,z

4.1 Problem description

A detailed description of the problem. What problem is this blueprint addressing?

4.1.1 Use Cases

What use cases does this address? What impact on actors does this change have? Ensure you are clear about the actors in each use case: Developer, End User, Deployer etc.

4.2 Proposed change

Here is where you cover the change you propose to make in detail. How do you propose to solve this problem?

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?

At this point, if you would like to just get feedback on if the problem and proposed change fit in blazar, you can stop here and post this for review to get preliminary feedback. If so please say: Posting to get preliminary feedback on the scope of this spec.

4.2.1 Alternatives

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.

4.2.2 Data model impact

Changes which require modifications to the data model often have a wider impact on the system. The community often has strong opinions on how the data model should be evolved, from both a functional and performance perspective. It is therefore important to capture and gain agreement as early as possible on any proposed changes to the data model.

Questions which need to be addressed by this section include:

- What new data objects and/or database schema changes is this going to require?
- What database migrations will accompany this change.
- How will the initial set of new data objects be generated, for example if you need to take into account existing instances, or modify other existing data describe how that will work.

4.2.3 REST API impact

Each API method which is either added or changed should have the following

- Specification for the method
  - A description of what the method does suitable for use in user documentation
  - Method type (POST/PUT/GET/DELETE)
  - Normal http response code(s)
  - Expected error http response code(s)
    - A description for each possible error code should be included describing semantic errors which can cause it such as inconsistent parameters supplied to the method, or when an instance is not in an appropriate state for the request to succeed. Errors caused by syntactic problems covered by the JSON schema definition do not need to be included.
  - URL for the resource
    - URL should not include underscores, and use hyphens instead.
  - Parameters which can be passed via the url
  - JSON schema definition for the request body data if allowed
    - Field names should use snake_case style, not CamelCase or MixedCase style.
  - JSON schema definition for the response body data if any
    - Field names should use snake_case style, not CamelCase or MixedCase style.
- Example use case including typical API samples for both data supplied by the caller and the response
- Discuss any policy changes, and discuss what things a deployer needs to think about when defining their policy.

Note that the schema should be defined as restrictively as possible. Parameters which are required should be marked as such and only under exceptional circumstances should additional parameters which are not defined in the schema be permitted (eg additionaProperties should be False).

Reuse of existing predefined parameter types such as regexps for passwords and user defined names is highly encouraged.

4.2.4 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 the 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.

- Can this change enable a resource exhaustion attack, such as allowing a single API interaction to consume significant server resources? Some examples of this include launching subprocesses for each connection, or entity expansion attacks in XML.

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.

4.2.5 Notifications impact

Please specify any changes to notifications. Be that an extra notification, changes to an existing notification, or removing a notification.

4.2.6 Other end user impact

Aside from the API, are there other ways a user will interact with this feature?

- Does this change have an impact on python-blazarclient? What does the user interface there look like?

4.2.7 Performance Impact

Describe any potential performance impact on the system, for example how often will new code be called, and is there a major change to the calling pattern of existing code.

Examples of things to consider here include:

- A small change in a utility function or a commonly used decorator can have a large impacts on performance.
- Calls which result in a database queries can have a profound impact on performance when called in critical sections of the code.
- Will the change include any locking, and if so what considerations are there on holding the lock?

4.2.8 Other deployer impact

Discuss things that will affect how you deploy and configure OpenStack that have not already been mentioned, such as:

- What config options are being added? Should they be more generic than proposed (for example a flag that other hypervisor drivers might want to implement as well)? 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 directory name that instances are stored in, how do we handle instance directories created before the change landed? Do we move them? Do we have a special case in the code? Do we assume that the operator will recreate all the instances in their cloud?

4.2.9 Developer impact

Discuss things that will affect other developers working on OpenStack, such as:

- If the blueprint proposes a change to the driver API, discussion of how other hypervisors would implement the feature is required.

4.2.10 Upgrade impact

Describe any potential upgrade impact on the system.

4.3 Implementation

4.3.1 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>

4.3.2 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.

4.4 Dependencies

- Include specific references to specs and/or blueprints in blazar, or in other projects, that this one either depends on or is related to.
- If this requires functionality of another project that is not currently used by Blazar (such as the glance v2 API when we previously only required v1), document that fact.
- Does this feature require any new library dependencies or code otherwise not included in OpenStack? Or does it depend on a specific version of library?

4.5 Testing

Please discuss the important scenarios needed to test here, as well as specific edge cases we should be ensuring work correctly. For each scenario please specify if this requires specialized hardware, a full openstack environment, or can be simulated inside the Blazar tree.

Please discuss how the change will be tested. We especially want to know what tempest tests will be added. It is assumed that unit test coverage will be added so that doesnt need to be mentioned explicitly, but discussion of why you think unit tests are sufficient and we dont need to add more tempest tests would need to be included.

Is this untestable in gate given current limitations (specific hardware / software configurations available)? If so, are there mitigation plans (3rd party testing, gate enhancements, etc).

4.6 Documentation Impact

Which audiences are affected most by this change, and which documentation titles on docs.openstack.org should be updated because of this change? Dont repeat details discussed above, but reference them here in the context of documentation for multiple audiences. For example, the Operations Guide targets cloud operators, and the End User Guide would need to be updated if the change offers a new feature available through the CLI or dashboard. If a config option changes or is deprecated, note here that the documentation needs to be updated to reflect this specifications change.

4.7 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 notes from a summit session
- Links to relevant research, if appropriate
- Related specifications as appropriate (e.g. if its an EC2 thing, link the EC2 docs)
- Anything else you feel it is worthwhile to refer to

4.8 History

Optional section intended to be used each time the spec is updated to describe new design, API or any database schema updated. Useful to let reader understand whats happened along the time.

Table 1: Revisions

Release Name	Description
Victoria	Introduced

CHAPTER

FIVE

SUPPORT PREEMPTIBLE INSTANCES WITH BLAZAR

https://blueprints.launchpad.net/blazar/+spec/blazar-preemptible-instances

This spec proposes to add support for preemptible instances with Blazar. It would allow to increase utilization by filling gaps between reservations, as well as provide an implementation of preemptible instances that does not require modifications to the Nova compute service.

5.1 Problem description

To ensure resources allocated for reservations are not in use when reservations reach their start time, Blazar relies on exclusive management of dedicated pools of resources. In the case of compute resources, compute hosts managed by Blazar are initially moved to a host aggregate named *freepool*. Regular instances which are not launched inside a reservation are not allowed to run on nodes in the *freepool*. This can lead to low utilization if reservations are not making full use of the resources managed by Blazar. This spec proposes to add support for preemptible instances, i.e. instances that can be terminated at any time when required by the infrastructure, to increase utilization by filling gaps between reservations.

This feature would also provide a minimally intrusive implementation of preemptible instances for OpenStack: an alternative implementation based on adding a pending state to Nova has proved difficult to be accepted.

5.1.1 Use Cases

This feature primarily targets deployers who operate Blazar and want to increase the utilization of their infrastructure by filling gaps between user reservations. It would also benefit end users who can run preemptible workloads by providing them with extra resources.

5.2 Proposed change

Due to Blazars design, preemptible instances can be supported with few modifications to the code. The following mechanisms are required:

1. A way to identify different types of instance creation requests and of running instances: regular (non-reserved) instances, reserved instances, and preemptible instances.
2. A modification of the custom Nova scheduler filter, BlazarFilter, to allow instances considered as preemptibles to be launched on hosts located in the freepool and only in the freepool: we do not want preemptibles running on hosts not managed by Blazar or on hosts part of a reservation.
3. Before a lease becomes active, the Blazar compute plugins implementing instance reservation and host reservation would terminate any preemptible instance running on allocated hosts, identifying them using the method previously mentioned.

These mechanisms are described in more details below.

5.2.1 Identification of preemptible instance creation requests

To identify an instance creation request as one for a preemptible instance, some possible methods are:

1. Operators would configure via Blazar static configuration or API call a list of project IDs describes projects for which any instance creation request is treated as one for preemptibles.
    - Pros: can restrict preemptibles to specific projects
    - Cons: cannot allow preemptibles for all projects, unless a special globing pattern is used; may also need a method for projects to requests non-preemptible instances; may be difficult to identify running preemptibles.
2. Via scheduler hints: an instance creation request is identified as being for a preemptible instance if the scheduler hint `preemptible=true` is passed.
    - Pros: can be used by any project without configuration by operator
    - Cons: can be used by any project without configuration by operator; may be difficult to identify running preemptibles.
3. Via flavor properties: a flavor property is used to identify preemptible instances. For example, an existing flavor could be configured to launch preemptible instances by using: `openstack flavor set FLAVOR-NAME --property blazar:preemptible=true`
    - Pros: flavor can be made public or shared only with specific projects; easy to identify running preemptibles.
    - Cons: duplicates the number of flavors if all flavors can potentially be preemptible
4. Via image properties: an image property is used to identify preemptible instances. For example, an existing image could be configured to launch preemptible instances by using: `openstack image set --property blazar:preemptible=true IMAGE_UUID`
    - Pros: easy to identify running preemptibles
    - Cons: any project could use preemptibles, unless policy prevents setting image properties; would need to cover volumes as well; could lead to image duplication.

5.2.2 Identification of running preemptible instances

For compute hosts allocated to host reservations, running preemptible instances are simply instances that are executing on allocated hosts outside of any reservation.

For compute hosts allocated to instance reservation, instances launched as part of a reservation are identified by the flavor they use, which is encoded as `flavor_id` in the reservation information. Blazar flavors also follow a specific naming scheme: `reservation:<reservation_id>`. Instances running on allocated hosts that dont use such flavors would be identified as preemptibles.

5.2.3 Termination of preemptible instances

Preemtible instances need to be terminated before a lease becomes active. The Blazar compute plugins implementing instance reservation and host reservation would terminate any preemptible instance running on allocated hosts, identifying them using the previously mentioned method.

To allow preemptible instances to terminate their workload cleanly, a soft shutdown operation could first be triggered, followed by a hard shutdown (instance deletion). In order to keep the lease start operation

quick, we may want to implement a `before_start_lease` event, similar to the `before_end_lease` event, which would terminate any preemptible instance prior to triggering the `start_lease` event.

However, we would also want to prevent any new preemptible from launching after having sent instance termination requests. If using the `before_start_lease` approach, this could be difficult to implement from BlazarFilter which does not have access to the Blazar database, and thus cannot efficiently check the status of leases and events. API calls to Blazar may be too slow to process large number of hosts going through the filter.

5.2.4 Interaction with quotas

Preemptible instances would count as regular instances for Nova quotas of number of instances, vCPUs, memory, and disk. This means that projects having reached their quota by running preemptible instances would not be able to launch any new regular instance: they would first need to terminate preemptible instances to reduce their usage below the maximum allowed by their quota.

This could be prevented by the operator by configuring dedicated projects to exclusively run preemptible instances, i.e. not mix preemptible and regular instances within the same project.

5.2.5 Alternatives

A native implementation within Nova would be more flexible, as it would allow preemptible instances to run on reserved hosts until they become actively used by regular instances. However, a proposed implementation based on introducing a pending state has proved difficult to be accepted by the community.

5.2.6 Data model impact

None

5.2.7 REST API impact

None

5.2.8 Security impact

Race conditions between the Nova scheduler and the Blazar manager would have to be handled correctly to prevent preemptible instances from launching while a reservation is started. Failure to terminate all preemptible instances would result in reservation owners not being able to fully use their reservation.

5.2.9 Notifications impact

None

5.2.10 Other end user impact

Nova end users would use one of the mechanisms described earlier to request preemptible instances.

5.2.11 Performance Impact

Terminating preemptible instances during `on_start()` would make leases longer to start, particularly if a soft shutdown signal is sent to instances. A solution based on introducing a new event is described earlier in this spec, but it has shortcomings.

5.2.12 Other deployer impact

The following configuration options would be added to blazar.conf:

- [preemptible]/graceful_shutdown_time: time to wait between a soft shutdown request and a terminate request

5.2.13 Developer impact

None

5.2.14 Upgrade impact

None

5.3 Implementation

5.3.1 Assignee(s)

Primary assignee:
priteau

5.3.2 Work Items

1. Proof of concept with host reservation plugin only
2. Support for instance reservation
3. General improvements (e.g. soft shutdown signal)
4. Documentation and Tempest scenario testing

5.4 Dependencies

None

5.5 Testing

A Tempest scenario verifying that preemptible instances can be launched and terminated when a reservation starts must be implemented. This requires a full OpenStack environment. The current DevStack can be used.

5.6 Documentation Impact

Operators and users launching preemptible instances are most affected by this change. The admin and user guides should be updated.

5.7 References

5.8 History

Table 1: Revisions

Release Name	Description
Ussuri	Introduced

CHAPTER

SIX

FLEXIBLE RESERVATION USAGE ENFORCEMENT

Include the URL of your launchpad blueprint:

https://blueprints.launchpad.net/blazar/+spec/flexible-reservation-usage-enforcement

Blazar reservations provide temporary ownership of resources. In the absence of quota imposed on reservations, users may be able to reserve all cloud resources for long periods of time. To avoid resource shortage, operators may want to restrict reservation usage from users. Because clouds have different usage patterns, enforcement policies should be flexible. However, there are some basic rules and policies that would apply to many clouds, such as restrictions on the length of leases.

This specification proposes to enable policy restrictions via a chain of filters to apply to a particular reservation request. By default, Blazar will provide a set of simple default filters. Operators or developers can choose to implement their own custom filters as Python modules. Additionally, Blazar will provide one built-in filter that delegates to an external HTTP service that provides a compatible interface. This specification includes a proposal for that interface. The overall approach is analogous to Novas scheduling filters.

6.1 Problem description

Blazar reservations provide temporary ownership of resources. In the absence of quota imposed on reservations, users may be able to reserve all cloud resources for long periods of time. To avoid resource shortage, operators may want to restrict reservation usage from users. Because clouds have different usage patterns, enforcement policies should be flexible. However, there are some basic rules and policies that would apply to many clouds, such as restrictions on the length of leases; these types of policies ideally should not require much setup from the operator.

6.1.1 Use Cases

To ensure good resource sharing between users, an operator may want to apply policies such as:

- limiting the duration of reservations
- limiting the size (number of hosts, instances, floating IPs) of reservations
- limiting the number of pending or active leases
- more complex policies taking into account compute node type, usage allocations per project, etc.

From the users point of view, the Blazar interface would remain the same. The only difference may be that an error message describes why a reservation operation was denied.

6.2 Proposed change

We propose to add a chain of *enforcement filters* to Blazar. The filters will execute in series, with each filter taking as its input information about the users request (e.g., the lease length, its desired reservations, and the users ID and their project ID, etc.) The filter will either return no value, or it will raise an error with a specific message about why the request failed. If a filter returns no value, the next filter in the chain will have the chance to respond to the input. If no filters raise an error, the users request is allowed.

We will implement two initial enforcement filters: the first will reject any leases with length exceeding a threshold, and the second will delegate to an external HTTP service. Filters that operate against Nova quotas, for example, disallowing more reservations for physical hosts than a projects instance quota allows for, would be future extensions of this pattern.

The enforcement filters will grant or deny reservation operations, specifically lease create and lease update. Lease delete should generally not require an approval, however, the enforcement filters should be informed when a lease ends so that they can respond to early lease termination if desired.

The enforcement filters would thus be called at three specific points in the reservation lifecycle:

1. During lease create, after allocation candidates have been selected, the lease and its reservations and candidate resources will be passed to the set of enforcement filters. If all of the filters pass the request, Blazar will create the allocations and transition the lease and reservations to the PENDING state as usual. If any filter rejects the request, then the lease will transition to the ERROR state, with a message from the failed filter included in the error raised to the user.
2. During lease update, after allocation candidates have been selected, both the prior lease/reservations/allocations and the new proposed values will be passed to the enforcement filters. If all of the filters pass the request, Blazar will perform the update to the lease as usual. If any filter rejects the request, then an error with a message from the failed filter will be raised to the user. The lease will not transition to an ERROR state; it will remain in its prior state.
3. On lease end, pass the lease and its reservations and current allocated resources to the enforcement filters. The filter result in this case does not affect the rest of the lease lifecycle and the lease should continue its on_end actions. This integration point acts more like a notification, and is intended to support budget-based policies that may be affected by a reservations early termination (e.g., by refunding the user/project the unused portion of the lease in a charge up front model.)

6.2.1 Filter API

Lease, reservation, and allocation information will be passed to each filter. For leases and reservations, the subset of user-controllable attributes (e.g., `start_date`, `end_date`, `resource_type` and plugin-specific reservation attributes) will be passed to the filter; UUIDs and other internal bookkeeping attributes such as `created_at` will not be sent. Each allocation object will similarly contain the relevant operator-defined attributes, e.g., `hypervisor_properties` for `physical:host` reservations and `floating_network_id` for `virtual:floatingip` reservations. If extra capabilities are defined for the resource, they will additionally be sent. This allows for operator-defined custom metadata to be tagged to resources and later utilized when determining policy decisions.

Each filter must support the following operations:

- `check_create`: Check that a new lease request can be satisfied. Receives a `context` object and a `lease` object as arguments.
  - The context contains information about the request:

```
{
  "user_id": "c631173e-dec0-4bb7-a0c3-f7711153c06c",
  "project_id": "a0b86a98-b0d3-43cb-948e-00689182efd4",
  "auth_url": "https://api.example.com:5000/v3",
  "region_name": "RegionOne"
}
```

  - The lease object contains the user-definable aspects of the lease and its reservations, and includes the allocations already chosen by Blazar:

```
{
  "start_date": "2020-05-13 00:00",
  "end_time": "2020-05-14 23:59",
  "reservations": [
    {
      "resource_type": "virtual:floatingip",
      "network_id": "external-network-id",
      "amount": 1,
      "allocations": [
        {
          "floating_network_id": "external-network-id",
          "floating_ip_address": "192.168.1.100",
          "id": "7375755f-717e-4883-91c1-06ceba2da96e"
        }
      ]
    },
    {
      "resource_type": "physical:host",
      "min": 1,
      "max": 2,
      "hypervisor_properties": "[]",
      "resource_properties": "[\"==\", \"$availability_zone\", \"az1\"]",
      "allocations": [
        {
          "id": "1",
          "hypervisor_hostname": "32af5a7a-e7a3-4883-a643-828e3f63bf54",
          "extra": {
            "availability_zone": "az1"
          }
        }
      ]
    }
  ]
}
```

- `check_update`: Check that a lease update request can be satisfied. Receives a `context` object and both the leases current state and desired state as arguments. Sending both sets of state relieve the filter from having to look up the information itself.

  Note: a project lease can be created by one user and updated by another. In the update request, the `user_id` is the ID of the user performing the update.

  - The context contains information about the request (see above).
  - The lease (current and requested) states contain the user-definable aspects of the lease and include the allocations chosen by Blazar.

- `on_end`: Notify the filter that a lease is terminating. Receives a `context` object and the leases current state as arguments.
  - The context contains information about the request (see above).
  - The lease contains the user-definable aspects of the lease and include the allocations chosen by Blazar.

6.2.2 MaximumReservationLengthFilter

This filter simply examines the leases `end_time` and `start_time` and rejects the lease if its length exceeds a threshold.

6.2.3 ExternalServiceFilter

This filter delegates the decision for each API to an external HTTP service. The service should adhere to the API-WG HTTP Response Codes guidelines and return the appropriate HTTP status for success versus error responses (as described in the following endpoint specifications.)

A Keystone token for the `blazar` service user will be sent to the usage service via the `X-Auth-Token` header for all endpoints and can be used to verify a requests authenticity (to prevent spoofing or abuse).

External service API

To be compatible with Blazars filter, the HTTP service must provide the following interface:

- `POST /v1/check-create`
  - Check that a new lease request can be satisfied with the usage policies.
  - Normal response code: `204 No Content`
  - Error response codes:
    - `403 Forbidden`: if the lease is determined to not be allowed according to the implemented usage policies. An error message can be returned as JSON in the response body.
  - Request example:

```
{
  "context": {
    "user_id": "c631173e-dec0-4bb7-a0c3-f7711153c06c",
    "project_id": "a0b86a98-b0d3-43cb-948e-00689182efd4",
    "auth_url": "https://api.example.com:5000/v3",
    "region_name": "RegionOne"
  },
  "lease": {
    "start_date": "2020-05-13 00:00",
    "end_time": "2020-05-14 23:59",
    "reservations": [
      {
        "resource_type": "virtual:floatingip",
```

(continues on next page)

(continued from previous page)


```
        "network_id": "external-network-id",
        "amount": 1,
        "allocations": [
          {
            "floating_network_id": "external-network-id",
            "floating_ip_address": "192.168.1.100",
            "id": "7375755f-717e-4883-91c1-06ceba2da96e"
          }
        ]
      },
      {
        "resource_type": "physical:host",
        "min": 1,
        "max": 2,
        "hypervisor_properties": "[]",
        "resource_properties": "[\"==\", \"$availability_zone\", \"az1\"]
↪",
        "allocations": [
          {
            "id": "1",
            "hypervisor_hostname": "32af5a7a-e7a3-4883-a643-828e3f63bf54",
            "extra": {
              "availability_zone": "az1"
            }
          }
        ]
      }
    ]
  }
}
```


  - Error response body example:


```
{
  "message": "Your lease exceeds the maximum length of 24 hours."
}
```


- `POST /v1/check-update`
  - Check that a lease update request can be satisfied with the usage policies.
  - Normal response code: `204 No Content`
  - Error response codes:
    - `403 Forbidden`: if the leases updated state is determined not to be allowed according to the usage policies. An error message can be returned as JSON in the response body.
  - Request example:


```
{
  "context": {
```


(continues on next page)

(continued from previous page)


```
    "user_id": "c631173e-dec0-4bb7-a0c3-f7711153c06c",
    "project_id": "a0b86a98-b0d3-43cb-948e-00689182efd4",
    "auth_url": "https://api.example.com:5000/v3",
    "region_name": "RegionOne"
  },
  "current_lease": {
    "start_date": "2020-05-13 00:00",
    "end_time": "2020-05-14 23:59",
    "reservations": [
      {
        "resource_type": "physical:host",
        "min": 1,
        "max": 2,
        "hypervisor_properties": "[]",
        "resource_properties": "[\"==\", \"$availability_zone\", \"az1\"]
↪",
        "allocations": [
          {
            "id": "1",
            "hypervisor_hostname": "32af5a7a-e7a3-4883-a643-828e3f63bf54",
            "extra": {
              "availability_zone": "az1"
            }
          }
        ]
      }
    ]
  },
  "lease": {
    "start_date": "2020-05-13 00:00",
    "end_time": "2020-05-14 23:59",
    "reservations": [
      {
        "resource_type": "physical:host",
        "min": 2,
        "max": 3,
        "hypervisor_properties": "[]",
        "resource_properties": "[\"==\", \"$availability_zone\", \"az1\"]
↪",
        "allocations": [
          {
            "id": "1",
            "hypervisor_hostname": "32af5a7a-e7a3-4883-a643-828e3f63bf54",
            "extra": {
              "availability_zone": "az1"
            }
          },
          {
            "id": "2",
```


(continues on next page)

(continued from previous page)

```
            "hypervisor_hostname": "af69aabd-8386-4053-a6dd-1a983787bd7f",
            "extra": {
              "availability_zone": "az1"
            }
          }
        ]
      }
    ]
  }
}
```

  - Error response body example:

```
{
  "message": "Your project is limited to reserving 1 physical host."
}
```

- `POST /v1/on-end`
  - Notify the usage service that a lease is terminating.
  - Normal response code: `204 No Content`
  - Error response codes: None
  - Request example:

```
{
  "context": {
    "user_id": "c631173e-dec0-4bb7-a0c3-f7711153c06c",
    "project_id": "a0b86a98-b0d3-43cb-948e-00689182efd4",
    "auth_url": "https://api.example.com:5000/v3",
    "region_name": "RegionOne"
  },
  "lease": {
    "start_date": "2020-05-13 00:00",
    "end_time": "2020-05-14 23:59",
    "reservations": [
      {
        "resource_type": "physical:host",
        "min": 1,
        "max": 2,
        "hypervisor_properties": "[]",
        "resource_properties": "[\"==\", \"$availability_zone\", \"az1\"]
↪",
        "allocations": [
          {
            "id": "1",
            "hypervisor_hostname": "32af5a7a-e7a3-4883-a643-828e3f63bf54",
            "extra": {
              "availability_zone": "az1"
            }
```

(continues on next page)

(continued from previous page)

```
            }
        ]
    }
]
}
}
```

6.2.4 Alternatives

This approach encompasses several competing designs, notably including standard policies within Blazar (addressed by the inclusion of default filters that address common use-cases) and Python module plugins that provide filters (addressed by the ability to provide custom plugins similarly to the mechanism used for Nova scheduler filters).

An alternative is to simplify the approach to only allow one method of use/extension. However, this limits the functionalitys future utility due to additional constraints placed on the operator/developer.

6.2.5 Data model impact

None

6.2.6 REST API impact

None

6.2.7 Security impact

Information about users, projects, and their usage will be sent to an external web service. If the communication with this service isnt secure, an attacker may be able to capture this information, approve reservation operations which should have been denied, and deny operations which should have been approved. An attacker may also be able to create bogus requests that nonetheless cause real usage charges to accrue on the target account; for this reason, the service should be exposed internally on the same service as Blazar, or should at minimum check the `X-Auth-Token` header contains a valid token for the `blazar` service account.

A high level of reservation operations to Blazar would create a similar level of calls to the policy web service, which may cause a denial of service.

6.2.8 Notifications impact

None

6.2.9 Other end user impact

From an end user point of view, the Blazar interface would remain the same. The only difference may be that an error message describes why a reservation operation was denied.

6.2.10 Performance Impact

Reservation operations can trigger synchronous calls to other services when evaluating enforcement filters, which may cause a slowdown of blazar-manager RPC replies, and thus a slowdown of API responses. Monitoring the performance of the external enforcement service is important, when one is used.

6.2.11 Other deployer impact

A new configuration section `[enforcement]` would be added, with one initial property:

- `available_filters`: a list of modules that provide enforcement filters. Can be specified multiple times. Defaults to `blazar.enforcement.filters.all_filters`
- `enabled_filters`: a comma-separated list of filter names, in the order that they will execute. Defaults to `MaximumReservationLengthFilter,ExternalServiceFilter`
- `reservation_max_length`: the maximum length, in seconds, of a reservation. This is used by the `MaximumReservationLengthFilter`. Defaults to 0, meaning there is no limit, effectively disabling this filter.
- `exempted_projects`: a list of project names or IDs that are not subject to any enforcement filter. If project names are used, the domain name must be provided, e.g. `my-project@Default` in the case of the Default domain.

Additionally, a new configuration section `[enforcement_external]` would be added for the `ExternalServiceFilter`, with the following properties:

- `endpoint_url`: if set to a URL, enables enforcement of reservation usage policies. Defaults to None, which effectively disables this filter.
- `allow_on_error`: if the external enforcement service responds with an HTTP error code or an error occurs elsewhere in the transport layer, allow the reservation request. Defaults to False.

6.2.12 Developer impact

Developer wanting to test reservation usage enforcement will need to deploy an extra web service. We may want to package an example one with Blazar.

6.2.13 Upgrade impact

None

6.3 Implementation

6.3.1 Assignee(s)

Primary assignee:
diurnalist

6.3.2 Work Items

- Implement enforcement filter mechanism (default pass all requests)
- Implement filter that checks for maximum reservation length
- Implement filter for external enforcement service
- Add scenario test to blazar-tempest-plugin
- Update Blazar documentation

6.4 Dependencies

None

6.5 Testing

Unit tests and, if possible, a scenario test should be implemented.

6.6 Documentation Impact

The configuration documentation will need to be updated with the new configuration option. Calls to the policy web service should also be documented so that operators can write their own services.

6.7 References

1. IRC meeting discussion: http://eavesdrop.openstack.org/meetings/blazar/2019/blazar.2019-05-23-16.00.log.html
2. Nova scheduler filters: https://docs.openstack.org/nova/latest/user/filter-scheduler.html

6.8 History

Table 1: Revisions

Release Name	Description
Ussuri	Introduced

CHAPTER

SEVEN

NETWORK RESERVATION

https://blueprints.launchpad.net/blazar/+spec/basic-network-plugin

This plugin supports reserving network segments.

7.1 Problem description

In OpenStack deployments with switches that support VLAN tagging, the Neutron service can create isolated network segments by associating VLAN tags with Neutron networks on a first come first serve basis. Possible VLAN tags are limited to a 12-bit field to comply with 802.1Q and, in many cases, only a small subset of the 4,096 possible VLAN tags can be made available to users. Likewise, if the cloud provider makes external Layer-2 connections (stitching) available, users can only employ these connections using a finite number of available network segments.

7.1.1 Use Cases

- A user requires more network isolation for either research or additional security.
- A user wants to make use of an external layer 2 connection and requires a VLAN tag dedicated to a provider endpoint.
- A user plans to create an isolated VLAN and wants to make sure a VLAN tag is available at the reservation start time.
- A cloud admin does not have enough VLAN tags to give to all users.
  - The admin wants to give a user an isolated VLAN with either host or instance reservation.

7.2 Proposed change

Blazar enables users to reserve network VLAN tags (and, by extension, other segmentation types) by specifying the physical network type the users want to reserve. Users can treat the reserved network as usual, except for the create network operation.

A basic idea for the network reservation and its scenario are as follows:

1. The admin registers some VLAN tags used for network reservation with Blazar. The admin calls Blazars network API with a request body which includes the physical network name, the network type, and the segment ID.
2. A user calls the create lease API specifying a network reservation and the start and end dates of the lease. Optional parameters would include segment ID, network name, and network type. Blazar checks the availability of a network segment for the request. If available, Blazar creates an

allocation between network segment and the reservation, then returns the reservation ID. If not, Blazar doesnt return a reservation ID.

3. At the start time, Blazar creates the reserved network in the users tenant (project). The user can then create, configure, or associate subnet and router to network as usual.
4. At the end time, Blazar deletes the network and any other associated network components such as subnets, router, ports, etc., if the user hasnt deleted or disassociated them already.

7.2.1 Alternatives

None

7.2.2 Data model impact

The plugin introduces four new tables, network_reservations, networksegment_extra_capabilities, network_allocations, network_segments.

The network_reservations table keeps user request information for their network reservation. The role of this table is similar to the role of the computehost_reservations table in the host reservation plugin. This table has id, resource_properties, network_properties, network_name, network_description and network_id columns.

The networksegment_extra_capabilities can contain additional informations about particular network segments that a cloud provider wants to provide.

The network_allocations table has the relationship between the network_reservations table and the network_segments table.

The network_segments table stores information about network segments themselves. The reservable network segments are registered in the table.

The table definitions are as follows:

```
CREATE TABLE `network_reservations` (
    `created_at` datetime DEFAULT NULL,
    `updated_at` datetime DEFAULT NULL,
    `deleted_at` datetime DEFAULT NULL,
    `deleted` varchar(36) DEFAULT NULL,
    `id` varchar(36) NOT NULL,
    `reservation_id` varchar(36) DEFAULT NULL,
    `resource_properties` mediumtext,
    `network_properties` mediumtext,
    `before_end` varchar(36) DEFAULT NULL,
    `network_name` varchar(255) DEFAULT NULL,
    `network_description` varchar(255) DEFAULT NULL,
    `network_id` varchar(255) DEFAULT NULL,
    PRIMARY KEY (`id`),
    KEY `reservation_id` (`reservation_id`),
    CONSTRAINT `network_reservations_ibfk_1`
    FOREIGN KEY (`reservation_id`) REFERENCES `reservations` (`id`)
);

CREATE TABLE `network_segments` (
    `created_at` datetime DEFAULT NULL,
```

(continues on next page)

(continued from previous page)

```
    `updated_at` datetime DEFAULT NULL,
    `id` varchar(36) NOT NULL,
    `network_type` varchar(255) NOT NULL,
    `physical_network` varchar(255) DEFAULT NULL,
    `segmentation_id` int(11),
    `reservable` boolean NOT NULL,
    PRIMARY KEY (`id`),
    UNIQUE KEY `network_type`
    (`network_type`,`physical_network`,`segmentation_id`)
);

CREATE TABLE `networksegment_extra_capabilities` (
    `created_at` datetime DEFAULT NULL,
    `updated_at` datetime DEFAULT NULL,
    `id` varchar(36) NOT NULL,
    `network_id` varchar(36) NOT NULL,
    `capability_name` varchar(64) NOT NULL,
    `capability_value` mediumtext NOT NULL,
    PRIMARY KEY (`id`),
    KEY `network_id` (`network_id`),
    CONSTRAINT `networksegment_extra_capabilities_ibfk_1`
    FOREIGN KEY (`network_id`) REFERENCES `network_segments` (`id`)
);


CREATE TABLE `network_allocations` (
    `created_at` datetime DEFAULT NULL,
    `updated_at` datetime DEFAULT NULL,
    `deleted_at` datetime DEFAULT NULL,
    `deleted` varchar(36) DEFAULT NULL,
    `id` varchar(36) NOT NULL,
    `network_id` varchar(36) DEFAULT NULL,
    `reservation_id` varchar(36) DEFAULT NULL,
    PRIMARY KEY (`id`),
    KEY `network_id` (`network_id`),
    KEY `reservation_id` (`reservation_id`),
    CONSTRAINT `network_allocations_ibfk_1`
    FOREIGN KEY (`network_id`) REFERENCES `network_segments` (`id`),
    CONSTRAINT `network_allocations_ibfk_2`
    FOREIGN KEY (`reservation_id`) REFERENCES `reservations` (`id`)
);
```

7.2.3 REST API impact

The network segment reservation introduces a new resource_type to the lease APIs and five new admin APIs to manage the segments.

Changes in the lease APIs

- URL: POST /v1/leases
  - Introduced new resource_type, network, for a reservation.

Request Example:

```
{
  "name": "network-reservation-1",
  "reservations": [
    {
      "resource_type": "network",
      "network_name": "my-network-1",
      "physical_network": "physnet-name", # optional
      "segmentation_id": 1234,            # optional
      "network_type": "vlan"              # optional
    }
  ],
  "start_date": "2019-05-17 09:07",
  "end_date": "2019-05-17 09:10",
  "events": []
}
```

Response Example

```
{
  "lease": {
    "name": "network-reservation-1"
    "reservations": [
      {
        "id": "reservation-id",
        "status": "pending",
        "lease_id": "lease-id-1",
        "resource_id": "resource_id",
        "resource_type": "network",
        "created_at": "2019-05-17 10:00:00",
        "updated_at": "2017-05-01 11:00:00",
      }],
    "start_date": "2019-05-17 09:07",
    "end_date": "2019-05-17 09:10",
    ..snip..
  }
}
```

- URL: GET /v1/leases
- URL: GET /v1/leases/{lease-id}
- URL: PUT /v1/leases/{lease-id}
- URL: DELETE /v1/leases/{lease-id}
- The change is the same as POST /v1/leases

New network APIs

The five new APIs are admin APIs by default.

- URL: POST /v1/networks
  - The segmentation_id is a specific VLAN tag the admin wants to add. The tag must be out of allocations pools in Neutron. As of now, only networks making use of VLAN tagging require a segment id per IEEE 802.1Q. A reservable flat network would leave this field empty.
  - The network_type is the type of physical mechanism associated with the network segment. Examples include flat, geneve, gre, local, vlan, vxlan.
  - They physical_network is the name of the physical network in which the network segment is available. This is required for VLANs.

Request Example:

```
{
  "network_type": "vlan",
  "physical_network": "physical-network-1",
  "segmentation_id": 1234
}
```

- The reservable key is a flag describing if the network segment is reservable or not. The flag is always True until the network plugin supports the resource healing feature. (Supporting resource healing to network segments is out of scope in this spec)

Response Example:

```
{
  "network": {
      "id": "network-id",
      "network_type": "vlan",
      "physical_network": "physical-network-1",
      "segmentation_id": 1234,
      "reservable": true,
      "created_at": "2020-01-01 10:00:00",
      "updated_at": null
  }
}
```

- URL: GET /v1/networks

Response Example:

```
{
  "networks": [
      {
          "id": "network-id",
          "network_type": "vlan",
          "physical_network": "physical-network-1",
          "segmentation_id": 1234,
          "reservable": true,
          "created_at": "2020-01-01 10:00:00",
```

(continues on next page)

(continued from previous page)

```
            "updated_at": null
        }
    ]
}
```

- URL: GET /v1/networks/{network-id}

Response Example:

```
{
    "network": {
        "id": "network-id",
        "network_type": "vlan",
        "physical_network": "physical-network-1",
        "segmentation_id": "1234",
        "reservable": true,
        "created_at": "2020-01-01 10:00:00",
        "updated_at": null
    }
}
```

- URL: DELETE /v1/networks/{network-id}

No Request body or Response body.

- URL: PUT /v1/networks/{network-id}

  Request Example:

```
{
    "extra_capability_sample": "bar"
}

Response Example:

{
    "network": {
        "id": "network-id",
        "network_type": "vlan",
        "physical_network": "physical-network-1",
        "segmentation_id": "1234",
        "reservable": true,
        "created_at": "2020-01-01 10:00:00",
        "updated_at": null,
        "extra_capability_sample": "bar"
    }
}
```

7.2.4 Security impact

None

7.2.5 Notifications impact

None

7.2.6 Other end user impact

A user can reserve a network segment as well as host, instance or floating IP in the same lease.

python-blazarclient will support the network segment reservation.

7.2.7 Performance Impact

None

7.2.8 Other deployer impact

None

7.2.9 Developer impact

None

7.2.10 Upgrade impact

Some configuration options for the Neutron util class will be introduced to blazar.conf. If the cloud admin want to activate the network reservation, they will need to set up the configuration.

7.3 Implementation

7.3.1 Assignee(s)

Primary assignee:
priteau

Other contributors:
diurnalist jakecoll

7.3.2 Work Items

- Create the new DB tables
- Create the network reservation plugin
- Create the network API object and its route in blazar.api.v1
- Add network reservation support in python-blazarclient
- Add scenario tests and API tests in blazar-tempest-plugin
- Update Blazar docs, API reference and user guide

7.4 Dependencies

None

7.5 Testing

API tests and scenario tests need to be implemented.

7.6 Documentation Impact

This BP adds new APIs and resource type to the lease APIs. The API reference and the Blazar documentation need to be updated.

7.7 References

1. https://etherpad.openstack.org/p/network-resource-reservation
2. https://etherpad.openstack.org/p/blazar-ptg-stein

7.8 History

Table 1: Revisions

Release Name	Description
Train	Introduced
Ussuri	Re-proposed

CHAPTER

EIGHT

RESOURCE PROPERTIES DISCOVERY API

https://blueprints.launchpad.net/blazar/+spec/resource-properties-discovery-api

We propose to add an API that allows users to retrieve a list of all possible resource properties: which keys are supported for each resource type, and what range of values are valid for those keys. This will allow users to craft resource requests with more precision, and allows more advanced workflows in tooling and the Horizon dashboard.

8.1 Problem description

Currently, only operators are given permission to enumerate resources registered in Blazar. Users are allowed to make lease requests and can provide resource constraints, yet have no way of identifying what constraints are valid/possible. Providing some mechanism that allows users to understand which resource keys and values are allowed can improve the user experience around reservations.

8.1.1 Use Cases

- As a user, I know I can reserve resources based on resource properties. I want to know what resource properties exist for my target pool of resources so I can utilize this functionality effectively.
- As an operator, I want users to be able to automatically discover changes I make to Blazar resources without announcing them or otherwise updating a separate set of documentation.
- As a CLI user, I want to be able to retrieve a list of resource properties so I can make future valid CLI requests for reservations.

8.2 Proposed change

We propose adding a new REST API endpoint that, for each resource type, returns a list of resource properties and their possible values. This API can be exposed to users to query directly.

8.2.1 Alternatives

Instead of exposing a new API to users, a better user interface can be implemented in the blazar-dashboard plugin. An admin token can fetch a list of all resources and the properties can be read from there and presented to the user in a number of ways. However, the general pattern with Horizon is that the users token is used for all requests, and adding admin tokens would require configuring Horizon with the Blazar service user credentials.

Another alternative is to allow users to fetch the Blazar resources themselves and do their own inspection. By default, only admins are allowed to list and view resources. Operators are able to loosen this restriction, though this risks exposing sensitive data about the resource accidentally. Additionally, this workflow is less intuitive and doesnt scale well when comparing a large amount of resources.

8.2.2 Data model impact

Extra capabilities will gain an additional property, `private`, which marks them as non-enumerable. This allows operators to add sensitive metadata to resources without worrying about exposing them to users.

- The notion of private capabilities has not existed prior, and assumes that a given property will be either private for all resources, or public for all. Setting some resources to have a private value and others a public value will not be supported by the API. However, the existing data model theoretically supports it, as capabilities are not stored in a normal form (they are simply rows with a given key, value, and attached resource ID).
- To ensure that the private constraint holds, a new table will be created:

```
CREATE TABLE extra_capabilities (
  id VARCHAR(36) NOT NULL,
  resource_type VARCHAR(255) NOT NULL,
  capability_name VARCHAR(255) NOT NULL,
  private BOOLEAN NOT NULL,

  PRIMARY key (id),
  UNIQUE INDEX (resource_type, capability_name),
);
```

- A migration will be included that creates `extra_capability` entries for all existing capabilities, and then updates the extra_capability tables for any resources that implement this feature, removing the capability_name column and replacing it with capability_id.
- Normalizing the extra capabilities means the constraint lookups for existing resources will require joining against the extra_capabilities table.

8.2.3 REST API impact

Because the space of properties and their valuesets for a given resource type may be quite large, this functionality will be broken into two endpoints: one to list all possible keys, and another to list possible values for each key.

- `GET /v1/<resource_type>/properties`
  - Returns a list of public properties supported by the resource type.
  - Normal response code: `200 OK`
  - Request query parameters:
    - `detail` (optional boolean): whether to return the list of possible values for each property in addition to the property name. If present, an additional `values` key will be returned with each property item, with a list of all the values, similar to how they are returned in the endpoint that returns values for a given property name.
  - Error response code(s):
    - `404 Not Found`: if the resource type is not valid.
  - Example response (for `physical:host` resource type):

```
[
  {
```

(continues on next page)

(continued from previous page)

```
    "property": "local_gb"
  },
  {
    "property": "memory_mb"
  },
  {
    "property": "custom_capabilities.first"
  },
  {
    "property": "custom_capabilities.second"
  }
]
```

- `GET /v1/<resource_type>/properties/<property_name>`
  - Returns a list of valid values for the property name.
  - Normal response code: `200 OK`
  - Error response code(s):
    - `403 Forbidden`: if the property is marked private, but is requested with a non-admin token.
    - `404 Not Found`: if the property name is not valid for the resource type. A message indicating this will be returned in the response body.
  - Example response (for an `arch` property of `physical:host` resources):

```
{
  "private": false,
  "values": [
    {
      "value": "x86"
    },
    {
      "value": "arm"
    }
  ]
}
```

- `PATCH /v1/<resource_type>/properties/<property_name>`
  - Updates a given property with some metadata. Currently the `private` option is the only supported.
  - Normal response code: `204 No Content`
  - Error response code(s):
    - `403 Forbidden`: if the policy disallows `put:extra_capability` for the requesting user.
    - `404 Not Found`: if the property name is not valid for the resource type. A message indicating this will be returned in the response body.

– Example request:

```
{
  "private": true
}
```

8.2.4 Security impact

All capabilities will be private by default, which should reduce the likelihood of exposure. By default, future capabilities will also be private. Operators will be able to override this behavior via the `something` flag.

We could improve security at the expense of operator experience if we require that capabilities are explicitly registered before being attached to a resource, instead of creating new capabilities on-demand when a resource is updated to have a property that has not yet been seen for the resource type.

8.2.5 Notifications impact

None

8.2.6 Other end user impact

The Blazar CLI client will be updated to support the following functions:

- `host-capability-list`: lists all properties for physical:host resources
- `host-capability-get <name>`: gets all values for a given host capability
- `host-capability-set <name> [--private|--public]`: updates the visibility for a given host capability
- If we want to enforce creating capabilities explicitly: `host-capability-create <name> [--private|--public]`: creates a new host capability with given visibility settings.

8.2.7 Performance Impact

The normalization of the capabilities into a new table introduces an additional join when looking up resources for allocation candidates, which could add time to that operation. In practice this overhead should be low, as the space of the capability table will be quite small, and an index will be used over the resource type and capability name, which would be the two columns needed by the allocation selection.

8.2.8 Other deployer impact

This new API will default to only be accessible to operators to mitigate risk of accidentally exposing properties to users. Operators will have to opt-in by allowing all users to access the API, after making sure that any sensitive capabilities they have attached to resources are made private. We may elect to change the default visibility for the API in a future release.

- `[DEFAULT] capability_default_visibility`: Defines how new capabilities added to resources are exposed for end-users. Defaults to `private`, meaning users will not be able to enumerate the property. Operators can set this to `public` to instead default capabilities to public visibility.

8.2.9 Developer impact

None

8.2.10 Upgrade impact

A database migration will be required to add the new column to the extra capability tables.

8.3 Implementation

8.3.1 Assignee(s)

Primary assignee:
jasonandersonatuchicago

8.3.2 Work Items

- Create REST API endpoints for retrieving keys/value sets
- Create database migration for existing extra capabilities
- Update Blazar CLI to support discovering resource properties
- Update Blazar CLI to support updating resource properties as private

8.4 Dependencies

None

8.5 Testing

Unit tests will be written, and a scenario test for the new API endpoints, if the supporting infrastructure for these tests is in place.

8.6 Documentation Impact

REST API documentation will need to be updated to include these new endpoints. A release note will indicate the new API and the ability it grants users.

8.7 References

- IRC discussion: http://eavesdrop.openstack.org/meetings/blazar/2020/blazar.2020-02-27-16.00.log.html

8.8 History

Optional section intended to be used each time the spec is updated to describe new design, API or any database schema updated. Useful to let reader understand whats happened along the time.

Table 1: Revisions

Release Name	Description
Ussuri	Introduced

CHAPTER

NINE

EXAMPLE SPEC - THE TITLE OF YOUR BLUEPRINT

Include the URL of your launchpad blueprint:

https://blueprints.launchpad.net/blazar/+spec/example

Introduction paragraph why are we doing anything? A single paragraph of prose that operators can understand. The title and this first paragraph should be used as the subject line and body of the commit message respectively.

Some notes about the blazar-spec and blueprint process:

- Not all blueprints need a spec.
- The aim of this document is first to define the problem we need to solve, and second agree the overall approach to solve that problem.
- This is not intended to be extensive documentation for a new feature. For example, there is no need to specify the exact configuration changes, nor the exact details of any DB model changes. But you should still define that such changes are required, and be clear on how that will affect upgrades.
- You should aim to get your spec approved before writing your code. While you are free to write prototypes and code before getting your spec approved, its possible that the outcome of the spec review process leads you towards a fundamentally different solution than you first envisaged.
- But, API changes are held to a much higher level of scrutiny. As soon as an API change merges, we must assume it could be in production somewhere, and as such, we then need to support that API change forever. To avoid getting that wrong, we do want lots of details about API changes upfront.

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/blazar/+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
- For help with syntax, see http://sphinx-doc.org/rest.html
- To test out your formatting, build the docs using tox and see the generated HTML file in doc/build/html/specs/<path_of_your_file>
- 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.

- If your specification proposes any changes to the Blazar REST API such as changing parameters which can be returned or accepted, or even the semantics of what happens when a client calls into the API, then you should add the APIImpact flag to the commit message. Specifications with the APIImpact flag can be found with the following query:

  https://review.opendev.org/#/q/project:openstack/blazar-specs+message:apiimpact,n,z

9.1 Problem description

A detailed description of the problem. What problem is this blueprint addressing?

9.1.1 Use Cases

What use cases does this address? What impact on actors does this change have? Ensure you are clear about the actors in each use case: Developer, End User, Deployer etc.

9.2 Proposed change

Here is where you cover the change you propose to make in detail. How do you propose to solve this problem?

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?

At this point, if you would like to just get feedback on if the problem and proposed change fit in blazar, you can stop here and post this for review to get preliminary feedback. If so please say: Posting to get preliminary feedback on the scope of this spec.

9.2.1 Alternatives

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.

9.2.2 Data model impact

Changes which require modifications to the data model often have a wider impact on the system. The community often has strong opinions on how the data model should be evolved, from both a functional and performance perspective. It is therefore important to capture and gain agreement as early as possible on any proposed changes to the data model.

Questions which need to be addressed by this section include:

- What new data objects and/or database schema changes is this going to require?
- What database migrations will accompany this change.
- How will the initial set of new data objects be generated, for example if you need to take into account existing instances, or modify other existing data describe how that will work.

9.2.3 REST API impact

Each API method which is either added or changed should have the following

- Specification for the method
  - A description of what the method does suitable for use in user documentation
  - Method type (POST/PUT/GET/DELETE)
  - Normal http response code(s)
  - Expected error http response code(s)
    - A description for each possible error code should be included describing semantic errors which can cause it such as inconsistent parameters supplied to the method, or when an instance is not in an appropriate state for the request to succeed. Errors caused by syntactic problems covered by the JSON schema definition do not need to be included.
  - URL for the resource
    - URL should not include underscores, and use hyphens instead.
  - Parameters which can be passed via the url
  - JSON schema definition for the request body data if allowed
    - Field names should use snake_case style, not CamelCase or MixedCase style.
  - JSON schema definition for the response body data if any
    - Field names should use snake_case style, not CamelCase or MixedCase style.
- Example use case including typical API samples for both data supplied by the caller and the response
- Discuss any policy changes, and discuss what things a deployer needs to think about when defining their policy.

Note that the schema should be defined as restrictively as possible. Parameters which are required should be marked as such and only under exceptional circumstances should additional parameters which are not defined in the schema be permitted (eg additionaProperties should be False).

Reuse of existing predefined parameter types such as regexps for passwords and user defined names is highly encouraged.

9.2.4 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 the 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.

- Can this change enable a resource exhaustion attack, such as allowing a single API interaction to consume significant server resources? Some examples of this include launching subprocesses for each connection, or entity expansion attacks in XML.

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.

9.2.5 Notifications impact

Please specify any changes to notifications. Be that an extra notification, changes to an existing notification, or removing a notification.

9.2.6 Other end user impact

Aside from the API, are there other ways a user will interact with this feature?

- Does this change have an impact on python-blazarclient? What does the user interface there look like?

9.2.7 Performance Impact

Describe any potential performance impact on the system, for example how often will new code be called, and is there a major change to the calling pattern of existing code.

Examples of things to consider here include:

- A small change in a utility function or a commonly used decorator can have a large impacts on performance.
- Calls which result in a database queries can have a profound impact on performance when called in critical sections of the code.
- Will the change include any locking, and if so what considerations are there on holding the lock?

9.2.8 Other deployer impact

Discuss things that will affect how you deploy and configure OpenStack that have not already been mentioned, such as:

- What config options are being added? Should they be more generic than proposed (for example a flag that other hypervisor drivers might want to implement as well)? 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 directory name that instances are stored in, how do we handle instance directories created before the change landed? Do we move them? Do we have a special case in the code? Do we assume that the operator will recreate all the instances in their cloud?

9.2.9 Developer impact

Discuss things that will affect other developers working on OpenStack, such as:

- If the blueprint proposes a change to the driver API, discussion of how other hypervisors would implement the feature is required.

9.2.10 Upgrade impact

Describe any potential upgrade impact on the system.

9.3 Implementation

9.3.1 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>

9.3.2 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.

9.4 Dependencies

- Include specific references to specs and/or blueprints in blazar, or in other projects, that this one either depends on or is related to.
- If this requires functionality of another project that is not currently used by Blazar (such as the glance v2 API when we previously only required v1), document that fact.
- Does this feature require any new library dependencies or code otherwise not included in OpenStack? Or does it depend on a specific version of library?

9.5 Testing

Please discuss the important scenarios needed to test here, as well as specific edge cases we should be ensuring work correctly. For each scenario please specify if this requires specialized hardware, a full openstack environment, or can be simulated inside the Blazar tree.

Please discuss how the change will be tested. We especially want to know what tempest tests will be added. It is assumed that unit test coverage will be added so that doesnt need to be mentioned explicitly, but discussion of why you think unit tests are sufficient and we dont need to add more tempest tests would need to be included.

Is this untestable in gate given current limitations (specific hardware / software configurations available)? If so, are there mitigation plans (3rd party testing, gate enhancements, etc).

9.6 Documentation Impact

Which audiences are affected most by this change, and which documentation titles on docs.openstack.org should be updated because of this change? Dont repeat details discussed above, but reference them here in the context of documentation for multiple audiences. For example, the Operations Guide targets cloud operators, and the End User Guide would need to be updated if the change offers a new feature available through the CLI or dashboard. If a config option changes or is deprecated, note here that the documentation needs to be updated to reflect this specifications change.

9.7 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 notes from a summit session
- Links to relevant research, if appropriate
- Related specifications as appropriate (e.g. if its an EC2 thing, link the EC2 docs)
- Anything else you feel it is worthwhile to refer to

9.8 History

Optional section intended to be used each time the spec is updated to describe new design, API or any database schema updated. Useful to let reader understand whats happened along the time.

Table 1: Revisions

Release Name	Description
Ussuri	Introduced

CHAPTER

TEN

API VALIDATION USING JSONSCHEMA

https://blueprints.launchpad.net/blazar/+spec/json-schema-validation

Currently, Blazar has different implementations for validating request bodies. The purpose of this blueprint is to validate the request body in the API layer using json schema validation before it is forwarded to any of the other Blazar components. Validating the request bodies sent to the Blazar server, accepting requests that fit the resource schema and rejecting requests that do not fit the schema. Depending on the content of the request body, the request should be accepted or rejected.

10.1 Problem description

Currently Blazar doesnt have a consistent request validation layer. Some resources validate input at the resource controller and some fail out in the backend. Ideally, Blazar would have some validation in place to catch disallowed parameters and return a validation error to the user.

The end user will benefit from having consistent and helpful feedback, regardless of which resource they are interacting with.

10.2 Use Cases

As a user or developer, I want to observe consistent API validation and values passed to the Blazar API server.

10.3 Proposed change

One possible way to validate the Blazar API is to use jsonschema similar to Nova, Keystone and Glance (https://pypi.python.org/pypi/jsonschema). A jsonschema validator object can be used to check each resource against an appropriate schema for that resource. If the validation passes, the request can follow the existing flow of control through the resource manager to the backend. If the request body parameters fails the validation specified by the resource schema, a validation error wrapped in HTTPBadRequest will be returned from the server.

10.3.1 Alternatives

Before the API validation framework, we needed to add the validation code into each API method in ad-hoc. These changes would make the API method code dirty and we need to create multiple patches due to incomplete validation.

If using JSON Schema definitions instead, acceptable request formats are clear and we dont need to do ad-hoc works in the future.

10.3.2 Data model impact

None

10.3.3 REST API impact

API Response code changes:

There are some occurrences where API response code will change while adding schema layer for them. For example, On current master computehosts table has hypervisor_hostname of maximum 255 characters in database table. While creating host user can pass host of more than 255 characters which obviously fails with 404 HostNotFound wasting a database call. For this we can restrict the host of maximum 255 characters only in schema definition of os-hosts. If user passes more than 255 characters, he/she will get 400 BadRequest in response.

API Response error messages:

There will be change in the error message returned to user. For example, On current master if user passes nothing for host name like name: then below error message is returned to user from blazar-api:

Host not found!.

With schema validation below error message will be returned to user for this case:

Invalid input for field/attribute name. Value: <value passed by user>. <value passed by user> is too short.

10.3.4 Security impact

The output from the request validation layer should not compromise data or expose private data to an external user. Request validation should not return information upon successful validation. In the event a request body is not valid, the validation layer should return the invalid values and/or the values required by the request, of which the end user should know. The parameters of the resources being validated are public information, described in the Blazar API reference document with the exception of private data. In the event the users private data fails validation, a check can be built into the error handling of the validator not to return the actual value of the private data.

Jsonschema documentation notes security considerations for both schemas and instances.

Better up front input validation will reduce the ability for malicious user input to exploit security bugs.

10.3.5 Notifications impact

None

10.3.6 Other end user impact

None

10.3.7 Performance Impact

Blazar will need some performance cost for this comprehensive request parameters validation, because the checks will be increased for API parameters which are not validated now.

10.3.8 Other deployer impact

None

10.3.9 Developer impact

This will require developers contributing new extensions to Blazar to have a proper schema representing the extensions API.

10.4 Implementation

10.4.1 Assignee(s)

Primary assignee: Asmita Singh : <asmita.singh@nttdata.com>

10.4.2 Work Items

1. Initial validator implementation, which will contain common validator code designed to be shared across all resource controllers validating request bodies.
2. Introduce validation schemas for existing API resources.
3. Enforce validation on proposed API additions and extensions.
4. Remove duplicated ad-hoc validation code.
5. Add unit and end-to-end tests of related APIs.
6. Add/Update Blazar documentation.

10.5 Dependencies

None

10.6 Testing

Some tests can be added as each resource is validated against its schema. These tests should walk through invalid request types.

10.7 Documentation Impact

1. The Blazar API documentation will need to be updated to reflect the REST API changes.
2. The Blazar developer documentation will need to be updated to explain how the schema validation will work and how to add json schema for new APIs.

10.8 References

Some useful links:

- JSON Schema A Media Type for Describing JSON Documents
- Validation examples used in Nova project for jsonschema validation
- Library PyPI jsonschema for implementation of JSON Schema validation for Python

- Explaining the JSON Schema Core Definitions and Terminology
- JSON Schema specification documentation

10.9 History

Table 1: Revisions

Release Name	Description
Train	Introduced

CHAPTER

ELEVEN

EXAMPLE SPEC - THE TITLE OF YOUR BLUEPRINT

Include the URL of your launchpad blueprint:

https://blueprints.launchpad.net/blazar/+spec/example

Introduction paragraph why are we doing anything? A single paragraph of prose that operators can understand. The title and this first paragraph should be used as the subject line and body of the commit message respectively.

Some notes about the blazar-spec and blueprint process:

- Not all blueprints need a spec.
- The aim of this document is first to define the problem we need to solve, and second agree the overall approach to solve that problem.
- This is not intended to be extensive documentation for a new feature. For example, there is no need to specify the exact configuration changes, nor the exact details of any DB model changes. But you should still define that such changes are required, and be clear on how that will affect upgrades.
- You should aim to get your spec approved before writing your code. While you are free to write prototypes and code before getting your spec approved, its possible that the outcome of the spec review process leads you towards a fundamentally different solution than you first envisaged.
- But, API changes are held to a much higher level of scrutiny. As soon as an API change merges, we must assume it could be in production somewhere, and as such, we then need to support that API change forever. To avoid getting that wrong, we do want lots of details about API changes upfront.

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/blazar/+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
- For help with syntax, see http://sphinx-doc.org/rest.html
- To test out your formatting, build the docs using tox and see the generated HTML file in doc/build/html/specs/<path_of_your_file>
- 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.

- If your specification proposes any changes to the Blazar REST API such as changing parameters which can be returned or accepted, or even the semantics of what happens when a client calls into the API, then you should add the APIImpact flag to the commit message. Specifications with the APIImpact flag can be found with the following query:

  https://review.openstack.org/#/q/project:openstack/blazar+message:apiimpact,n,z

11.1 Problem description

A detailed description of the problem. What problem is this blueprint addressing?

11.1.1 Use Cases

What use cases does this address? What impact on actors does this change have? Ensure you are clear about the actors in each use case: Developer, End User, Deployer etc.

11.2 Proposed change

Here is where you cover the change you propose to make in detail. How do you propose to solve this problem?

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?

At this point, if you would like to just get feedback on if the problem and proposed change fit in blazar, you can stop here and post this for review to get preliminary feedback. If so please say: Posting to get preliminary feedback on the scope of this spec.

11.2.1 Alternatives

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.

11.2.2 Data model impact

Changes which require modifications to the data model often have a wider impact on the system. The community often has strong opinions on how the data model should be evolved, from both a functional and performance perspective. It is therefore important to capture and gain agreement as early as possible on any proposed changes to the data model.

Questions which need to be addressed by this section include:

- What new data objects and/or database schema changes is this going to require?
- What database migrations will accompany this change.
- How will the initial set of new data objects be generated, for example if you need to take into account existing instances, or modify other existing data describe how that will work.

11.2.3 REST API impact

Each API method which is either added or changed should have the following

- Specification for the method
  - A description of what the method does suitable for use in user documentation
  - Method type (POST/PUT/GET/DELETE)
  - Normal http response code(s)
  - Expected error http response code(s)
    - A description for each possible error code should be included describing semantic errors which can cause it such as inconsistent parameters supplied to the method, or when an instance is not in an appropriate state for the request to succeed. Errors caused by syntactic problems covered by the JSON schema definition do not need to be included.
  - URL for the resource
    - URL should not include underscores, and use hyphens instead.
  - Parameters which can be passed via the url
  - JSON schema definition for the request body data if allowed
    - Field names should use snake_case style, not CamelCase or MixedCase style.
  - JSON schema definition for the response body data if any
    - Field names should use snake_case style, not CamelCase or MixedCase style.
- Example use case including typical API samples for both data supplied by the caller and the response
- Discuss any policy changes, and discuss what things a deployer needs to think about when defining their policy.

Note that the schema should be defined as restrictively as possible. Parameters which are required should be marked as such and only under exceptional circumstances should additional parameters which are not defined in the schema be permitted (eg additionaProperties should be False).

Reuse of existing predefined parameter types such as regexps for passwords and user defined names is highly encouraged.

11.2.4 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 the 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.

- Can this change enable a resource exhaustion attack, such as allowing a single API interaction to consume significant server resources? Some examples of this include launching subprocesses for each connection, or entity expansion attacks in XML.

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.

11.2.5 Notifications impact

Please specify any changes to notifications. Be that an extra notification, changes to an existing notification, or removing a notification.

11.2.6 Other end user impact

Aside from the API, are there other ways a user will interact with this feature?

- Does this change have an impact on python-blazarclient? What does the user interface there look like?

11.2.7 Performance Impact

Describe any potential performance impact on the system, for example how often will new code be called, and is there a major change to the calling pattern of existing code.

Examples of things to consider here include:

- A small change in a utility function or a commonly used decorator can have a large impacts on performance.
- Calls which result in a database queries can have a profound impact on performance when called in critical sections of the code.
- Will the change include any locking, and if so what considerations are there on holding the lock?

11.2.8 Other deployer impact

Discuss things that will affect how you deploy and configure OpenStack that have not already been mentioned, such as:

- What config options are being added? Should they be more generic than proposed (for example a flag that other hypervisor drivers might want to implement as well)? 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 directory name that instances are stored in, how do we handle instance directories created before the change landed? Do we move them? Do we have a special case in the code? Do we assume that the operator will recreate all the instances in their cloud?

11.2.9 Developer impact

Discuss things that will affect other developers working on OpenStack, such as:

- If the blueprint proposes a change to the driver API, discussion of how other hypervisors would implement the feature is required.

11.2.10 Upgrade impact

Describe any potential upgrade impact on the system.

11.3 Implementation

11.3.1 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>

11.3.2 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.

11.4 Dependencies

- Include specific references to specs and/or blueprints in blazar, or in other projects, that this one either depends on or is related to.
- If this requires functionality of another project that is not currently used by Blazar (such as the glance v2 API when we previously only required v1), document that fact.
- Does this feature require any new library dependencies or code otherwise not included in OpenStack? Or does it depend on a specific version of library?

11.5 Testing

Please discuss the important scenarios needed to test here, as well as specific edge cases we should be ensuring work correctly. For each scenario please specify if this requires specialized hardware, a full openstack environment, or can be simulated inside the Blazar tree.

Please discuss how the change will be tested. We especially want to know what tempest tests will be added. It is assumed that unit test coverage will be added so that doesnt need to be mentioned explicitly, but discussion of why you think unit tests are sufficient and we dont need to add more tempest tests would need to be included.

Is this untestable in gate given current limitations (specific hardware / software configurations available)? If so, are there mitigation plans (3rd party testing, gate enhancements, etc).

11.6 Documentation Impact

Which audiences are affected most by this change, and which documentation titles on docs.openstack.org should be updated because of this change? Dont repeat details discussed above, but reference them here in the context of documentation for multiple audiences. For example, the Operations Guide targets cloud operators, and the End User Guide would need to be updated if the change offers a new feature available through the CLI or dashboard. If a config option changes or is deprecated, note here that the documentation needs to be updated to reflect this specifications change.

11.7 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 notes from a summit session
- Links to relevant research, if appropriate
- Related specifications as appropriate (e.g. if its an EC2 thing, link the EC2 docs)
- Anything else you feel it is worthwhile to refer to

11.8 History

Optional section intended to be used each time the spec is updated to describe new design, API or any database schema updated. Useful to let reader understand whats happened along the time.

Table 1: Revisions

Release Name	Description
Train	Introduced

CHAPTER

TWELVE

FLOATING IP RESERVATION

https://blueprints.launchpad.net/blazar/+spec/floatingip-reservation

This network plugin supports floating IP reservation.

12.1 Problem description

The Neutron service creates and associates floating IPs to projects in a first come first serve style. The order sometimes causes lack of floating IPs in a cloud, especially in a private cloud, since the cloud provider has a limited number of floating IPs available for their cloud.

12.1.1 Use Cases

- A user plans to scale up a service during a specific time window and the service requires a new floating IP for global access.
- A user plans to start a new service and wants to make sure a floating IP is available at the start time.
- A cloud admin does not have enough floating IPs to give to all users
  - The admin wants to give a user a floating IP with either the host or instance reservation.

12.2 Proposed change

Blazar enables users to reserve floating IPs by specifying the external network ID of floating IPs the users want to reserve. Users can treat the reserved floating IP as usual, except for the create floating IP operation.

A basic idea for the floating IP reservation and its scenario are as follows:

1. The admin registers some floating IPs used for floating IP reservation with Blazar. The admin calls Blazars floating IP API with a request body which includes public network ID and its floating IP address. The floating IP address must be out of allocation_pools in the external network subnets.
2. A user calls the create lease API with external network ID, start date, and end date. Blazar checks availability of a floating IP for the request. If a floating IP is available, Blazar creates an allocation between the floating IP and the reservation, then returns the reservation ID. If not, Blazar doesnt return a reservation ID.
3. At the start time, Blazar creates the reserved floating IP in the users tenant (project). Then the user can attach, detach, and delete the floating IP as usual.
4. At the end time, Blazar removes the reserved floating IP from the users tenant, if the user hasnt deleted the floating IP already.

Blazar regards IP addresses out of Neutrons allocation_pools parameter as reservable resources. The allocation_pools is just a range of IP addresses from which Neutron automatically creates floating IPs. When an admin creates a floating IP with a specific IP address which is out of the range, Neutron accepts the request and the new floating IP has the requested IP address. It enables Blazar to manage creation of the reserved floating IPs by itself.

Blazar adds two tags, blazar and reservation:<reservation-id> to the reserved floating IP to make it easy for users to query their reserved floating IPs.

To realize the scenario, this blueprint introduces a new resource plugin, named virtual:floatingip for Blazar.

12.2.1 Alternatives

Dedicated External Network approach

Blazar maps the reserved floating IPs to a dedicated external network. The external network is hidden from the regular users by neutrons policy.json. Blazar creates the reserving users external network at lease start time and deletes the network at lease end time on behalf of the user.

The advantage of this approach is the reserved resource is separated from users non-reserved resources. Its easy for Blazar to handle the reserved resources at start date and end date.

The disadvantage is that this approach needs the same amount of physical networks/configuration as the number of reserved networks. For example, if a cloud admin sets up their external network as type_driver is flat and mechanical_driver is ovs, Neutron needs as many OVS bridges as the number of reserved external networks.

Associated Port approach

Blazar attaches the reserved floating IPs to a specific port while the floating IP reservation is active. When the reservable floating IP is not in use, the IP belongs to Blazars service tenant. It prevents users from creating floating IPs using the reservable IPs.

The advantage is that Blazar can handle the creation and deletion of floating IP and the reservable floating IPs belongs to the range of the allocation_pools parameter.

The drawback is that users need to use a new workflow to manage the reserved floating IP. Without Blazar, users can associate and de-associate a floating IP to/from a port. But Blazar does in this approach instead of users. It requires users to have two workflows for managing floating IPs.

12.2.2 Data model impact

This plugin introduces four new tables, floatingip_reservations, required_floatingips, floatingip_allocations and floatingips.

The floatingip_reservations table keeps user request information for their floating IP reservations. The role of this table is similar to the role of the computehost_reservations table in the host reservation plugin. This table has id, network_id and amount columns. The table has a relationship with the set of floating IPs requested by user.

The required_floatingips table represents floating IPs that a user requests for a reservation.

The floatingip_allocations table has the relationship between the floatingip_reservations table and the floatingips table.

The floatingips table stores information of floating IPs themselves. The reservable floating IPs are registered in the table. The floating_ip_address column has unique constraints because the id column is generated by Blazar. Neutron generates floating ips id during floating ip creation.

The table definitions are as follows:

```
CREATE TABLE floatingip_reservations (
    id VARCHAR(36) NOT NULL,
    reservation_id VARCHAR(255) NOT NULL,
    network_id VARCHAR(255) NOT NULL,
    amount INT UNSIGNED NOT NULL,

    PRIMARY key (id),
    INDEX (id, reservation_id)
    FOREIGN KEY (reservation_id)
      REFERENCES reservations(id)
      ON DELETE CASCADE,
);

CREATE TABLE required_floatingips (
    id VARCHAR(36) NOT NULL,
    floatingip_reservation_id VARCHAR(36) NOT NULL,
    address VARCHAR(255) NOT NULL,

    PRIMARY key (id),
    FOREIGN KEY (floatingip_reservation_id)
      REFERENCES floatingip_reservations(id)
      ON DELETE CASCADE,
);

CREATE TABLE floatingip_allocations (
    id VARCHAR(36) NOT NULL,
    reservation_id VARCHAR(255) NOT NULL,
    floatingip_id VARCHAR(255) NOT NULL,
);

CREATE TABLE floatingips (
    id VARCHAR(36) NOT NULL,
    floating_network_id VARCHAR(255) NOT NULL,
    subnet_id VARCHAR(255) NOT NULL,
    floating_ip_address VARCHAR(255) NOT NULL,
    reservable BOOLEAN NOT NULL,

    UNIQUE (subnet_id, floating_ip_address)
);
```

12.2.3 REST API impact

The floating IP reservation introduces a new resource_type to the lease APIs and four new admin APIs to manages floating IPs.

Changes in the lease APIs

- URL: POST /v1/leases
  - Introduced new resource_type, virtual:floatingip, for a reservation.
  - The network_id is an external network ID from which the user wants to reserve a floating ip.
  - The required_floatingips is an optional key. The key represents a list of floating IPs which must be included in the reservation. In the request sample, an user wants 3 floating IPs, and wants to spcifiy 2 of 3 floating IPs and doesnt care of 1 of 3 floating IP.

Request Example:

```
{
  "name": "floatingip-reservation-1",
  "reservations": [
    {
      "resource_type": "virtual:floatingip",
      "network_id": "external-network-id",
      "required_floatingips": [
        "172.24.4.10",
        "172.24.4.11"
      ],
      "amount": 3
    }
  ],
  "start_date": "2017-05-17 09:07",
  "end_date": "2017-05-17 09:10",
  "events": []
}
```

Response Example:

```
{
  "lease": {
    "name": "floatingip-reservation-1",
    "reservations": [
      {
        "id": "reservation-id",
        "status": "pending",
        "lease_id": "lease-id-1",
        "resource_id": "resource_id",
        "resource_type": "virtual:floatingip",
        "network_id": "external-network-id",
        "required_floatingips": [
          "172.24.4.10",
          "172.24.4.11"
        ],
        "allocated_floatingips": [
          "172.24.4.10",
          "172.24.4.11",
          "172.24.4.100"
```

(continues on next page)

(continued from previous page)

```
        ],
        "amount": 3,
        "created_at": "2017-05-01 10:00:00",
        "updated_at": "2017-05-01 11:00:00",
      }],
    "start_date": "2017-05-17 09:07",
    "end_date": "2017-05-17 09:07",
    ..snip..
  }
}
```

- URL: GET /v1/leases
- URL: GET /v1/leases/{lease-id}
- URL: PUT /v1/leases/{lease-id}
- URL: DELETE /v1/leases/{lease-id}
  - The change is the same as POST /v1/leases

New floating IP APIs

The four new APIs are admin APIs by default.

- URL: POST /v1/floatingips
  - The floating_network_id is an external network ID the admin adds as Blazars resource.
  - The floating_ip_address is a specific floating IP address the admin wants to add. The IP address must be the out of allocation_pools. When admin calls the API, Blazar fetches the subnet info from Neutron and verifies the floating IP is out of allocation_pools and within its CIDR network.
  - The floating_ip_address cant be an optional parameter since IPs outside of the allocation_pool is commonly used by network equipment, a router, a loadbalancer and etc.

Request Example:

```
{
  "floating_network_id": "external-network-id",
  "floating_ip_address": "floating_ip_address"
}
```

- The reservable key is a flag describing if the floating IP is reservable or not. The flag is always True until the floating IP plugin supports the resource healing feature. (Supporting resource healing to floating IP is out of scope in this spec)

Response Example:

```
{
  "floatingip": {
      "id": "floating-ip-id",
      "floating_network_id": "external-network-id",
      "floating_ip_address": "floating_ip_address",
```

(continues on next page)

(continued from previous page)

```
        "subnet_id": "subnet-id",
        "reservable": true,
        "created_at": "2020-01-01 10:00:00",
        "updated_at": null
    }
}
```

- URL: GET /v1/floatingips

Response Example:

```
{
    "floatingips": [
        {
            "id": "floating-ip-id",
            "floating_network_id": "external-network-id",
            "floating_ip_address": "floating_ip_address",
            "subnet_id": "subnet-id",
            "reservable": true,
            "created_at": "2020-01-01 10:00:00",
            "updated_at": null
        }
    ]
}
```

- URL: GET /v1/floatingips/{floatingip-id}

Response Example:

```
{
    "floatingip": {
        "id": "floating-ip-id",
        "floating_network_id": "external-network-id",
        "floating_ip_address": "floating_ip_address",
        "subnet_id": "subnet-id",
        "reservable": true,
        "created_at": "2020-01-01 10:00:00",
        "updated_at": null
    }
}
```

- URL: DELETE /v1/floatingips/{floatingip-id}

No Request body and Response body.

The floating IP API doesnt have an update API because all of the information is retrieved from Neutron API.

12.2.4 Security impact

None

12.2.5 Notifications impact

None

12.2.6 Other end user impact

An user can reserve floating IPs as well as host or instance reservation in one lease.

python-blazarclient will support the floating IP reservation.

12.2.7 Performance Impact

None

12.2.8 Other deployer impact

None

12.2.9 Developer impact

This is a first implementation for networking resources.

12.2.10 Upgrade impact

Some configurations for Neutron util class will be introduced to blazar.conf. If the cloud admin want to activate the network reservation, they needs to setup the configuration.

12.3 Implementation

12.3.1 Assignee(s)

Primary assignee:
muroi-masahito

Other contributors:
None

12.3.2 Work Items

- Create Neutron API utility class
- Create the new DB tables
- Create the floating IP reservation plugin
- Create the floating IP API object and its route in blazar.api.v1
- Add floating IP reservation supports in python-blazarclient
- Add scenario tests and API tests in blazar-tempest-plugin
- Update Blazar docs, API reference and user guide

12.4 Dependencies

None

12.5 Testing

API tests and scenario tests need to be implemented.

12.6 Documentation Impact

This BP adds new APIs and resource type to the lease APIs. The API reference and the Blazar documentation need to be updated.

12.7 References

1. Draft for floating IP reservation: https://etherpad.openstack.org/p/network-resource-reservation
2. Denver PTG discussion: https://etherpad.openstack.org/p/blazar-ptg-stein

12.8 History

Table 1: Revisions

Release Name	Description
Stein	Introduced

CHAPTER

THIRTEEN

EXAMPLE SPEC - THE TITLE OF YOUR BLUEPRINT

Include the URL of your launchpad blueprint:

https://blueprints.launchpad.net/blazar/+spec/example

Introduction paragraph why are we doing anything? A single paragraph of prose that operators can understand. The title and this first paragraph should be used as the subject line and body of the commit message respectively.

Some notes about the blazar-spec and blueprint process:

- Not all blueprints need a spec.
- The aim of this document is first to define the problem we need to solve, and second agree the overall approach to solve that problem.
- This is not intended to be extensive documentation for a new feature. For example, there is no need to specify the exact configuration changes, nor the exact details of any DB model changes. But you should still define that such changes are required, and be clear on how that will affect upgrades.
- You should aim to get your spec approved before writing your code. While you are free to write prototypes and code before getting your spec approved, its possible that the outcome of the spec review process leads you towards a fundamentally different solution than you first envisaged.
- But, API changes are held to a much higher level of scrutiny. As soon as an API change merges, we must assume it could be in production somewhere, and as such, we then need to support that API change forever. To avoid getting that wrong, we do want lots of details about API changes upfront.

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/blazar/+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
- For help with syntax, see http://sphinx-doc.org/rest.html
- To test out your formatting, build the docs using tox and see the generated HTML file in doc/build/html/specs/<path_of_your_file>
- 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.

- If your specification proposes any changes to the Blazar REST API such as changing parameters which can be returned or accepted, or even the semantics of what happens when a client calls into the API, then you should add the APIImpact flag to the commit message. Specifications with the APIImpact flag can be found with the following query:

  https://review.openstack.org/#/q/project:openstack/blazar+message:apiimpact,n,z

13.1 Problem description

A detailed description of the problem. What problem is this blueprint addressing?

13.1.1 Use Cases

What use cases does this address? What impact on actors does this change have? Ensure you are clear about the actors in each use case: Developer, End User, Deployer etc.

13.2 Proposed change

Here is where you cover the change you propose to make in detail. How do you propose to solve this problem?

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?

At this point, if you would like to just get feedback on if the problem and proposed change fit in blazar, you can stop here and post this for review to get preliminary feedback. If so please say: Posting to get preliminary feedback on the scope of this spec.

13.2.1 Alternatives

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.

13.2.2 Data model impact

Changes which require modifications to the data model often have a wider impact on the system. The community often has strong opinions on how the data model should be evolved, from both a functional and performance perspective. It is therefore important to capture and gain agreement as early as possible on any proposed changes to the data model.

Questions which need to be addressed by this section include:

- What new data objects and/or database schema changes is this going to require?
- What database migrations will accompany this change.
- How will the initial set of new data objects be generated, for example if you need to take into account existing instances, or modify other existing data describe how that will work.

13.2.3 REST API impact

Each API method which is either added or changed should have the following

- Specification for the method
  - A description of what the method does suitable for use in user documentation
  - Method type (POST/PUT/GET/DELETE)
  - Normal http response code(s)
  - Expected error http response code(s)
    - A description for each possible error code should be included describing semantic errors which can cause it such as inconsistent parameters supplied to the method, or when an instance is not in an appropriate state for the request to succeed. Errors caused by syntactic problems covered by the JSON schema definition do not need to be included.
  - URL for the resource
    - URL should not include underscores, and use hyphens instead.
  - Parameters which can be passed via the url
  - JSON schema definition for the request body data if allowed
    - Field names should use snake_case style, not CamelCase or MixedCase style.
  - JSON schema definition for the response body data if any
    - Field names should use snake_case style, not CamelCase or MixedCase style.
- Example use case including typical API samples for both data supplied by the caller and the response
- Discuss any policy changes, and discuss what things a deployer needs to think about when defining their policy.

Note that the schema should be defined as restrictively as possible. Parameters which are required should be marked as such and only under exceptional circumstances should additional parameters which are not defined in the schema be permitted (eg additionaProperties should be False).

Reuse of existing predefined parameter types such as regexps for passwords and user defined names is highly encouraged.

13.2.4 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 the 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.

- Can this change enable a resource exhaustion attack, such as allowing a single API interaction to consume significant server resources? Some examples of this include launching subprocesses for each connection, or entity expansion attacks in XML.

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.

13.2.5 Notifications impact

Please specify any changes to notifications. Be that an extra notification, changes to an existing notification, or removing a notification.

13.2.6 Other end user impact

Aside from the API, are there other ways a user will interact with this feature?

- Does this change have an impact on python-blazarclient? What does the user interface there look like?

13.2.7 Performance Impact

Describe any potential performance impact on the system, for example how often will new code be called, and is there a major change to the calling pattern of existing code.

Examples of things to consider here include:

- A small change in a utility function or a commonly used decorator can have a large impacts on performance.
- Calls which result in a database queries can have a profound impact on performance when called in critical sections of the code.
- Will the change include any locking, and if so what considerations are there on holding the lock?

13.2.8 Other deployer impact

Discuss things that will affect how you deploy and configure OpenStack that have not already been mentioned, such as:

- What config options are being added? Should they be more generic than proposed (for example a flag that other hypervisor drivers might want to implement as well)? 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 directory name that instances are stored in, how do we handle instance directories created before the change landed? Do we move them? Do we have a special case in the code? Do we assume that the operator will recreate all the instances in their cloud?

13.2.9 Developer impact

Discuss things that will affect other developers working on OpenStack, such as:

- If the blueprint proposes a change to the driver API, discussion of how other hypervisors would implement the feature is required.

13.2.10 Upgrade impact

Describe any potential upgrade impact on the system.

13.3 Implementation

13.3.1 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>

13.3.2 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.

13.4 Dependencies

- Include specific references to specs and/or blueprints in blazar, or in other projects, that this one either depends on or is related to.
- If this requires functionality of another project that is not currently used by Blazar (such as the glance v2 API when we previously only required v1), document that fact.
- Does this feature require any new library dependencies or code otherwise not included in OpenStack? Or does it depend on a specific version of library?

13.5 Testing

Please discuss the important scenarios needed to test here, as well as specific edge cases we should be ensuring work correctly. For each scenario please specify if this requires specialized hardware, a full openstack environment, or can be simulated inside the Blazar tree.

Please discuss how the change will be tested. We especially want to know what tempest tests will be added. It is assumed that unit test coverage will be added so that doesnt need to be mentioned explicitly, but discussion of why you think unit tests are sufficient and we dont need to add more tempest tests would need to be included.

Is this untestable in gate given current limitations (specific hardware / software configurations available)? If so, are there mitigation plans (3rd party testing, gate enhancements, etc).

13.6 Documentation Impact

Which audiences are affected most by this change, and which documentation titles on docs.openstack.org should be updated because of this change? Dont repeat details discussed above, but reference them here in the context of documentation for multiple audiences. For example, the Operations Guide targets cloud operators, and the End User Guide would need to be updated if the change offers a new feature available through the CLI or dashboard. If a config option changes or is deprecated, note here that the documentation needs to be updated to reflect this specifications change.

13.7 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 notes from a summit session
- Links to relevant research, if appropriate
- Related specifications as appropriate (e.g. if its an EC2 thing, link the EC2 docs)
- Anything else you feel it is worthwhile to refer to

13.8 History

Optional section intended to be used each time the spec is updated to describe new design, API or any database schema updated. Useful to let reader understand whats happened along the time.

Table 1: Revisions

Release Name	Description
Stein	Introduced

CHAPTER

FOURTEEN

MULTI AVAILABILITY ZONES SUPPORT

https://blueprints.launchpad.net/blazar/+spec/multi-freepools

Support multiple availability zones and enable users to reserve both of hosts and instances with aware of az.

14.1 Problem description

Blazar manages hosts registered in the freepool for reservations. The freepool is agnostic for Novas availability zone (az) now. All hosts belong to one freepool and reserved hosts for a reservation can be picked up from different az in Nova. Additionally, users cant specify az the reserved hosts/instances belong to when they create a reservation.

14.1.1 Use Cases

- An user want to reserve instances to deploy a software cluster in one az.
- An user want to reserve hosts in specific az due to the location of the az.

14.2 Proposed change

This BP enables users to specify az in the host reservation and the instance reservation. If users specify az in their request, Blazar picks up hosts which belong to the specified az. If not, Blazar picks up hosts as usual. For the details of API change, please read the Rest API impact section.

Blazar records the original az information a host belongs to when operators register a host by the create host API. The az information comes from Novas service list API.

For backword compatibility, this BP introduce a new config, az_aware, to utils.openstack.nova.py. If its False, Blazar handles reservation requests like before. If its True, Blazar tracks availability zone of each hosts.

14.2.1 Alternatives

Multi freepools approach

Blazar manages multi freepools that is one-to-one mapping to each availability zone. Then users specify a freepool when they reserve resources if needed.

This approach also can support multiple availability zone. However, Blazar need to introduce new API sets to create the one-to-one mapping between az and freepool. The API set add extra tasks that operators define the mappings before they call the create host API.

ComputeExtraCapability approach

Operators define az information as ComputeExtraCapability to enable users can specify az when they create a reservation.

The good point of this approach is there is no need to change Blazars APIs and config since operators only call existing APIs to create extra_capability key and value set.

The drawback is that if Blazar automatically stores az info to ComputeExtraCapability its not a good place to store Novas info queried by Blazar. ComputeExtraCapability is a table for data specified by operators and ComputeHost is a table for data queried by Blazar.

14.2.2 Data model impact

A availability_zone column is added to the ComputeHost table. This column stores the availability zone the host belongs to.

```
ALTER TABLE computehosts ADD
    availability_zone VARCHAR(255) AFTER status;
```

NULL is assigned to the colum for the upgrade from Pike to later.

14.2.3 REST API impact

- URL: POST /v1/leases
  - The hypervisor_properties in physical:host and the resource_properties in virtual:instance support a query for availability_zone key.

Request Example:

```
{
  "name": "instance-reservation-1",
  "reservations": [
    {
      "resource_type": "virtual:instance",
      "vcpus": 4,
      "memory_mb": 4096,
      "disk_gb": 10,
      "amount": 5,
      "affinity": False,
      "resource_properties": "[\"==\", \"$availability_zone\", \"az1\
↪"]"
    },
    {
      "resource_type": "physical:host",
      "min": 3,
      "max": 4,
      "hypervisor_properties": "[]",
      "resource_properties": "[\"==\", \"$availability_zone\", \"az1\
↪"]"
    }
  ],
  "start": "2020-05-17 09:00"
```

(continues on next page)

(continued from previous page)

```
  "end": "2020-05-17 10:00",
  "events": []
}
```

Response Example:

```
{
  "leases": {
    "reservations": [
      {
        "id": "reservation-id",
        "status": "pending",
        "lease_id": "lease-id-1",
        "resource_id": "resource_id",
        "resource_type": "virtual:instance",
        "vcpus": 4,
        "memory_mb": 4096,
        "disk_gb": 10,
        "amount": 5,
        "affinity": False,
        "resource_properties": "[\"==\", \"$availability_zone\", \
↪"az1\"]",
        "created_at": "2017-05-01 10:00:00",
        "updated_at": "2017-05-01 11:00:00",
      }],
    ..snippet..
  }
}
```

- URL: GET /v1/leases
- URL: GET /v1/leases/{lease-id}
- URL: PUT /v1/leases/{lease-id}
- URL: DELETE /v1/leases/{lease-id}
  - The change is the same as POST /v1/leases

14.2.4 Security impact

None

14.2.5 Notifications impact

None

14.2.6 Other end user impact

The original az name a hypervisor belongs to is only visible through Blazar API. Nova returns az name based on meta data of host aggregate and Blazar sets blazar_* az name to an aggregate of host reservation. It results users need to call Blazar Host details API when they want to know what az value is available in availability_zone key.

In most cases, only admin is allowed to configure az in Nova. Admins/cloud providers/cloud deployers inform end users of list of az name. So the impact described above has less impact to end users.

14.2.7 Performance Impact

None

14.2.8 Other deployer impact

When upgrading Blazar, availability_zone column is filled by NULL. If depoloyers set the az_aware flag to True, they need to re-create all hosts registered in Blazars freeppol after upgrading to store availability zone information into computehost table. If hosts are used for a host reservation Blazar cant find out the original az information while deployers upgrade Blazar.

If deployers move a host to another availability zone by Nova API, the deployers need to re-create the host by the Blazar host create API to apply the new availability zone to the Blazar DB. The information is automatically registered by Blazar only in the Blazar host create API.

14.2.9 Developer impact

None

14.3 Implementation

14.3.1 Assignee(s)

Primary assignee:
muroi-masahito

Other contributors:
None

14.3.2 Work Items

- Add availability_zone column to computehosts table
- Implement availability_zone support in the create host API
- Support availability_zone flag in blazarclient

14.4 Dependencies

None

14.5 Testing

- Unit tests

14.6 Documentation Impact

- API reference

14.7 References

None

14.8 History

Table 1: Revisions

Release Name	Description
Queens	Introduced
Rocky	Re-proposed

CHAPTER

FIFTEEN

PLACEMENT API SUPPORT FOR INSTANCE RESERVATION

https://blueprints.launchpad.net/blazar/+spec/placement-api

Placement API[1] was introduced to Nova at the 14.0.0 Newton release to separate API and data model for tracking resource provider inventories and usages. It can be used for improving instance reservation.

15.1 Problem description

Current instance reservation has the following constraints:

- A user has to create instance reservation with the anti-affinity policy. Therefore, the amount of instances in one reservation cannot be larger than the amount of hosts.
- A user has to specify the server group when the user launches instances on reserved resources. If it is not specified, more instances than the reserved amount are possibly launched.

15.1.1 Use Cases

- A user wants to reserve instance resources with arbitrary affinity policy.
- A user wants to reserve more instances than the number of hosts.

15.2 Proposed change

Use the custom resource class to represent reservation resources and use the nested resource provider to manage capacity and usage of reservation resources. The following sections describe how Blazar interacts with Nova and Placement for supporting instance reservation.

15.2.1 When creating a host:

1. Get hypervisor information and store it into the computehosts table.
2. Create a nested resource provider as a child of the compute node resource provider by calling the Create resource provider API. The UUID of the compute node resource provider can be retrieved by calling the List resource providers API with the **name** option query, e.g. `GET /placement/resource_proiders?name=compute-1`.

   The child resource provider is referred to as *reservation provider* in the following sections.

   Create reservation provider request body example:

   ```
   POST /placement/resource_providers
   ```

[1] https://docs.openstack.org/nova/latest/user/placement.html

```
{
    "name": "blazar_compute-1",
    "parent_provider_uuid": "542df8ed-9be2-49b9-b4db-6d3183ff8ec8"
}
```

Note

542df8ed-9be2-49b9-b4db-6d3183ff8ec8 is the UUID of the compute-1 compute node.

3. Add the host into the freepool.

15.2.2 When creating a lease:

1. Look for available resources with arbitrary affinity policy.
2. Update the computehost_allocations table.
3. Create a custom resource class **CUSTOM_RESERVATION_{reservation UUID}** by calling the Create resource class API.

   Create resource class request body example:

   `POST /placement/resource_classes`

```
{
    "name": "CUSTOM_RESERVATION_4D17D41A_830D_47B2_91C7_4F9FC0AE611E"
}
```

Note

Use upper case and under score for the custom resource class name because lower case and hyphen cannot be used.

4. Create a private flavor which has `resources:CUSTOM_RESERVATION_{reservation UUID}=1` in its extra_spec.

Note

- A host aggregate is not created for each instance reservation anymore because reserved hosts can be distinguished by the reservation provider inventory.
- A server group is not created anymore because the proposed approach does not depend on the ServerGroup(Anti)AffinityFilter.

15.2.3 When starting a lease:

1. Add the custom resource class **CUSTOM_RESERVATION_{reservation UUID}** into the reservation providers inventory by calling the Update resource provider inventories API with the **total** parameter which equals to the amount of instances reserved for the host.

   Update resource provider inventories request body example:

   `PUT /placement/resource_providers/{reservation_provider_uuid}/inventories`

```
{
    "inventories": {
        "CUSTOM_RESERVATION_4D17D41A_830D_47B2_91C7_4F9FC0AE611E": {
            "total": 3,
            "allocation_ratio": 1.0,
            "min_unit": 1,
            "max_unit": 1,
            "step_size": 1
        },
        "snip"
    },
    "resource_provider_generation": 5
}
```

Note

Existing hosts which were created before this spec is implemented do not have the reservation provider. So, check if the reservation provider exists and create it if it does not exist before this step.

2. Add the lease owners project to the private flavor access rights list.

Note

The previous implementation of starting lease should be kept until the previous instance reservation is deprecated and completely removed. The previous instance reservations can be distinguished by checking the aggregate_id or server_group_id column in the instance_reservations table.

15.2.4 When launching instances (from user point of view):

1. A lease owner uses the private flavor and the instance is launched on the reserved host which has the **CUSTOM_RESERVATION_{reservation UUID}** in its child resource provider inventory, i.e. reservation provider inventory.

   Consumption of **CUSTOM_RESERVATION_{reservation UUID}** resources in the reservation provider inventory is claimed by the Nova scheduler. It means that usage of reserved resources is automatically tracked by the Placement.

Note

It still depends on the *BlazarFilter* though the *BlazarFilter* will be ideally removed in the future. The *BlazarFilter* is changed to check if `resources:CUSTOM_RESERVATION_*` is in flavor extra specs to distinguish the request from normal, i.e. non-reserved, instance creation requests.

Traits or other features would be able to be used for solving *BlazarFilter* dependency. It would be addressed by another blueprint.

On the other hand, dependency on the following filters are solved. These filters are not needed any more.

- AggregateInstanceExtraSpecsFilter

- AggregateMultiTenancyIsolationFilter
- ServerGroupAntiAffinityFilter

Note that above filters and existing logic in the BlazarFilter should be kept to keep backward compatibility until the previous instance reservation is deprecated and completely removed.

15.2.5 When terminating a lease:

1. Delete related instances and the private flavor.
2. Remove the **CUSTOM_RESERVATION_{reservation UUID}** class from the reservation providers inventory by calling the Delete resource provider inventory API.
3. Delete the **CUSTOM_RESERVATION_{reservation_UUID}** resource class by calling the Delete resource class API.

Note

The previous implementation of terminating lease should be kept until the previous instance reservation is deprecated and completely removed. The previous instance reservations can be distinguished by checking the aggregate_id or server_group_id column in the instance_reservations table.

15.2.6 When deleting a host:

1. Delete the reservation provider which is associated with the host by calling the Delete resource provider API.
2. Remove the host from the freepool.
3. Update the computehosts table.

15.2.7 Alternatives

Dummy resources approach

Update inventories of the general resources, e.g. VCPU, of compute nodes in the freepool to be **zero** or **reserved**. And add dummy resources like **CUSTOM_VCPU_{reservation UUID}** into the inventory. This approach complicates resource usage tracking because real usage of each general resource cannot be seen through the top level compute node inventory.

Traits approach

Use Traits to express reserved resources. The problem is that traits are just traits and they cannot be used for managing capacity and usage of reserved resources.

15.2.8 Data model impact

The **affinity** column of the instance_reservations table is changed to allow `NULL`. `NULL` means `no affinity policy is applied` while `True` means `affinity is applied` and `False` means `anti-affinity is applied`.

The instance_reservations table:

```
ALTER TABLE instance_reservations
    ALTER COLUMN affinity NULL;
```

After the previous instance reservation is deprecated and completely removed, drop the following columns in the instance_reservations table:

```
ALTER TABLE instance_reservations
    DROP COLUMN aggregate_id, server_group_id;
```

15.2.9 REST API impact

The **affinity** parameter of the Create lease API is changed to be an optional parameter. If the **affinity** parameter is not given, no affinity policy is applied.

15.2.10 Security impact

None

15.2.11 Notifications impact

None

15.2.12 Other end user impact

None

15.2.13 Performance Impact

None

15.2.14 Other deployer impact

- The Placement API has to be newer than or equal to Ver. 1.29.
- To upgrade from the previous version, run the DB upgrade script and the instance_reservations table schema will be updated.

15.2.15 Developer impact

None

15.3 Implementation

15.3.1 Assignee(s)

Primary assignee:
<tetsuro>

Other contributors:
<hiro-kobayashi>

15.3.2 Work Items

Base:

- Update DB schema: update the instance_reservations table.
- Add placement library in blazar/utils/openstack module.

To support the host creation:

- Update the create_computehost() of the host plugin to call Placement APIs and update related tables.

To support the host deletion:

- Update the delete_computehost() of the host plugin to delete Placement related resources.

To support the lease creation:

- Update the query_available_hosts() to return how many instance can be launched on each available host.
- Update the pickup_hosts() to support arbitrary affinity policy.
- Update the reserve_resource() and update_reservation() to support multiple allocations which have the same pair of reservation_id and computehost_id.
- Update the _create_resources() of the instance plugin to create the **CUSTOM_RESERVATION_{reservation UUID}** class and add it into the private flavor extra specs.

To support starting the lease:

- Update the on_start() of the instance plugin to add the **CUSTOM_RESERVATION_{reservation UUID}** into the reservation provider inventory. The **total** parameter equals to the number of entries of the computehost_allocations table which have the same reservation id and computehost id.

To support launching reserved instances:

- Update the *BlazarFilter*.

To support termination of the lease:

- Update the on_end() of the instance plugin to remove the custom resource from the reservation provider inventory and delete the class itself.

Others:

- Update the api module and the python-blazarclient to support arbitrary affinity policies.
- Update the blazar-dashboard to support arbitrary affinity policies.
- Update documentation.

15.4 Dependencies

WIP: Check Placement API development status.

15.5 Testing

- Add unit tests for new features of each method described in the work items section.
- Add test scenarios of instance reservation with the affinity policy and no affinity policy.

15.6 Documentation Impact

- Parameter description of the Create Lease API reference will be updated.
- Instance reservation part of the Command-Line Interface Reference will be updated.
- Release notes will be added.

15.7 References

15.8 History

Table 1: Revisions

Release Name	Description
Rocky	Introduced

CHAPTER

SIXTEEN

RESOURCE ALLOCATION API

https://blueprints.launchpad.net/blazar/+spec/resource-availability-api

Introducing new APIs for querying current usage of each resource.

16.1 Problem description

A Blazar reservation consumes at least one reservable resource. For host reservation and instance reservation, a reservation is tied to specific hosts and the relationship is stored in the Blazar DB.

Blazar has no API describing the consumption relationship. Blazar has list APIs for leases and hosts, which show cloud users and cloud admins either list of leases or reservable hosts. However, the scope of both APIs is only individual resource information.

Cloud admins have no way to find the relationship through the Blazar API. If they would like to know the usage of hosts for a specific time window, they need to query the Blazar DB directly. Direct DB querying by users is not supported in general.

16.1.1 Use Cases

- A cloud admin want to find the upcoming usage of a specific host for maintenance.

16.2 Proposed change

Introducing a new API set, get and list reservation allocation API, to host APIs. This APIs show list of reservations which consumes reservable hosts. If cloud admins call this API, they can find all reservations consuming specific hosts.

Note

The spec and the blueprint are named resource **availability** API. However, the proposed API change responds existing reservations allocations. The name of the API set are changed from availability to reservation allocation API.

The API set are part of Hosts API. The default authorization policy is admin API.

See the REST API impact section for the details of the API.

16.2.1 Alternatives

Appending a new key-value pair to the lease get API and the lease list API. The pair could form like `"hosts": [{"id": 1}, {"id": 2}]`, and be added to each reservation details.

The good point of this change is not introducing a new API. Introducing a new API always has an impact for pythonclient, too.

The drawback is the authentification and the authorization for the API call become more complex. The response body changes depending on the keystone token. If a token scopes admin role, the API needs to create its response with host information. If not, the API doesnt have to add the information.

16.2.2 Data model impact

None.

16.2.3 REST API impact

- URL: GET /v1/os-hosts/allocations
  - The API replies the all allocations between reservations and hosts.
  - Nomal response code: 200
  - Error response code: Bad Request(400), Unauthorized(401), Forbidden(403), Internal Server Error(500)

Response Example:

```
{
  "allocations": [
      {
           "resource_id": "host-id1",
           "reservations": [
             {
               "id": "reservation-id1",
               "lease_id": "lease-id1"
             },
             {
               "id": "reservation-id2",
               "lease_id": "lease-id1"
             }
           ]
      },
      ..snippet..
  ]
}
```

- URL: GET /v1/os-hosts/{host-id}/allocation
  - The API replies the all allocations only for the host.
  - Nomal response code: 200
  - Error response code: Bad Request(400), Unauthorized(401), Forbidden(403), Not Found(404), Internal Server Error(500)

Response Example:

```
{
  "allocation": {
      "resource_id": "host-id1",
      "reservations": [
        {
          "id": "reservation-id1",
          "lease_id": "lease-id1"
        },
        {
          "id": "reservation-id2",
          "lease_id": "lease-id1"
        }
      ]
   }
}
```

Both APIs support some query parameters.

- lease_id: A parameter that filters allocations belonging to the lease_id
- reservation_id: A parameter that filters allocations belonging to the reservation_id
- terminated: A flag that filters allocations already terminated or not

16.2.4 Security impact

None

16.2.5 Notifications impact

None

16.2.6 Other end user impact

The pythonclient will support the allocation APIs.

16.2.7 Performance Impact

List all allocations API, GET /v1/os-hosts/allocations, returns all allocations. When the number of hosts and reservations are huge, the DB query and response body could become huge, too.

To try reducing the number of DB query, the two API use queries like followings.

```
 # List reservation allocations API
 SELECT computehost_allocations.host, reservation.id, reservations.
↪lease_id
   FROM computehost_allocations
     JOIN reservations ON computehost_allocations.reservation_id =␣
↪reservations.id;

# Get reservation allocations API
 SELECT computehost_allocations.host, reservation.id, reservations.
↪lease_id
   FROM computehost_allocations
```

(continues on next page)

(continued from previous page)

```
    JOIN reservations ON computehost_allocations.reservation_id =␣
↪reservations.id
  WHERE computehost_allocations.host = host_id;
```

16.2.8 Other deployer impact

None

16.2.9 Developer impact

None

16.3 Implementation

16.3.1 Assignee(s)

Primary assignee:
muroi-masahito

Other contributors:
None

16.3.2 Work Items

- Support query parameters for GET request
- Implement the reservation allocation API in host plugin
- Support the reservation allocation API in blazarclient

16.4 Dependencies

None

16.5 Testing

- Unit tests
- Tempest scenario tests

16.6 Documentation Impact

- API reference

16.7 References

16.8 History

Table 1: Revisions

Release Name	Description
Rocky	Introduced

CHAPTER

SEVENTEEN

EXAMPLE SPEC - THE TITLE OF YOUR BLUEPRINT

Include the URL of your launchpad blueprint:

https://blueprints.launchpad.net/blazar/+spec/example

Introduction paragraph why are we doing anything? A single paragraph of prose that operators can understand. The title and this first paragraph should be used as the subject line and body of the commit message respectively.

Some notes about the blazar-spec and blueprint process:

- Not all blueprints need a spec.
- The aim of this document is first to define the problem we need to solve, and second agree the overall approach to solve that problem.
- This is not intended to be extensive documentation for a new feature. For example, there is no need to specify the exact configuration changes, nor the exact details of any DB model changes. But you should still define that such changes are required, and be clear on how that will affect upgrades.
- You should aim to get your spec approved before writing your code. While you are free to write prototypes and code before getting your spec approved, its possible that the outcome of the spec review process leads you towards a fundamentally different solution than you first envisaged.
- But, API changes are held to a much higher level of scrutiny. As soon as an API change merges, we must assume it could be in production somewhere, and as such, we then need to support that API change forever. To avoid getting that wrong, we do want lots of details about API changes upfront.

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/blazar/+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
- For help with syntax, see http://sphinx-doc.org/rest.html
- To test out your formatting, build the docs using tox and see the generated HTML file in doc/build/html/specs/<path_of_your_file>
- 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.

- If your specification proposes any changes to the Blazar REST API such as changing parameters which can be returned or accepted, or even the semantics of what happens when a client calls into the API, then you should add the APIImpact flag to the commit message. Specifications with the APIImpact flag can be found with the following query:

  https://review.openstack.org/#/q/project:openstack/blazar+message:apiimpact,n,z

17.1 Problem description

A detailed description of the problem. What problem is this blueprint addressing?

17.1.1 Use Cases

What use cases does this address? What impact on actors does this change have? Ensure you are clear about the actors in each use case: Developer, End User, Deployer etc.

17.2 Proposed change

Here is where you cover the change you propose to make in detail. How do you propose to solve this problem?

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?

At this point, if you would like to just get feedback on if the problem and proposed change fit in blazar, you can stop here and post this for review to get preliminary feedback. If so please say: Posting to get preliminary feedback on the scope of this spec.

17.2.1 Alternatives

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.

17.2.2 Data model impact

Changes which require modifications to the data model often have a wider impact on the system. The community often has strong opinions on how the data model should be evolved, from both a functional and performance perspective. It is therefore important to capture and gain agreement as early as possible on any proposed changes to the data model.

Questions which need to be addressed by this section include:

- What new data objects and/or database schema changes is this going to require?
- What database migrations will accompany this change.
- How will the initial set of new data objects be generated, for example if you need to take into account existing instances, or modify other existing data describe how that will work.

17.2.3 REST API impact

Each API method which is either added or changed should have the following

- Specification for the method
  - A description of what the method does suitable for use in user documentation
  - Method type (POST/PUT/GET/DELETE)
  - Normal http response code(s)
  - Expected error http response code(s)
    - A description for each possible error code should be included describing semantic errors which can cause it such as inconsistent parameters supplied to the method, or when an instance is not in an appropriate state for the request to succeed. Errors caused by syntactic problems covered by the JSON schema definition do not need to be included.
  - URL for the resource
    - URL should not include underscores, and use hyphens instead.
  - Parameters which can be passed via the url
  - JSON schema definition for the request body data if allowed
    - Field names should use snake_case style, not CamelCase or MixedCase style.
  - JSON schema definition for the response body data if any
    - Field names should use snake_case style, not CamelCase or MixedCase style.
- Example use case including typical API samples for both data supplied by the caller and the response
- Discuss any policy changes, and discuss what things a deployer needs to think about when defining their policy.

Note that the schema should be defined as restrictively as possible. Parameters which are required should be marked as such and only under exceptional circumstances should additional parameters which are not defined in the schema be permitted (eg additionaProperties should be False).

Reuse of existing predefined parameter types such as regexps for passwords and user defined names is highly encouraged.

17.2.4 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 the 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.

- Can this change enable a resource exhaustion attack, such as allowing a single API interaction to consume significant server resources? Some examples of this include launching subprocesses for each connection, or entity expansion attacks in XML.

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.

17.2.5 Notifications impact

Please specify any changes to notifications. Be that an extra notification, changes to an existing notification, or removing a notification.

17.2.6 Other end user impact

Aside from the API, are there other ways a user will interact with this feature?

- Does this change have an impact on python-blazarclient? What does the user interface there look like?

17.2.7 Performance Impact

Describe any potential performance impact on the system, for example how often will new code be called, and is there a major change to the calling pattern of existing code.

Examples of things to consider here include:

- A small change in a utility function or a commonly used decorator can have a large impacts on performance.
- Calls which result in a database queries can have a profound impact on performance when called in critical sections of the code.
- Will the change include any locking, and if so what considerations are there on holding the lock?

17.2.8 Other deployer impact

Discuss things that will affect how you deploy and configure OpenStack that have not already been mentioned, such as:

- What config options are being added? Should they be more generic than proposed (for example a flag that other hypervisor drivers might want to implement as well)? 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 directory name that instances are stored in, how do we handle instance directories created before the change landed? Do we move them? Do we have a special case in the code? Do we assume that the operator will recreate all the instances in their cloud?

17.2.9 Developer impact

Discuss things that will affect other developers working on OpenStack, such as:

- If the blueprint proposes a change to the driver API, discussion of how other hypervisors would implement the feature is required.

17.3 Implementation

17.3.1 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>

17.3.2 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.

17.4 Dependencies

- Include specific references to specs and/or blueprints in blazar, or in other projects, that this one either depends on or is related to.
- If this requires functionality of another project that is not currently used by Blazar (such as the glance v2 API when we previously only required v1), document that fact.
- Does this feature require any new library dependencies or code otherwise not included in OpenStack? Or does it depend on a specific version of library?

17.5 Testing

Please discuss the important scenarios needed to test here, as well as specific edge cases we should be ensuring work correctly. For each scenario please specify if this requires specialized hardware, a full openstack environment, or can be simulated inside the Blazar tree.

Please discuss how the change will be tested. We especially want to know what tempest tests will be added. It is assumed that unit test coverage will be added so that doesnt need to be mentioned explicitly, but discussion of why you think unit tests are sufficient and we dont need to add more tempest tests would need to be included.

Is this untestable in gate given current limitations (specific hardware / software configurations available)? If so, are there mitigation plans (3rd party testing, gate enhancements, etc).

17.6 Documentation Impact

Which audiences are affected most by this change, and which documentation titles on docs.openstack.org should be updated because of this change? Dont repeat details discussed above, but reference them here in the context of documentation for multiple audiences. For example, the Operations Guide targets cloud operators, and the End User Guide would need to be updated if the change offers a new feature available through the CLI or dashboard. If a config option changes or is deprecated, note here that the documentation needs to be updated to reflect this specifications change.

17.7 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 notes from a summit session
- Links to relevant research, if appropriate
- Related specifications as appropriate (e.g. if its an EC2 thing, link the EC2 docs)
- Anything else you feel it is worthwhile to refer to

17.8 History

Optional section intended to be used each time the spec is updated to describe new design, API or any database schema updated. Useful to let reader understand whats happened along the time.

Table 1: Revisions

Release Name	Description
Rocky	Introduced

CHAPTER

EIGHTEEN

EXTRA SPECS FOR INSTANCE RESERVATION

https://blueprints.launchpad.net/blazar/+spec/flavors-extra-specs

Support extra-specs in the instance reservation.

Note

There are two different phrases, *extra specs* and *resource_properties* in this spec. The two phrases refer to the same thing that is defined as ComputeHostExtraCapability. In this document, the title and the problem description use *extra specs* to retain the original phrase used in the BP.

18.1 Problem description

Users can request vcpus, memory, disk, amount and affinity as a parameter for instance reservation. Blazar checks whether the requested instances can be reserved or not based on the users request.

The demands for reserved instances are not only size of its flavor and amount, but specific hardware spec or features. For example, some users want to reserve GPU instances. However, the instance reservation plugin doesnt support extra specs as a parameter.

18.1.1 Use Cases

For NFV area, some kinds of instances need to run on specific hypervisors, like DPDK, SR-IOV, etc. If the instance reservation doesnt support extra specs, reserved instances could be scheduled to non-DPDK hypervisors though the user want the instance to be scheduled to DPDK hypervisors.

For HPC area, GPGPU is common in the area. So when users reserve instances, they may want to request GPU instances.

18.2 Proposed change

Instance reservation plugin supports resource_properties key in its request body. See the REST API impact section for the change of the request body. This specs focuses only on supporting resource_properties matches to ComputeHostExtraCapability.

When an user reserves instances with resource_properties, Blazar picks up hypervisors which can accommodate the requested flavor and the resource_properties.

When admins update ComputeHostExtraCapability, Blazar re-allocates reservations related to the updated ExtraCapability. The re-allocation strategy is the same as used by the update_reservation API and resource-monitoring feature.

18.2.1 Alternatives

An user directly specifies the list of hypervisors which have the specific features in a request body. Users check which hypervisors have the features before they create instance reservations, then they decide which hypervisors they want to use.

This approach could be easier to implement than the proposed change since what Blazar needs to do is just pick up hypervisors from the list. However, the admin can change ComputeHostExtraCapability anytime. When a specific feature users want to use is changed, the user have to send a new list of hypervisors again. Additionally, a cloud may be configured to forbid users from looking up hosts and their extra capabilities.

18.2.2 Data model impact

A resource_properties column is added to the InstanceReservations table. This column stores the raw string of the resource_properties sent by users.

The change for the InstanceReservations table is as follows:

```
ALTER TABLE instance_reservations ADD
    resource_properties MEDIUMTEXT AFTER affinity;
```

NULL is assigned to the column for the upgrade from Pike to later.

18.2.3 REST API impact

- URL: POST /v1/leases
  - Introduce the resource_properties key in virtual:instance resource_type

Request Example:

```
{
  "name": "instance-reservation-1",
  "reservations": [
    {
      "resource_type": "virtual:instance",
      "vcpus": 4,
      "memory_mb": 4096,
      "disk_gb": 10,
      "amount": 5,
      "affinity": False,
      "resource_properties": {
          "property_key1": "value1",
          "property_key2": "value2"
      }
    }
   ],
  "start": "2020-05-17 09:00"
  "end": "2020-05-17 10:00",
  "events": []
}
```

Response Example:

```
{
  "leases": {
    "reservations": [
      {
        "id": "reservation-id",
        "status": "pending",
        "lease_id": "lease-id-1",
        "resource_id": "resource_id",
        "resource_type": "virtual:instance",
        "vcpus": 4,
        "memory_mb": 4096,
        "disk_gb": 10,
        "amount": 5,
        "affinity": False,
        "resource_properties": {
           "property_key1": "value1",
           "property_key2": "value2"
        }
        "created_at": "2017-05-01 10:00:00",
        "updated_at": "2017-05-01 11:00:00",
      }],
    ..snippet..
  }
}
```

- URL: GET /v1/leases
- URL: GET /v1/leases/{lease-id}
- URL: PUT /v1/leases/{lease-id}
- URL: DELETE /v1/leases/{lease-id}
  - The change is the same as POST /v1/leases

18.2.4 Security impact

None

18.2.5 Notifications impact

None

18.2.6 Other end user impact

python-blazarclient needs to support resource_properties parameter in lease handling commands.

18.2.7 Performance Impact

None

18.2.8 Other deployer impact

None

18.2.9 Developer impact

None

18.3 Implementation

18.3.1 Assignee(s)

Primary assignee:
muroi-masahito

Other contributors:
None

18.3.2 Work Items

- Add resource_properties column to InstanceReservation table
- Support resource_properties key in instance reservation plugin
- Add re-allocation logic to ComputeHostExtraCapability management
- Support resource_properties parameter at python-blazarclient

18.4 Dependencies

None

18.5 Testing

- The scenario test for instance reservation should support resource_properties

18.6 Documentation Impact

- API reference

18.7 References

1. OPNFV Promise : http://artifacts.opnfv.org/promise/docs/development_manuals/index.html
2. resource-monitoring BP: https://blueprints.launchpad.net/blazar/+spec/resource-monitoring

18.8 History

Table 1: Revisions

Release Name	Description
Queens	Introduced

CHAPTER

NINETEEN

EXAMPLE SPEC - THE TITLE OF YOUR BLUEPRINT

Include the URL of your launchpad blueprint:

https://blueprints.launchpad.net/blazar/+spec/example

Introduction paragraph why are we doing anything? A single paragraph of prose that operators can understand. The title and this first paragraph should be used as the subject line and body of the commit message respectively.

Some notes about the blazar-spec and blueprint process:

- Not all blueprints need a spec.
- The aim of this document is first to define the problem we need to solve, and second agree the overall approach to solve that problem.
- This is not intended to be extensive documentation for a new feature. For example, there is no need to specify the exact configuration changes, nor the exact details of any DB model changes. But you should still define that such changes are required, and be clear on how that will affect upgrades.
- You should aim to get your spec approved before writing your code. While you are free to write prototypes and code before getting your spec approved, its possible that the outcome of the spec review process leads you towards a fundamentally different solution than you first envisaged.
- But, API changes are held to a much higher level of scrutiny. As soon as an API change merges, we must assume it could be in production somewhere, and as such, we then need to support that API change forever. To avoid getting that wrong, we do want lots of details about API changes upfront.

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/blazar/+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
- For help with syntax, see http://sphinx-doc.org/rest.html
- To test out your formatting, build the docs using tox and see the generated HTML file in doc/build/html/specs/<path_of_your_file>
- 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.

- If your specification proposes any changes to the Blazar REST API such as changing parameters which can be returned or accepted, or even the semantics of what happens when a client calls into the API, then you should add the APIImpact flag to the commit message. Specifications with the APIImpact flag can be found with the following query:

  https://review.openstack.org/#/q/project:openstack/blazar+message:apiimpact,n,z

19.1 Problem description

A detailed description of the problem. What problem is this blueprint addressing?

19.1.1 Use Cases

What use cases does this address? What impact on actors does this change have? Ensure you are clear about the actors in each use case: Developer, End User, Deployer etc.

19.2 Proposed change

Here is where you cover the change you propose to make in detail. How do you propose to solve this problem?

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?

At this point, if you would like to just get feedback on if the problem and proposed change fit in blazar, you can stop here and post this for review to get preliminary feedback. If so please say: Posting to get preliminary feedback on the scope of this spec.

19.2.1 Alternatives

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.

19.2.2 Data model impact

Changes which require modifications to the data model often have a wider impact on the system. The community often has strong opinions on how the data model should be evolved, from both a functional and performance perspective. It is therefore important to capture and gain agreement as early as possible on any proposed changes to the data model.

Questions which need to be addressed by this section include:

- What new data objects and/or database schema changes is this going to require?
- What database migrations will accompany this change.
- How will the initial set of new data objects be generated, for example if you need to take into account existing instances, or modify other existing data describe how that will work.

19.2.3 REST API impact

Each API method which is either added or changed should have the following

- Specification for the method
  - A description of what the method does suitable for use in user documentation
  - Method type (POST/PUT/GET/DELETE)
  - Normal http response code(s)
  - Expected error http response code(s)
    - A description for each possible error code should be included describing semantic errors which can cause it such as inconsistent parameters supplied to the method, or when an instance is not in an appropriate state for the request to succeed. Errors caused by syntactic problems covered by the JSON schema definition do not need to be included.
  - URL for the resource
    - URL should not include underscores, and use hyphens instead.
  - Parameters which can be passed via the url
  - JSON schema definition for the request body data if allowed
    - Field names should use snake_case style, not CamelCase or MixedCase style.
  - JSON schema definition for the response body data if any
    - Field names should use snake_case style, not CamelCase or MixedCase style.
- Example use case including typical API samples for both data supplied by the caller and the response
- Discuss any policy changes, and discuss what things a deployer needs to think about when defining their policy.

Note that the schema should be defined as restrictively as possible. Parameters which are required should be marked as such and only under exceptional circumstances should additional parameters which are not defined in the schema be permitted (eg additionaProperties should be False).

Reuse of existing predefined parameter types such as regexps for passwords and user defined names is highly encouraged.

19.2.4 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 the 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.

- Can this change enable a resource exhaustion attack, such as allowing a single API interaction to consume significant server resources? Some examples of this include launching subprocesses for each connection, or entity expansion attacks in XML.

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.

19.2.5 Notifications impact

Please specify any changes to notifications. Be that an extra notification, changes to an existing notification, or removing a notification.

19.2.6 Other end user impact

Aside from the API, are there other ways a user will interact with this feature?

- Does this change have an impact on python-blazarclient? What does the user interface there look like?

19.2.7 Performance Impact

Describe any potential performance impact on the system, for example how often will new code be called, and is there a major change to the calling pattern of existing code.

Examples of things to consider here include:

- A small change in a utility function or a commonly used decorator can have a large impacts on performance.
- Calls which result in a database queries (whether direct or via conductor) can have a profound impact on performance when called in critical sections of the code.
- Will the change include any locking, and if so what considerations are there on holding the lock?

19.2.8 Other deployer impact

Discuss things that will affect how you deploy and configure OpenStack that have not already been mentioned, such as:

- What config options are being added? Should they be more generic than proposed (for example a flag that other hypervisor drivers might want to implement as well)? 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 directory name that instances are stored in, how do we handle instance directories created before the change landed? Do we move them? Do we have a special case in the code? Do we assume that the operator will recreate all the instances in their cloud?

19.2.9 Developer impact

Discuss things that will affect other developers working on OpenStack, such as:

- If the blueprint proposes a change to the driver API, discussion of how other hypervisors would implement the feature is required.

19.3 Implementation

19.3.1 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>

19.3.2 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.

19.4 Dependencies

- Include specific references to specs and/or blueprints in blazar, or in other projects, that this one either depends on or is related to.
- If this requires functionality of another project that is not currently used by Blazar (such as the glance v2 API when we previously only required v1), document that fact.
- Does this feature require any new library dependencies or code otherwise not included in OpenStack? Or does it depend on a specific version of library?

19.5 Testing

Please discuss the important scenarios needed to test here, as well as specific edge cases we should be ensuring work correctly. For each scenario please specify if this requires specialized hardware, a full openstack environment, or can be simulated inside the Blazar tree.

Please discuss how the change will be tested. We especially want to know what tempest tests will be added. It is assumed that unit test coverage will be added so that doesnt need to be mentioned explicitly, but discussion of why you think unit tests are sufficient and we dont need to add more tempest tests would need to be included.

Is this untestable in gate given current limitations (specific hardware / software configurations available)? If so, are there mitigation plans (3rd party testing, gate enhancements, etc).

19.6 Documentation Impact

Which audiences are affected most by this change, and which documentation titles on docs.openstack.org should be updated because of this change? Dont repeat details discussed above, but reference them here in the context of documentation for multiple audiences. For example, the Operations Guide targets cloud operators, and the End User Guide would need to be updated if the change offers a new feature available through the CLI or dashboard. If a config option changes or is deprecated, note here that the documentation needs to be updated to reflect this specifications change.

19.7 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 notes from a summit session
- Links to relevant research, if appropriate
- Related specifications as appropriate (e.g. if its an EC2 thing, link the EC2 docs)
- Anything else you feel it is worthwhile to refer to

19.8 History

Optional section intended to be used each time the spec is updated to describe new design, API or any database schema updated. Useful to let reader understand whats happened along the time.

Table 1: Revisions

Release Name	Description
Queens	Introduced

CHAPTER

TWENTY

MONITOR STATES OF RESOURCES

https://blueprints.launchpad.net/blazar/+spec/resource-monitoring

States of leased/reserved resources are not tracked currently. This spec proposes a new feature that monitors the states of leased/reserved resources and heals the lease from failure conditions.

20.1 Problem description

A lease owner cannot use leased resources if the resources go down, even though the lease was successfully created. To make matters worse, the lease owner cannot even notice such a failure condition from the lease object because Blazar does not track states of leased/reserved resources.

20.1.1 Use Cases

- Lease owners expects leased resources are available if the status of the lease is fine.
- Admins who have a privilege of resource operations, e.g. CRUD for /v1/hosts, expects Blazar to notify them about failures of resources in the pool.

20.2 Proposed change

20.2.1 Overview

Blazar monitors the states of resources, and heals damaged leases.

Monitor states of resources

Have the blazar-manager monitor states of resources. At least one of the following two types of monitoring features should be supported for each resource plugin.

1. Push-based monitoring

   The blazar-manager listens to notification messages sent by other components, e.g. sent by Nova for the host/instance plugins. And it picks up messages which refer to the resources managed by Blazar, i.e. a record of the resource is stored in the Blazar database. A publisher ID and topics to subscribe are provided by each resource plugin.

2. Polling-based monitoring

   The blazar-manager periodically calls a states check method of each resource plugin. Then, the resource plugin checks states of resources, e.g. *List Hypervisors* of the Compute API is used for the host/instance plugins. If any failure is detected, it is notified to the blazar-manager.

The blazar-manager starts monitoring after loading resource plugins. A type of monitoring feature is configurable for each resource plugin.

Update states of resources

If any failure is detected, the blazar-manager calls the resource plugin to handle it. The resource plugin updates the field named *reservable* in the record of the failure resource to *FALSE*. The record here means the record stored in the resource table of the Blazar database, e.g. the record of the computehosts table for the host/instance plugins.

Even though the computehosts table has a field named *status*, we do not use this field because it leads to misunderstanding that the *status* field copies the status of the hypervisor in Nova. Instead, we introduce a new boolean field named *reservable* which is decided by the combination of the state and the status of the hypervisor in Nova. Only if the state is *up* and the status is *enabled*, the *reservable* field becomes *TRUE*. i.e. it becomes *FALSE* if the state becomes *down* or the status becomes *disabled*.

Heal damaged leases

The following boolean flags are introduced for expressing resource conditions. They are set to False by default:

- Lease
  - degraded
- Reservation
  - missing_resources
  - resources_changed

After updating the status (= *reservable* field) of the failed resource, the resource plugin discovers leases which suffer from the failure, and tries to heal the leases. In the healing process, the resource plugin looks for available resources which satisfy requirements of the lease. Then, it takes the following actions:

- Case 1. Lease is not started yet
  - Case 1-1. Enough resources are available

    Reserve new resources instead of failure resources. Although resources are changed, keep the *resources_changed* flag False because it does not affect the lease from the lease owner perspective if it is not started yet.
  - Case 1-2. Alternative resource is not available

    Set the *degraded* flag of the lease and the *missing_resources* flag of the reservation to True.
- Case 2. Lease has been already started
  - Case 2-1. Enough resources are available:

    Set the *degraded* flag of the lease and the *resources_changed* flag of the reservation to True.
  - Case 2-2. Alternative resource is not available

    Set the *degraded* flag of the lease and the *missing_resources* flag of the reservation to True.

Once the *degraded* and *missing_resources* flags are set True, they are kept True even if the failed resource is recovered. To make them False, the lease owner sends an update lease request and requested capacity have to be satisfied. Note that the *resources_changed* flag never becomes False once it becomes True. In this case, the *degraded* flag never becomes False, neither.

From the architectural view, resource-dependent tables like the computehosts, computehost_allocations, computehost_reservations and instance_reservations tables are updated by the resource plugin. General tables like leases and reservations tables are updated by the manager.

Notifications

The blazar-mangaer publishes notifications if states of leases, reservations or resources are changed.

20.2.2 Alternatives

Other monitoring services like Monasca can be used instead. However, it is not a good solution in terms of cross-components dependencies. Dependencies should be reduced as much as possible.

20.2.3 Data model impact

- The leases table: a new boolean field *degraded* is added.
- The reservations table: new boolean fields *missing_resources* and *resource_changed* are added.
- The computehosts table: a new boolean field *reservable* is added.

20.2.4 REST API impact

- With the data model changes described above, some fields included in the REST API response body will be changed a little.

  New fields of the response of GET /v1/<lease-id>:

```
{
  "lease": {
    "degraded": false,
    "reservations": [
      {
        "missing_resources": false,
        "resources_changed": false,
      }
    ],
  }
}
```

20.2.5 Security impact

None.

20.2.6 Notifications impact

The blazar-manager sends the following two new notifications:

- Lease status change: notifies changes of the status of lease and reservations included in the lease.
- Resource status change: notifies the change of the status of the resource which is managed by Blazar. i.e. Notifies the change of the *reservable* field in the resource table of the Blazar database.

20.2.7 Other end user impact

None.

20.2.8 Performance Impact

None.

20.2.9 Other deployer impact

- New configuration options related to the monitoring feature like a type of monitoring, publisher ID and topics to subscribe will be added for each resource plugin.

20.2.10 Developer impact

All resource plugins (including new plugins supported in the future) have to support at least one type of resource monitoring feature.

20.3 Implementation

20.3.1 Assignee(s)

Primary assignee: hiro-kobayashi

20.3.2 Work Items

1. Define complete set of states of a lease and a reservation. This will be done by the state-machine blueprint[1].
2. Implement the monitoring mechanism into the blazar-manager.
3. Change the schema of the computehosts table. Concretely, remove the *status* field and add a new *reservable* field.
4. Change resource look-up features, e.g. _matching_hosts() method for the host plugin and the pickup_hosts() method for the instance plugin, to care for the *reservable* field of the record in the computehosts table.
5. Implement a resource specific monitoring feature called by the blazar-manager into each resource plugin. Focus on a push-based monitoring feature of the host plugin first.
6. Implement the lease-healing feature into each resource plugin.
7. Implement the notifications feature.
8. Change the DevStack setup script to enable the monitoring feature
9. Write a user guide

20.4 Dependencies

- Possible states of a lease and a reservation depend on the state-machine blueprint[1].

20.5 Testing

- Test the monitoring feature.
- Test the lease-healing feature.

20.6 Documentation Impact

- This new feature should be described in the introduction document.
- The installation guide will be updated to mention how to setup the monitoring feature.
- API references will be updated because the response body will be changed a little.

20.7 References

- [1] state-machine blueprint <https://blueprints.launchpad.net/blazar/+spec/state-machine>
- [2] Discussion at the Denver PTG <https://etherpad.openstack.org/p/blazar-resource-monitoring>

20.8 History

Table 1: Revisions

Release Name	Description
Queens	Introduced

CHAPTER

TWENTYONE

DEFINE STATE MACHINES

https://blueprints.launchpad.net/blazar/+spec/state-machine

Define state machines for leases, reservations, and events, and use the status field of the leases table which is unused for now.

21.1 Problem description

Statuses of leases, reservations, and events cannot properly describe some actual statuses because state machines for leases, reservations, and events are not well defined for now. It causes inconsistency of statuses, unrecoverable failures and so on.

Actually, currently leases have no status. Even though the leases table has a status field, it is not used. It should be used to describe the status of a lease properly.

21.1.1 Use Cases

- A lease owner can see the exact status of the lease in the lease parameters included in the request response.

21.2 Proposed change

21.2.1 Overview

Define state machines of leases, reservations, and events as follows.

Lease statuses

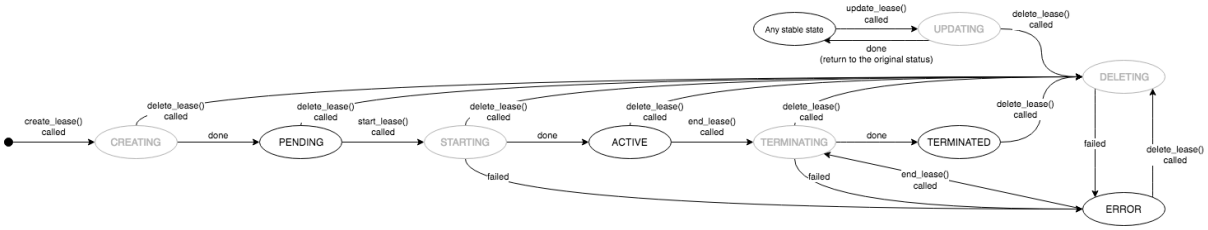
Lease statuses are categorized into 2 types: stable or transitional. In the state machine shown below, stable statuses are drawn as black nodes while transitional statuses are drawn as gray nodes. Stable statuses change to transitional statuses when a specific method of the blazar manager is called. After the method has completed or failed, the transitional statuses change to stable statuses.

A lease has the following four stable statuses:

- **PENDING**: A lease has been successfully created and is ready to start. The lease stays in this status until it starts.
- **ACTIVE**: A lease has been started and is active.
- **TERMINATED**: A lease has been successfully terminated.
- **ERROR**: Unrecoverable failures happened to the lease.

Transitional statuses are as follows:

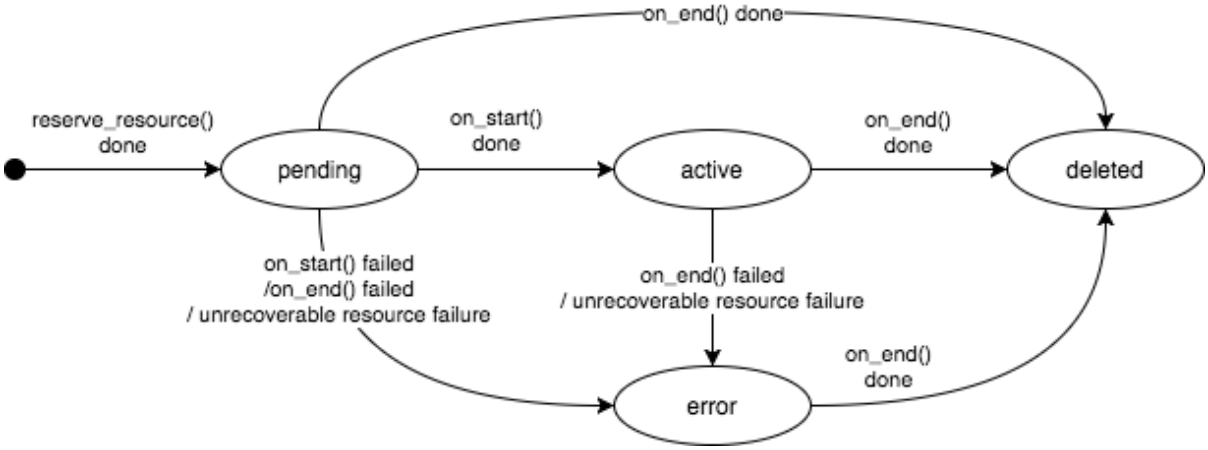
- **CREATING**: A lease is being created.
- **STARTING**: A lease is being started.
- **UPDATING**: A lease is being updated.
- **TERMINATING**: A lease is being terminated.
- **DELETING**: A lease is being deleted. Any status can change to this status because delete is the highest prioritized operation. e.g. when a lease hangs up in the STARTING status, delete should be allowed.



Reservation statuses

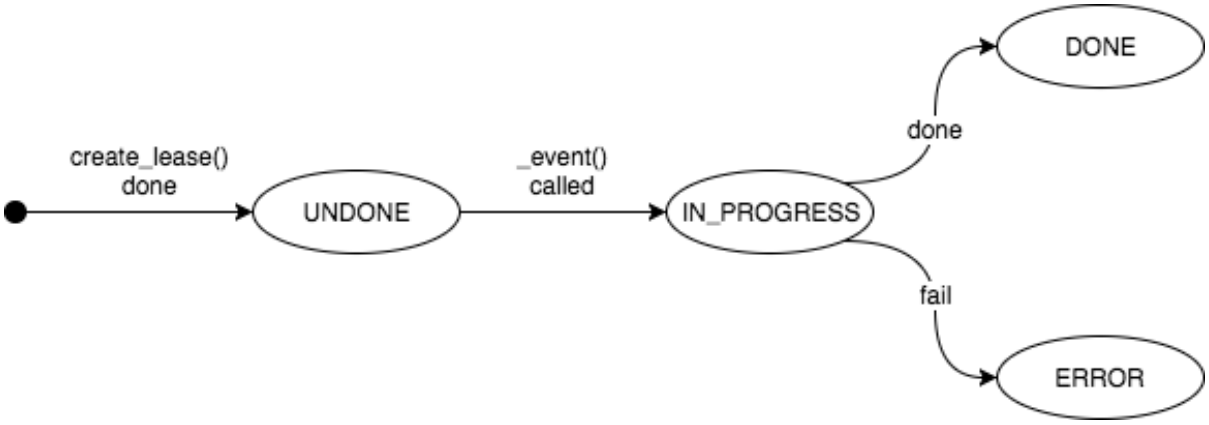
A reservation has the following four statuses. Small letters are used for backward compatibility:

- **pending**: A reservation has been successfully created and is ready to start. The reservation stays in this status until it starts.
- **active**: A reservation has been started and is active.
- **deleted**: Reserved resources have been successfully released.
- **error**: Unrecoverable failures happened to resources.



Event statuses

Event statuses are not changed.



Relationships between statuses

The following table shows conditions of statuses of reservations and events that have to be satisfied for each lease status.

Lease	Reservations	Events
CREATING	pending	start_lease: UNDONE , end_lease: UNDONE
PENDING	pending	start_lease: UNDONE , end_lease: UNDONE
STARTING	pending or active or error	start_lease: IN_PROGRESS , end_lease: UNDONE
ACTIVE	active	start_lease: DONE , end_lease: UNDONE
TERMINATING	active or deleted or error	start_lease: DONE , end_lease: IN_PROGRESS
TERMINATED	deleted	start_lease: DONE , end_lease: DONE
DELETING	Any status	Any status
UPDATING	Any status	Any status other than IN_PROGRESS

21.2.2 Alternatives

Express resource capacity sufficiency as a lease status like *_DEGRADED statuses and a reservation status like *_MISSING_RESOURCES and *_RESOURCES_CHANGED. The problem of this solution is that it complicates state machines. Instead, we will introduce boolean flags like *degraded* to leases and reservations for expressing such resource capacity sufficiency. See the resource-monitoring spec[1] in detail.

21.2.3 Data model impact

None

21.2.4 RESTAPI impact

None

21.2.5 Security impact

None

21.2.6 Notifications impact

None

21.2.7 Other end user impact

- Users can see the lease status.

21.2.8 Performance Impact

None

21.2.9 Other deployer impact

None

21.2.10 Developer impact

None

21.3 Implementation

21.3.1 Assignee(s)

Primary assignee:
hiro-kobayashi

21.3.2 Work Items

- Implement LeaseStatus, ReservationStatus and EventStatus class that contain statuses and basic methods for managing these statuses.
- Implement a decorator that checks/updates the lease status before/after *-lease methods of manager.
- Decorate *-lease methods with the decorator.

21.4 Dependencies

None

21.5 Testing

- Test status transitions.

21.6 Documentation Impact

None

21.7 References

- [1] resource-monitoring blueprint: https://blueprints.launchpad.net/blazar/+spec/resource-monitoring

21.8 History

Table 1: Revisions

Release Name	Description
Queens	Introduced

CHAPTER

TWENTYTWO

NEW INSTANCE RESERVATION

https://blueprints.launchpad.net/blazar/+spec/new-instance-reservation

Telecom operators want to keep instance slots for varied reasons, such as planned scale-out, maintenance works, Disaster Recovery, etc., and for high priority VNF service on hypervisors in a specific time window in order to accept an expected workload increase in their network, such as big events, sports games, etc. On the other hand, they also need to keep some free instance slots or hypervisors for non-planned scale-out against unexpected workload increases, such as bursty traffic.

Public cloud users often get the impression of unlimited resources in OpenStack, like instances and hypervisors, because of the scale of public cloud providers, but the resources are in reality limited. Operators need to handle the inconsistency.

22.1 Problem description

Some problems are well described in the Capacity Management development proposal[1]. Please check the story for details of the problems.

22.1.1 Use Cases

As Wei the project owner of a Telco operator, I want to specify my resource usage request for planned events. Some examples of time-based usage requests:

- I plan to use up to 60 vCPUs and 240GB of RAM from 06/01/2017 to 08/14/2017.
- I want guaranteed access to 4 instances with 1 vCPU, 1GB of RAM, and 10GB of disk. This example is similar to what would be described in the VNFD[2].

22.2 Proposed change

Blazar enables users to specify the amount of instances of a particular flavor. As described in use cases, users who reserve instances specify a tuple of amount of instances and a flavor definition of the instances. A flavor definition contains three pieces of information: the number of vcpus, the amount of RAM, and the size of the disk.

A basic idea and a sequence of the change follows:

1. A tenant user creates its reservation, in Blazar terms called a lease, with a set of time frame, definitions of a flavor for reserved instances, and how many instances.
2. Blazar issues a lease id to the user if the total amount of instances defined in the request (e.g. size of flavor times number of instances) is less than the amount of unused capacity for reservations in the time frame. Blazar leases can contain multiple reservations. Blazar checks whether the unused capacity can accommodate all reservations or not. If not, Blazar does not issue a lease id.

3. The user creates its instances via the Nova API with the reservation id. The request with the id is only accepted within the reservation time frame.
4. Nova creates the instance onto the hypervisor that Blazar marked as having capacity for the reservation.

To realize the sequence, this BP introduces a new resource plugin virtual:instance for the Blazar project. The plugin will be implemented in two phases because of the following reasons.

22.2.1 Short-term goal

With respect to affinity and anti-affinity rules, instance reservation only supports anti-affinity rule reservation because affinity rule reservation has already been achieved by host reservation. Affinity rule reservation by host reservation feature is not an ideal goal. For the data center usage efficiency, host reservation is not a good choice because a total amount of resources in a reservation is usually less than one hypervisor spec. It results in unused instance slots in the reserved hypervisors.

On the other hand, a hypervisor in the OpenStack cluster must accept total amount of instances in one reservation, it is equal to instance size times instance number, as affinity rule reservation. So the host reservation feature that is already implemented can handle instance reservation with affinity rule.

Prerequisites:

- The following three scheduler configuration values must be defined in nova.conf to use instance reservation:
  - AggregateInstanceExtraSpecsFilter
  - AggregateMultiTenancyIsolationFilter
  - ServerGroupAntiAffinityFilter

For the anti-affinity rule, Blazar will do the following steps:

0. As a preparation, Blazar adds filter_tenant_id=blazar-user-id to the freepool aggregate to prevent non-reservation instances from being scheduled into the freepool.
1. A tenant user creates their reservation, in Blazar terms called a lease, with a time frame, the instance size, and how many instances.

   One reservation in Blazar terms represents a tuple of <flavor, number of instances with the flavor> and one lease can have multiple reservations. Thus one lease can have multiple instance types.
2. Blazar checks whether the reservation is acceptable during the time frame or not. If acceptable, Blazar records the reservation request in its database and updates hypervisor usage in the freepool. Then Blazar returns the reservation id. If not, Blazar responds that the reservation is not acceptable and provides additional information to the tenant, e.g. the number of instances reserved is greater than the instance quota.
3. At the start time of the reservation, Blazar creates a server group, a flavor, and a host aggregate related to the reservation. Then it adds the hypervisors onto which reserved instances are scheduled to the aggregate.

   The tricks Blazar is doing here are:

   - create server group with anti-affinity policy
   - create a flavor with two extra_specs, is_public=False and flavor access rights to the user. The extra_specs are aggregate_instance_extra_specs:reservations:<reservation-id> and affinity_id:<server-group-id>

- create a new host aggregate with above aggregate_instance_extra_specs and filter_tenant_id of the requesting users project id
- does not bring out the hypervisor from the freepool because other users reservations also use other instance slots in the hypervisor

4. **The user fetches the server_group id by calling the flavor show API in**
   Nova, then creates reserved instances with a scheduling hint, like hint group=group-id, and the newly created flavor.

Scheduling mechanism in Nova

Blazar manages some host aggregates to handle instance scheduling in Nova. Blazar expects Nova to schedule instances as follows for non-reserved instances (usual instances), instances related to host reservation, and instances related to instance reservation:

- non-reserved instances: scheduled to hypervisors which are outside of both the freepool aggregate and reservation-related aggregates.
- instances related to host reservation: scheduled to hypervisors which are inside the reservation-related aggregate. The hypervisors are not included in the freepool aggregate.
- instances related to instance reservation: scheduled to hypervisors which are inside the reservation-related aggregate. The hypervisors are included in the freepool aggregate.

Nova filters used by Blazar choose hypervisors with the following rules:

- AggregateInstanceExtraSpecsFilter picks up hypervisors from the aggregate related to an instance reservation based on extra_specs of the flavor, if the request is related to instance reservation. If not, the filter picks up hypervisors from neither reservation-related aggregates nor the freepool.
- BlazarFilter picks up hypervisors from the aggregate related to a host reservation based on the reservation scheduler hint, if the request is related to host reservation. If not, the filter picks up hypervisors from neither host reservation-related aggregates nor the freepool.
- AggregateMultiTenancyIsolationFilter blocks requests to be scheduled to the freepool by users who do not have active reservation.
- Combination of AggregateInstanceExtraSpecsFilter and AggregateMultiTenancyIsolationFilter enables requests using instance reservation to be scheduled in the corresponding aggregate.
- ServerGroupAntiAffinityFilter ensures instance reservation related instances are spread on different hypervisors.

Summary of short term goal

- Use the host reservation function for an affinity rule reservation.
- Use the new instance reservation function for an anti-affinity rule reservation.
- Create reserved instances with a reserved flavor and a scheduling hint.

22.2.2 Long-term goal

Instance reservation supports both affinity rule and anti-affinity rule.

The affinity rule reservation allows other instances or reservation to use unused instance slots in reserved hypervisors. The Nova team is developing placement API[1]. The API already has custom resource

classes[2] and is now implementing a scheduler function[3] that uses custom resources classes. It enables operator to more efficiently manage hypervisors in the freepool.

Blazar will do the following steps:

1. A tenant user creates their reservation, in term of Blazar called lease, with a time frame, the instance size, and how many instances.
2. Blazar checks whether the reservation is acceptable during the time frame or not. If acceptable, Blazar records the reservation request in its database and updates the usage of hypervisor in freepool. Then Blazar returns the reservation id. If not, Blazar responds the reservation is not acceptable.
3. At the start time of the reservation, Blazar creates a custom resource class, a flavor, and a resource provider of the custom resource class.
4. The user creates reserved instances with the newly created flavor.

Some functionality of the placement API is under implementation. Once the development is finished, the Blazar team will start using the placement API.

22.2.3 Alternatives

This feature could be achieved on the Blazar side or on the Nova side.

Blazar side approach

- one reservation represents one instance

In the above sequence, a tenant user creates a reservation configured only with the instance size (e.g. flavor), reserving only one instance.

While it could technically work for users, they would need to handle a large number of reservations at client side when they would like to use many instances. The use case shows users would like to create multiple instances for one reservation.

Nova side approach

- Pre-block the slots by stopped instances

A user creates as many instances as they want to reserve, then stops them until start time. It would work from a user perspective.

On the other hand, from a cloud provider perspective, it is hard to accept this method of reservation. Stopped instances keep holding hypervisor resources, like vCPUs, for instances while they are stopped. It means cloud providers need to plan their hypervisor capacity to accept the total amount of usage of future reservations. For example, if all users reserve their instance for one year in advance, cloud providers need to plan hypervisors that can accept the total amount of instances reserved in the next year.

Of course, we do not prevent users from stopping their instances: users can call the stop API for their own reason and cloud provider bill them a usage fee for the hypervisor slot usage. However, from NFV motivations, telecom operators cannot prepare and deploy hypervisors with a large enough capacity to accommodate future usage demand in advance.

- Prepared images for the reservation by shelved instances

A user creates as many instances as they want to reserve, then shelves them until start time. It would work from a cloud provider perspective: shelved instances release their hypervisor slot, so the problem described earlier in the stopped instance solution would not happen.

On the other hand, from the user perspective, some problems could happen. As described in motivation section, VNF applications need affinity or anti-affinity rule for placement of their instances. Nova has a server group API for the affinity and anti-affinity placement, but it does not ensure the required amount of instances can be located on the same host. Similarly, it does not ensure the required amount of instances can be accommodated by hypervisors when hypervisors slots are consumed by others.

Of course, cloud providers should usually plan enough resources to accommodate user requests. However, it is hard to plan enough hypervisors to make the cloud look like unlimited resources in NFV use cases. Requiring a very large number of spare hypervisors is not realistic.

22.2.4 Data model impact

A new table, called instance_reservations, is introduced in the Blazar database. The instance reservation feature uses the existing computehost_allocations table to store allocation information. Usage of the table is as follows:

1. In the create lease/reservation, Blazar queries hosts that are used for instance reservations or are not used by any reservations during the reservation time window.
2. If some hosts are already used for instance reservations, Blazar checks that the reserved instances could be allocated onto the hosts.
3. If some hosts are not used by any reservation, Blazar adds a mapping of the reservation to computehost as computehost_allocations table.
4. For the host reservation, the current design will never pick hosts which have a mapping, a reservation to hosts, during the reservation time window, so instance reservation does not impact host reservation queries.

The table has size of reserved flavor, vcpu, memory size in MB and disk size in GB, amount of instances created with the flavor, and an affinity flag.

```
CREATE TABLE instance_reservations (
    id VARCHAR(36) NOT NULL,
    reservation_id VARCHAR(255) NOT NULL,
    vcpus INT UNSIGNED NOT NULL,
    memory_mb INT UNSIGNED NOT NULL,
    disk_gb INT UNSIGNED NOT NULL,
    amount INT UNSIGNED NOT NULL,
    affinity BOOLEAN NOT NULL,
    flavor_id VARCHAR(36),
    aggregate_id INT,
    server_group_id VARCHAR(36),

    PRIMARY key (id),
    INDEX (id, reservation_id)
    FOREIGN KEY (reservation_id)
      REFERENCES reservations(id)
      ON DELETE CASCADE,
);
```

In the short term goal, the affinity flag only supports False since instance reservation only supports anti-affinity rule. The plugin manages multiple types of Nova resources. The mappings with each resources to column data as follows:

- In the db * reservations.resource_id is equal to instance_reservations.id
- With Nova resources
  - flavor id is equal to reservations.id
    - the extra_spec for scheduling, aggregate_instance_extra_specs, is equal to prefix+reservations.id
  - aggregate name is equal to reservations.id
    - the metadata for scheduling is equal to prefix+reservations.id
  - server_group id is recorded in extra_spec of the flavor. This id will be removed in the long term goal, as it is better encapsulated in the Nova API.

22.2.5 REST API impact

- URL: POST /v1/leases
  - Introduce new resource_type virtual:instance for a reservation

Request Example:

```
{
  "name": "instance-reservation-1",
  "reservations": [
    {
      "resource_type": "virtual:instance",
      "vcpus": 4,
      "memory_mb": 4096,
      "disk_gb": 10,
      "amount": 5,
      "affinity": false
    }
  ],
  "start": "2017-05-17 09:07"
  "end": "2017-05-17 09:10",
  "events": []
}
```

Response Example:

```
{
  "lease": {
    "reservations": [
      {
        "id": "reservation-id",
        "status": "pending",
        "lease_id": "lease-id-1",
        "resource_id": "resource_id",
        "resource_type": "virtual:instance",
        "vcpus": 4,
        "memory_mb": 4096,
        "disk_gb": 10,
```

(continues on next page)

(continued from previous page)

```
        "amount": 5,
        "affinity": false,
        "created_at": "2017-05-01 10:00:00",
        "updated_at": "2017-05-01 11:00:00",
      }],
    "snippet": "..."
  }
}
```

- URL: GET /v1/leases
- URL: GET /v1/leases/{lease-id}
- URL: PUT /v1/leases/{lease-id}
- URL: DELETE /v1/leases/{lease-id}
  - The change is the same as POST /v1/leases

22.2.6 Security impact

None

22.2.7 Notifications impact

None

22.2.8 Other end user impact

python-blazarclient needs to support resource reservations of type virtual:instance in lease handling commands.

22.2.9 Performance Impact

None

22.2.10 Other deployer impact

The freepool that is used in physical:host plugin is also used by the virtual:instance plugin if the deployer activates the new plugin.

22.2.11 Developer impact

None

22.3 Implementation

22.3.1 Assignee(s)

Primary assignee:
muroi-masahito

Other contributors:
None

22.3.2 Work Items

- Create the new table in blazar
- Create instance reservation plugin
- Change reservation_pool.py and nova_inventory.py to be more generic since both host_plugin and instance_plugin will use these classes
- Change BlazarFilter to pass hosts which are in instance reservation aggregates if the reservations extra_spec is specified.
- Add instance reservation supports in python-blazarclient
- Add scenario tests in gate job, mainly Tempest job

22.4 Dependencies

For the long term goal, the Placement API needs to support custom resource classes and a mechanism to use them for Nova scheduling.

22.5 Testing

- The following scenarios should be tested:
  - Creating an anti-affinity reservation and verify all instances belonging to the reservation are scheduled onto different hosts.
  - Verify that both host reservation and instance reservation pick hosts from the same freepool and that Blazar coordinates all reservations correctly.

22.6 Documentation Impact

- API reference

22.7 References

1. Capacity Management development proposal: http://git.openstack.org/cgit/openstack/development-proposals/tree/development-proposals/proposed/capacity-management.rst
2. VNFD: http://www.etsi.org/deliver/etsi_gs/NFV-IFA
3. Placement API: https://docs.openstack.org/developer/nova/placement.html
4. Custom Resource Classes: https://specs.openstack.org/openstack/nova-specs/specs/ocata/implemented/custom-resource-classes.html
5. Custom Resource Classes Filter: http://specs.openstack.org/openstack/nova-specs/specs/pike/approved/custom-resource-classes-in-flavors.html

22.8 History

Table 1: Revisions

Release Name	Description
Pike	Introduced

CHAPTER

TWENTYTHREE

ADD OPTIONS FOR ON-END ACTION

https://blueprints.launchpad.net/blazar/+spec/on-end-options

For the host reservation feature, we propose to change the behavior at the end of a lease to force-delete running instances for avoiding failure of following leases. In addition, we propose to add options for the action taken before the force-deletion. Of course the owner of a lease is also free to request update lease prior to the lease end time to extend the lease.

23.1 Problem description

Currently, the physical host reservation feature does not terminate running instances at the end of lease period. That can cause failures of following leases. The solution is to delete the instances by default and provide options for the action before the deletion, e.g. snapshot.

23.1.1 Use Cases

- All users want Blazar to be punctual for guaranteeing fairness for all leases.
- A user wants to snapshot the instances of the lease if the workload does not finish in the lease period. Then, the user will resume the snapshotted instances and restart the workload at the next lease period.
- A user wants to use instances just temporarily and wants those instances to be auto-deleted at the end of the lease period.

23.2 Proposed change

Change the on_end() method of the PhysicalHostPlugin class to delete running instances at the end of the lease period by default.

Furthermore, support options for the action taken before the deletion. For this purpose, extend the existing before_end_lease event which is used only for the before_end notification for now. We plan to support the following actions for the before_end_lease event:

- notification: Notify the lease owner that the lease will end soon. (Currently supported)
- snapshot: Take snapshots of running instances before deletion.
- migration: Migrate running instances.

We expect other options will be proposed in the future.

The before_end_lease event is registered and changed when creating and updating a lease. A default before_end_lease action and the time triggering the event can be configured. In addition, users can specify them through request parameters.

23.2.1 Alternatives

Use NOT the before_end_lease BUT the end_lease event for the actions like snapshot. The end_lease event is triggered at the end of a lease for now. Change that to be triggered at the specific time window prior to the end of a lease. Make the length of the time window configurable.

This change may complicate event handling because the end_lease event can trigger multiple actions, e.g., take snapshot and then delete the instance, while the before_end_lease solution keeps one-event-to-one-action relationship. Therefore, we prefer the before_end_lease solution.

23.2.2 Data model impact

- A new attribute before_end_action will be added to the reservation table.

23.2.3 REST API impact

- Plan to update only v2 API and not to support before_end_lease related capabilities for v1 API.
- URL: POST /v2/leases
  - Add a new parameter before_end_action
  - Change the parameter before_end_lease to before_end_date. This change is for aligning the terminology with the other parameters.

    Example:

```
{
    "name": "lease_foo",
    "start_date": "2017-3-21 15:00",
    "end_date": "2017-03-24 15:00",
    "before_end_date": "2017-3-24 14:00",
    "reservations": [
        {
            "resource_id": "1234",
            "min": 1,
            "max": 3,
            "resource_type": "physical:host",
            "hypervisor_properties": "[\">=\", \"$memory_mb\", \
↪"4096\"]",
            "before_end_action": "snapshot"
        }
    ],
    "events": []
}
```

- URL: PUT /v2/leases

  Same changes as for the POST request.

23.2.4 Security impact

None.

23.2.5 Notifications impact

None.

23.2.6 Other end user impact

None.

23.2.7 Performance Impact

None.

23.2.8 Other deployer impact

- New config options will be added in the `physical:host` group:
  - before_end_action: Configure default action for the before_end_lease.
  - hours_before_end_lease: Configure default hours (prior to the end_lease) that triggers the before_end_lease action.

23.2.9 Developer impact

None.

23.3 Implementation

23.3.1 Assignee(s)

Primary assignee:
hiro-kobayashi

23.3.2 Work Items

- STEP1: Change the behavior at the end of a lease to force-delete running instances.
- STEP2: Change the before_end_lease event to support configurable actions.
- STEP3: Change REST APIs.

23.4 Dependencies

None.

23.5 Testing

Add the following tests:

- Unit tests
  - STEP1: Check all running instances being deleted at the end of lease.
  - STEP2: Check the before_end_lease action being triggered and completed.
  - STEP3: Check the new parameters being correctly processed.

23.6 Documentation Impact

Update the following documents:

- Add a release note
- Blazar REST API docs (API v2 docs will be auto updated.)

23.7 References

- Discussion log:
  - http://eavesdrop.openstack.org/meetings/blazar/2017/blazar.2017-03-07-09.00.log.html
  - http://eavesdrop.openstack.org/meetings/blazar/2017/blazar.2017-03-21-09.03.log.html

23.8 History

Table 1: Revisions

Release Name	Description
Pike	Introduced

CHAPTER

TWENTYFOUR

EXAMPLE SPEC - THE TITLE OF YOUR BLUEPRINT

Include the URL of your launchpad blueprint:

https://blueprints.launchpad.net/blazar/+spec/example

Introduction paragraph why are we doing anything? A single paragraph of prose that operators can understand. The title and this first paragraph should be used as the subject line and body of the commit message respectively.

Some notes about the blazar-spec and blueprint process:

- Not all blueprints need a spec.
- The aim of this document is first to define the problem we need to solve, and second agree the overall approach to solve that problem.
- This is not intended to be extensive documentation for a new feature. For example, there is no need to specify the exact configuration changes, nor the exact details of any DB model changes. But you should still define that such changes are required, and be clear on how that will affect upgrades.
- You should aim to get your spec approved before writing your code. While you are free to write prototypes and code before getting your spec approved, its possible that the outcome of the spec review process leads you towards a fundamentally different solution than you first envisaged.
- But, API changes are held to a much higher level of scrutiny. As soon as an API change merges, we must assume it could be in production somewhere, and as such, we then need to support that API change forever. To avoid getting that wrong, we do want lots of details about API changes upfront.

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/blazar/+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
- For help with syntax, see http://sphinx-doc.org/rest.html
- To test out your formatting, build the docs using tox and see the generated HTML file in doc/build/html/specs/<path_of_your_file>
- 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.

- If your specification proposes any changes to the Blazar REST API such as changing parameters which can be returned or accepted, or even the semantics of what happens when a client calls into the API, then you should add the APIImpact flag to the commit message. Specifications with the APIImpact flag can be found with the following query:

24.1 Problem description

A detailed description of the problem. What problem is this blueprint addressing?

24.1.1 Use Cases

What use cases does this address? What impact on actors does this change have? Ensure you are clear about the actors in each use case: Developer, End User, Deployer etc.

24.2 Proposed change

Here is where you cover the change you propose to make in detail. How do you propose to solve this problem?

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?

At this point, if you would like to just get feedback on if the problem and proposed change fit in blazar, you can stop here and post this for review to get preliminary feedback. If so please say: Posting to get preliminary feedback on the scope of this spec.

24.2.1 Alternatives

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.

24.2.2 Data model impact

Changes which require modifications to the data model often have a wider impact on the system. The community often has strong opinions on how the data model should be evolved, from both a functional and performance perspective. It is therefore important to capture and gain agreement as early as possible on any proposed changes to the data model.

Questions which need to be addressed by this section include:

- What new data objects and/or database schema changes is this going to require?
- What database migrations will accompany this change.
- How will the initial set of new data objects be generated, for example if you need to take into account existing instances, or modify other existing data describe how that will work.

24.2.3 REST API impact

Each API method which is either added or changed should have the following

- Specification for the method
  - A description of what the method does suitable for use in user documentation
  - Method type (POST/PUT/GET/DELETE)
  - Normal http response code(s)
  - Expected error http response code(s)
    - A description for each possible error code should be included describing semantic errors which can cause it such as inconsistent parameters supplied to the method, or when an instance is not in an appropriate state for the request to succeed. Errors caused by syntactic problems covered by the JSON schema definition do not need to be included.
  - URL for the resource
    - URL should not include underscores, and use hyphens instead.
  - Parameters which can be passed via the url
  - JSON schema definition for the request body data if allowed
    - Field names should use snake_case style, not CamelCase or MixedCase style.
  - JSON schema definition for the response body data if any
    - Field names should use snake_case style, not CamelCase or MixedCase style.
- Example use case including typical API samples for both data supplied by the caller and the response
- Discuss any policy changes, and discuss what things a deployer needs to think about when defining their policy.

Note that the schema should be defined as restrictively as possible. Parameters which are required should be marked as such and only under exceptional circumstances should additional parameters which are not defined in the schema be permitted (eg additionaProperties should be False).

Reuse of existing predefined parameter types such as regexps for passwords and user defined names is highly encouraged.

24.2.4 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 the 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.

- Can this change enable a resource exhaustion attack, such as allowing a single API interaction to consume significant server resources? Some examples of this include launching subprocesses for each connection, or entity expansion attacks in XML.

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.

24.2.5 Notifications impact

Please specify any changes to notifications. Be that an extra notification, changes to an existing notification, or removing a notification.

24.2.6 Other end user impact

Aside from the API, are there other ways a user will interact with this feature?

- Does this change have an impact on python-blazarclient? What does the user interface there look like?

24.2.7 Performance Impact

Describe any potential performance impact on the system, for example how often will new code be called, and is there a major change to the calling pattern of existing code.

Examples of things to consider here include:

- A small change in a utility function or a commonly used decorator can have a large impacts on performance.
- Calls which result in a database queries (whether direct or via conductor) can have a profound impact on performance when called in critical sections of the code.
- Will the change include any locking, and if so what considerations are there on holding the lock?

24.2.8 Other deployer impact

Discuss things that will affect how you deploy and configure OpenStack that have not already been mentioned, such as:

- What config options are being added? Should they be more generic than proposed (for example a flag that other hypervisor drivers might want to implement as well)? 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 directory name that instances are stored in, how do we handle instance directories created before the change landed? Do we move them? Do we have a special case in the code? Do we assume that the operator will recreate all the instances in their cloud?

24.2.9 Developer impact

Discuss things that will affect other developers working on OpenStack, such as:

- If the blueprint proposes a change to the driver API, discussion of how other hypervisors would implement the feature is required.

24.3 Implementation

24.3.1 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>

24.3.2 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.

24.4 Dependencies

- Include specific references to specs and/or blueprints in blazar, or in other projects, that this one either depends on or is related to.
- If this requires functionality of another project that is not currently used by Blazar (such as the glance v2 API when we previously only required v1), document that fact.
- Does this feature require any new library dependencies or code otherwise not included in OpenStack? Or does it depend on a specific version of library?

24.5 Testing

Please discuss the important scenarios needed to test here, as well as specific edge cases we should be ensuring work correctly. For each scenario please specify if this requires specialized hardware, a full openstack environment, or can be simulated inside the Blazar tree.

Please discuss how the change will be tested. We especially want to know what tempest tests will be added. It is assumed that unit test coverage will be added so that doesnt need to be mentioned explicitly, but discussion of why you think unit tests are sufficient and we dont need to add more tempest tests would need to be included.

Is this untestable in gate given current limitations (specific hardware / software configurations available)? If so, are there mitigation plans (3rd party testing, gate enhancements, etc).

24.6 Documentation Impact

Which audiences are affected most by this change, and which documentation titles on docs.openstack.org should be updated because of this change? Dont repeat details discussed above, but reference them here in the context of documentation for multiple audiences. For example, the Operations Guide targets cloud operators, and the End User Guide would need to be updated if the change offers a new feature available through the CLI or dashboard. If a config option changes or is deprecated, note here that the documentation needs to be updated to reflect this specifications change.

24.7 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 notes from a summit session
- Links to relevant research, if appropriate
- Related specifications as appropriate (e.g. if its an EC2 thing, link the EC2 docs)
- Anything else you feel it is worthwhile to refer to

24.8 History

Optional section intended to be used each time the spec is updated to describe new design, API or any database schema updated. Useful to let reader understand whats happened along the time.

Table 1: Revisions

Release Name	Description
Pike	Introduced

CHAPTER

TWENTYFIVE

TERMINATE LEASE AT ANY TIME

https://blueprints.launchpad.net/blazar/+spec/terminate-lease-at-anytime

Enable lease termination at any time even if the lease has already started.

25.1 Problem description

Blazar does not allow any leases to be deleted if they have already started. Though it is possible to change the end time of a lease by the lease update request with the appropriate end_date parameter, a more intuitive operation for immediate lease termination should be provided.

25.1.1 Use Cases

- As Wei, I want to be able to query/update/terminate a resource usage request at any point in time. (from the capacity management user story)

25.2 Proposed change

Support two ways for the lease termination.

1. Lease termination by the update lease request with end_date = now

   Change the update_lease() method of the ManagerService class to accept a request with the end_date parameter equal to now. Then, the update_lease() method calls the on_end() method of resource plugins for terminating the lease.

2. Lease termination by the delete lease request

   Change the delete_lease() method of the ManagerService class to accept a request even if the lease has been already started. Then, the update_lease() method calls the on_end() method of resource plugins and delete the entry of the lease from the Blazar DB.

25.2.1 Alternatives

None.

25.2.2 Data model impact

None.

25.2.3 REST API impact

- URL: PUT /<version>/leases/<id>

  Allow the end_date parameter to be now.

25.2.4 Security impact

None.

25.2.5 Notifications impact

None.

25.2.6 Other end user impact

None.

25.2.7 Performance Impact

None.

25.2.8 Other deployer impact

None.

25.2.9 Developer impact

None.

25.3 Implementation

25.3.1 Assignee(s)

Primary assignee:
hiro-kobayashi

25.3.2 Work Items

- Change the update_lease() method of the ManagerService class.
- Change the delete_lease() method of the ManagerService class.

25.4 Dependencies

Depends on the on-end-options blueprint because it changes the on_end behavior of resource plugins which are called by the update_lease() and delete_lease() method. This terminate-lease-at-anytime blueprint should be implemented after the force-deletion feature of the on-end-options blueprint is implemented.

25.5 Testing

- Check a lease can be terminated by the update lease request with the end_date equal to now.
- Check a lease can be terminated and deleted by the delete lease request even if it has already been started.

25.6 Documentation Impact

None.

25.7 References

- on-end-options blueprint
- Capacity management user story

25.8 History

Table 1: Revisions

Release Name	Description
Pike	Introduced

CHAPTER

TWENTYSIX

UPDATE A CAPACITY OF RESERVED RESOURCE

https://blueprints.launchpad.net/blazar/+spec/update-reserved-capacity

Support updating the capacity of an existing reservation.

26.1 Problem description

The start date and the end date of a lease can be updated through the update lease request. However, the capacity of reserved resource cannot be changed once the reservation is created for now. The capacity should be able to be changed for improving the flexibility of resource usage requests.

26.1.1 Use Cases

- As Wei, I want to be able to query/update/terminate a resource usage request at any point in time. (Required in the capacity management development proposal[1])

26.2 Proposed change

The update_reservation() method of a resource plugin currently checks only the *start_date* and *end_date* of the request body. Change it to check other parameters, e.g., min, max, hypervisor_properties and resource_properties for the host plugin. And enable the update_reservation() method to update allocations.

The update_reservation() succeeds if **all** of the request parameters can be satisfied. Otherwise, it raises exceptions.

First target is the host plugin. For the host plugin, min, max, hypervisor_properties and resource_properties can be updated if an update request satisfies all of following conditions:

- Enough resources are available for the new request.
- Any host does not removed from the aggregate associated with the lease if the lease has already started. This condition is needed for preventing unexpected deletion and error of instances on the reserved host.

Otherwise, Blazar returns an error and nothing is updated.

26.2.1 Alternatives

None.

26.2.2 Data model impact

None.

26.2.3 REST API impact

- Users send a update-lease request with some parameters that they want to update.

26.2.4 Security impact

None.

26.2.5 Notifications impact

None.

26.2.6 Other end user impact

None.

26.2.7 Performance Impact

The resource allocation algorithm of resource plugins can be more complex. So the performance impact should be carefully tested.

26.2.8 Other deployer impact

None.

26.2.9 Developer impact

Developers of new resource plugins should consider this capability for the update_reservation() method.

26.3 Implementation

26.3.1 Assignee(s)

Primary assignee:
hiro-kobayashi

26.3.2 Work Items

- Change the update_reservation() method of the host plugin.

26.4 Dependencies

None

26.5 Testing

- Adds unit tests of update capacity request for the update_reservation() method
- Adds scenario test of update capacity request

26.6 Documentation Impact

Write a release note.

26.7 References

1. Capacity management development proposal: http://git.openstack.org/cgit/openstack/development-proposals/tree/development-proposals/proposed/capacity-management.rst

26.8 History

Table 1: Revisions

Release Name	Description
Pike	Introduced