## **Zuul Variables**

Zuul supplies not only the variables specified by the job definition to Ansible, but also some variables from Zuul itself.

When a pipeline is triggered by an action, it enqueues items which may vary based on the pipeline’s configuration. For example, when a new change is created, that change may be enqueued into the pipeline, while a tag may be enqueued into the pipeline when it is pushed.

Information about these items is available to jobs. All of the items enqueued in a pipeline are git references, and therefore share some attributes in common. But other attributes may vary based on the type of item.

**zuul**

All items provide the following information as Ansible variables under the zuul key:

**zuul.artifacts   
Type:*list***

If the job has a job.requires attribute, and Zuul has found changes ahead of this change in the pipeline with matching job.provides attributes, then information about any artifacts returned from those jobs will appear here.

This value is a list of dictionaries with the following format:

**zuul.artifacts[].project**

The name of the project which supplied this artifact.

**zuul.artifacts[].change**

The change number which supplied this artifact.

**zuul.artifacts[].patchset**

The patchset of the change.

**zuul.artifacts[].job**

The name of the job which produced the artifact.

**zuul.artifacts[].name**

The name of the artifact (as supplied to Returning artifact URLs).

**zuul.artifacts[].url**

The URL of the artifact (as supplied to Returning artifact URLs).

**zuul.artifacts[].metadata**

The metadata of the artifact (as supplied to Returning artifact URLs).

**zuul.build**

The UUID of the build. A build is a single execution of a job. When an item is enqueued into a pipeline, this usually results in one build of each job configured for that item’s project. However, items may be re-enqueued in which case another build may run. In dependent pipelines, the same job may run multiple times for the same item as circumstances change ahead in the queue. Each time a job is run, for whatever reason, it is acompanied with a new unique id.

**zuul.buildset**

The build set UUID. When Zuul runs jobs for an item, the collection of those jobs is known as a buildset. If the configuration of items ahead in a dependent pipeline changes, Zuul creates a new buildset and restarts all of the jobs.

**zuul.child\_jobs**

A list of the first level dependent jobs to be run after this job has finished successfully.

**zuul.ref**

The git ref of the item. This will be the full path (e.g., refs/heads/master or refs/changes/…).

**zuul.override\_checkout**

If the job was configured to override the branch or tag checked out, this will contain the specified value. Otherwise, this variable will be undefined.

**zuul.pipeline**

The name of the pipeline in which the job is being run.

**zuul.post\_review   
Type:*bool***

Whether the current job is running in a post-review pipeline or not.

**zuul.job**

The name of the job being run.

**zuul.event\_id**

The UUID of the event that triggered this execution. This is mainly useful for debugging purposes.

**zuul.voting**

A boolean indicating whether the job is voting.

**zuul.attempts**

An integer count of how many attempts have been made to run this job for the current buildset. If there are pre-run failures or network connectivity issues then previous attempts may have been cancelled, and this value will be greater than 1.

**zuul.max\_attempts**

The number of attempts that will be be made for this job when encountering an error in a pre-playbook before it is reported as failed. This value is taken from job.attempts.

**zuul.ansible\_version**

The version of the Ansible community package release used for executing the job.

**zuul.project**

The item’s project. This is a data structure with the following fields:

**zuul.project.name**

The name of the project, excluding hostname. E.g., org/project.

**zuul.project.short\_name**

The name of the project, excluding directories or organizations. E.g., project.

**zuul.project.canonical\_hostname**

The canonical hostname where the project lives. E.g., git.example.com.

**zuul.project.canonical\_name**

The full canonical name of the project including hostname. E.g., git.example.com/org/project.

**zuul.project.src\_dir**

The path to the source code relative to the work dir. E.g., src/git.example.com/org/project.

**zuul.projects   
Type:*dict***

A dictionary of all projects prepared by Zuul for the item. It includes, at least, the item’s own project. It also includes the projects of any items this item depends on, as well as the projects that appear in job.required-projects.

This is a dictionary of dictionaries. Each value has a key of the canonical\_name, then each entry consists of:

**zuul.projects{}.name**

The name of the project, excluding hostname. E.g., org/project.

**zuul.projects{}.short\_name**

The name of the project, excluding directories or organizations. E.g., project.

**zuul.projects{}.canonical\_hostname**

The canonical hostname where the project lives. E.g., git.example.com.

**zuul.projects{}.canonical\_name**

The full canonical name of the project including hostname. E.g., git.example.com/org/project.

**zuul.projects{}.src\_dir**

The path to the source code, relative to the work dir. E.g., src/git.example.com/org/project.

**zuul.projects{}.required**

A boolean indicating whether this project appears in the job.required-projects list for this job.

**zuul.projects{}.checkout**

The branch or tag that Zuul checked out for this project. This may be influenced by the branch or tag associated with the item as well as the job configuration.

**zuul.projects{}.checkout\_description**

A human-readable description of why Zuul chose this particular branch or tag to be checked out. This is intended as a debugging aid in the case of complex jobs. The specific text is not defined and is subject to change.

**zuul.projects{}.commit**

The hex SHA of the commit checked out. This commit may appear in the upstream repository, or if it the result of a speculative merge, it may only exist during the run of this job.

For example, to access the source directory of a single known project, you might use:

{{ zuul.projects['git.example.com/org/project'].src\_dir }}

To iterate over the project list, you might write a task something like:

- name: Sample project iteration

debug:

msg: "Project {{ item.name }} is at {{ item.src\_dir }}

with\_items: {{ zuul.projects.values() | list }}

**zuul.playbook\_context   
Type:*dict***

This dictionary contains information about the execution of each playbook in the job. This may be useful for understanding exactly what playbooks and roles Zuul executed.

All paths herein are located under the root of the build directory (note that is one level higher than the workspace directory accessible to jobs on the executor).

**zuul.playbook\_context{}.playbook\_projects   
Type:*dict***

A dictionary of projects that have been checked out for playbook execution. When used in the trusted execution context, these will contain only merged commits in upstream repositories. In the case of the untrusted context, they may contain speculatively merged code.

The key is the path and each value is another dictionary with the following keys:

**zuul.playbook\_context{}.playbook\_projects{}.canonical\_name**

The canonical name of the repository.

**zuul.playbook\_context{}.playbook\_projects{}.checkout**

The branch or tag checked out.

**zuul.playbook\_context{}.playbook\_projects{}.commit**

The hex SHA of the commit checked out. As above, this commit may or may not exist in the upstream repository depending on whether it was the result of a speculative merge.

**zuul.playbook\_context{}.playbooks   
Type:*list***

An ordered list of playbooks executed for the job. Each item is a dictionary with the following keys:

**zuul.playbook\_context{}.playbooks[].path**

The path to the playbook.

**zuul.playbook\_context{}.playbooks[].roles   
Type:*list***

Information about the roles available to the playbook. The actual role path supplied to Ansible is the concatenation of the role\_path entry in each of the following dictionaries. The rest of the information describes what is in the role path.

In order to deal with the many possible role layouts and aliases, each element in the role path gets its own directory. Depending on the contents and alias configuration for that role repo, a symlink is added to one of the repo checkouts in zuul.playbook\_context.playbook\_projects so that the role may be supplied to Ansible with the correct name.

**zuul.playbook\_context{}.playbooks[].roles[].checkout**

The branch or tag checked out.

**zuul.playbook\_context{}.playbooks[].roles[].checkout\_description**

A human-readable description of why Zuul chose this particular branch or tag to be checked out. This is intended as a debugging aid in the case of complex jobs. The specific text is not defined and is subject to change.

**zuul.playbook\_context{}.playbooks[].roles[].link\_name**

The name of the symbolic link.

**zuul.playbook\_context{}.playbooks[].roles[].link\_target**

The target of the symbolic\_link.

**zuul.playbook\_context{}.playbooks[].roles[].role\_path**

The role path passed to Ansible.

**zuul.tenant**

The name of the current Zuul tenant.

**zuul.timeout**

The job timeout, in seconds.

**zuul.post\_timeout**

The post-run playbook timeout, in seconds.

**zuul.jobtags**

A list of tags associated with the job. Not to be confused with git tags, these are simply free-form text fields that can be used by the job for reporting or classification purposes.

**zuul.items   
Type:*list***

**Note**

zuul.items conflicts with the items() builtin so the variable can only be accessed with python dictionary like syntax, e.g: zuul['items']

A list of dictionaries, each representing an item being tested with this change with the format:

**zuul.items[].project**

The item’s project. This is a data structure with the following fields:

**zuul.items[].project.name**

The name of the project, excluding hostname. E.g., org/project.

**zuul.items[].project.short\_name**

The name of the project, excluding directories or organizations. E.g., project.

**zuul.items[].project.canonical\_hostname**

The canonical hostname where the project lives. E.g., git.example.com.

**zuul.items[].project.canonical\_name**

The full canonical name of the project including hostname. E.g., git.example.com/org/project.

**zuul.items[].project.src\_dir**

The path to the source code on the remote host, relative to the home dir of the remote user. E.g., src/git.example.com/org/project.

**zuul.items[].branch**

The target branch of the change (without the refs/heads/ prefix).

**zuul.items[].bundle\_id**

The id of the bundle if the change is in a circular dependency cycle.

**zuul.items[].change**

The identifier for the change.

**zuul.items[].change\_url**

The URL to the source location of the given change. E.g., https://review.example.org/#/c/123456/ or https://github.com/example/example/pull/1234.

**zuul.items[].patchset**

The patchset identifier for the change. If a change is revised, this will have a different value.

**zuul.items[].resources**

**zuul.items[].:type: dict**

A job using a container build resources has access to a resources variable that describes the resource. Resources is a dictionary of group keys, each value consists of:

**zuul.items[].:type: dict.namespace**

The resource’s namespace name.

**zuul.items[].:type: dict.context**

The kube config context name.

**zuul.items[].:type: dict.pod**

The name of the pod when the label defines a kubectl connection.

Project or namespace resources might be used in a template as:

- **hosts**: localhost

tasks:

- **name**: Create a k8s resource

**k8s\_raw**:

**state**: present

**context**: "{{ zuul.resources['node-name'].context }}"

**namespace**: "{{ zuul.resources['node-name'].namespace }}"

Kubectl resources might be used in a template as:

- **hosts**: localhost

tasks:

- **name**: Copy src repos to the pod

**command**: >

oc rsync -q --progress=false

{{ zuul.executor.src\_root }}/

{{ zuul.resources['node-name'].pod }}:src/

no\_log: true

**zuul\_success**

Post run playbook(s) will be passed this variable to indicate if the run phase of the job was successful or not. This variable is meant to be used with the bool filter.

**tasks**:

- **shell**: echo example

**when**: zuul\_success | bool

**zuul\_will\_retry**

Post run and cleanup playbook(s) will be passed this variable to indicate if the job will be retried. This variable is meant to be used with the bool filter.

**tasks**:

- **shell**: echo example

**when**: zuul\_will\_retry | bool

**nodepool**

Information about each host from Nodepool is supplied in the nodepool host variable. Availability of values varies based on the node and the driver that supplied it. Values may be null if they are not applicable.

**nodepool.label**

The nodepool label of this node.

**nodepool.az**

The availability zone in which this node was placed.

**nodepool.cloud**

The name of the cloud in which this node was created.

**nodepool.provider**

The name of the nodepool provider of this node.

**nodepool.region**

The name of the nodepool provider’s region.

**nodepool.host\_id**

The cloud’s host identification for this node’s hypervisor.

**nodepool.external\_id**

The cloud’s identifier for this node.

**nodepool.slot**

If the node supports running multiple jobs on the node, a unique numeric ID for the subdivision of the node assigned to this job. This may be used to avoid build directory collisions.

**nodepool.interface\_ip**

The best IP address to use to contact the node as determined by the cloud provider and nodepool.

**nodepool.public\_ipv4**

A public IPv4 address of the node.

**nodepool.private\_ipv4**

A private IPv4 address of the node.

**nodepool.public\_ipv6**

A public IPv6 address of the node.

**nodepool.private\_ipv6**

A private IPv6 address of the node.

### **Change Items**

A change to the repository. Most often, this will be a git reference which has not yet been merged into the repository (e.g., a gerrit change or a GitHub pull request). The following additional variables are available:

**zuul.branch**

The target branch of the change (without the refs/heads/ prefix).

**zuul.change**

The identifier for the change.

**zuul.patchset**

The patchset identifier for the change. If a change is revised, this will have a different value.

**zuul.change\_url**

The URL to the source location of the given change. E.g., https://review.example.org/#/c/123456/ or https://github.com/example/example/pull/1234.

**zuul.message**

The commit or pull request message of the change base64 encoded. Use the b64decode filter in ansible when working with it.

**zuul.change\_message**

The commit or pull request message of the change. When Zuul runs Ansible, this variable is tagged with the !unsafe YAML tag so that Ansible will not interpolate values into it. Note, however, that the inventory.yaml file placed in the build’s workspace for debugging and inspection purposes does not inclued the !unsafe tag.

**zuul.commit\_id**

The git sha of the change. This may be the commit sha of the current patchset revision or the tip of a pull request branch depending on the source. Because of Zuul’s speculative merge process, this commit may not even appear in the prepared git repos, so it should not be relied upon for git operations in jobs. It is included here to support interfacing with systems that identify a change by the commit.

### **Branch Items**

This represents a branch tip. This item may have been enqueued because the branch was updated (via a change having merged, or a direct push). Or it may have been enqueued by a timer for the purpose of verifying the current condition of the branch. The following additional variables are available:

**zuul.branch**

The name of the item’s branch (without the refs/heads/ prefix).

**zuul.oldrev**

If the item was enqueued as the result of a change merging or being pushed to the branch, the git sha of the old revision will be included here. Otherwise, this variable will be undefined.

**zuul.newrev**

If the item was enqueued as the result of a change merging or being pushed to the branch, the git sha of the new revision will be included here. Otherwise, this variable will be undefined.

**zuul.commit\_id**

The git sha of the branch. Identical to newrev or oldrev if defined.

### **Tag Items**

This represents a git tag. The item may have been enqueued because a tag was created or deleted. The following additional variables are available:

**zuul.tag**

The name of the item’s tag (without the refs/tags/ prefix).

**zuul.oldrev**

If the item was enqueued as the result of a tag being deleted, the previous git sha of the tag will be included here. If the tag was created, this variable will be undefined.

**zuul.newrev**

If the item was enqueued as the result of a tag being created, the new git sha of the tag will be included here. If the tag was deleted, this variable will be undefined.

**zuul.commit\_id**

The git sha of the branch. Identical to newrev or oldrev if defined.

### **Ref Items**

This represents a git reference that is neither a change, branch, or tag. Note that all items include a ref attribute which may be used to identify the ref. The following additional variables are available:

**zuul.oldrev**

If the item was enqueued as the result of a ref being deleted, the previous git sha of the ref will be included here. If the ref was created, this variable will be undefined.

**zuul.newrev**

If the item was enqueued as the result of a ref being created, the new git sha of the ref will be included here. If the ref was deleted, this variable will be undefined.

**zuul.commit\_id**

The git sha of the branch. Identical to newrev or oldrev if defined.

### Working Directory

Additionally, some information about the working directory and the executor running the job is available:

**zuul.executor**

A number of values related to the executor running the job are available:

**zuul.executor.hostname**

The hostname of the executor.

**zuul.executor.src\_root**

The path to the source directory.

**zuul.executor.log\_root**

The path to the logs directory.

**zuul.executor.work\_root**

The path to the working directory.

**zuul.executor.inventory\_file**

The path to the inventory. This variable is needed for jobs running without a nodeset since Ansible doesn’t set it for localhost; see this porting guide.

The inventory file is only readable by jobs running in a trusted execution context.