Node Sharing Plugin

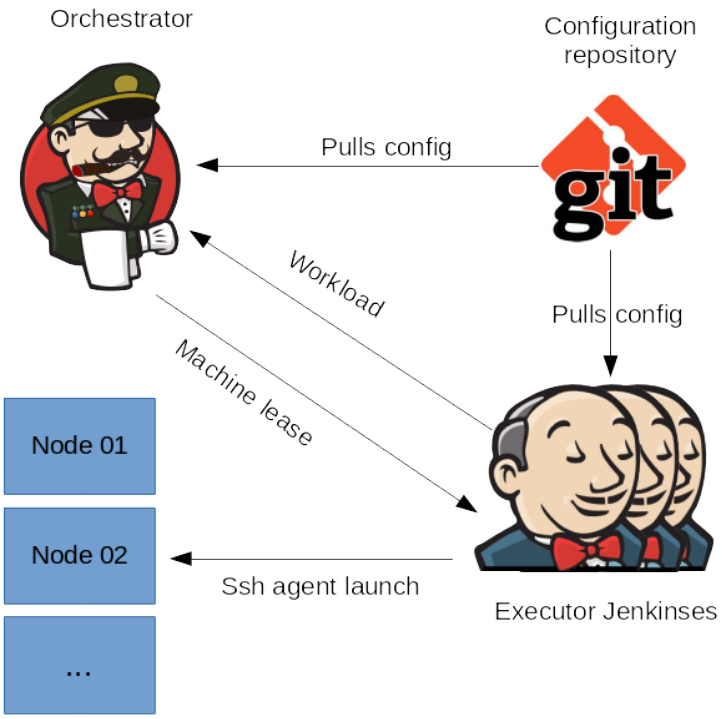
Plugin Information
View Node sharing executor on the plugin site for more information.

Share machines as Jenkins agents across multiple Jenkins masters.

Requirements

- The nodes are connected to the individual Jenkins masters so builds can be executed there as if those nodes would be good old Jenkins nodes.
- The node to use is determined by evaluating Jenkins labels.
- Nodes are use exclusively by individual Jenkins masters.
- In case there is no matching node available at the time build is scheduled the request will be queued and dispatched in FIFO fashion.

Basics



In order to facilitate fair sharing of agents, additional Jenkins master is needed to serve the role of an *Orchestrator*, leasing the *Nodes* to *Executor* Jenkinses.

The Executor Jenkins is an ordinary Jenkins with *Shared Nodes* cloud defined globally permitting it to lease, connect and utilize nodes from a shared pool. Such Jenkins can use static slaves or other clouds in the same time.

The Orchestrator Jenkins is a dedicated instance to instrument the sharing. It is expected not to have any static nodes, clouds or jobs defined as it is using the notion of computers, executors, builds and queue items to implement the sharing. It also serve the purpose of grid visualization UI for read only clients. It is not meant to be executing any other workload.

Config repository is a git based record of the current configuration of the pool the Orchestrator (and to some extend the Executors) takes as an input. Ideally, the Orchestrator should have no other user configuration except for what is in the config repository. In the name of reliability, the config repository hold the full list of Jenkins masters authorized to reserve from a particular pool. The repository, or the permission to commit there, is the access control mechanism for the pool management.

The Node is a host used for sharing. It have several representations in the Node Sharing Grid:

- Actual Jenkins agent connected by the cloud implementation to the Executor Jenkins
- The placeholder computer in Orchestrator.
  - Note this is not a real Jenkins computer capable of executing anything except for dummy reservation tasks. It does not even have a remoting channel open to the actual host despite it appears online.
- The record in config repository containing the node definition.

Setup

Config Repo

- Create git repository cloneable from both Orchestrator and Executor Jenkinses.
- Populate it with the grid definition.
  - See the example config repo or use the creation script to get started.

Orchestrator

- Create minimalistic Jenkins deployment
- Install `node-sharing-orchestrator` plugin including its dependencies
  - https://updates.jenkins.io/download/plugins/node-sharing-orchestrator/
- Start Jenkins with following java properties
  - `-Dcom.redhat.jenkins.nodesharingbackend.Pool.ENDPOINT=`*`cloneable_git_url_pointing_to_config_repo`*
  - `-Dcom.redhat.jenkins.nodesharingbackend.Pool.USERNAME=`*`name_of_the_REST_user`*
  - `-Dcom.redhat.jenkins.nodesharingbackend.Pool.PASSWORD=`*`password_of_the_REST_user`*
- Verify no Administrative Monitor warnings are issued once Jenkins is started

- Configure dedicated automation account to receive incoming REST calls granting it permission named `NodeSharing.Reserve`. No real user should be granted this permission. How to do this is specific to particular authorization strategy used.
    - Example Jenkins Configuration as Code definition:

```
unclassified:
  location:
    # Needs to be set correctly so orchestrator knows its own url
    url: https://ci.example.com/orchestrator

jenkins:
  numExecutors: 0
  quietPeriod: 0
  slaveAgentPort: -1
  securityRealm:
    local:
      users:
        - id: "admin"
          password: "secret"
        - id: "nodesharing"
          password: "secret too"
  # Set necessary permission for 'nodesharing' account as well for anonymous
  # user so executor users can use orchestrator as a dashboard
  authorizationStrategy:
    globalMatrix:
      grantedPermissions:
      - "Overall/Administer:admin"
      - "Overall/Read:anonymous"
      - "Job/Read:anonymous"
      - "Overall/Read:nodesharing"
      - "Job/Read:nodesharing"
      - "NodeSharing/Reserve:nodesharing"
```

    (Note it requires `configuration-as-code` and `configuration-as-code-support` plugins installed to execute, plus all the plugins needed by the declaration itself)

Executors

- Install `node-sharing-executor` plugin including its dependencies on instances to utilize the pool
    - https://updates.jenkins.io/download/plugins/node-sharing-executor/
- Note that node definitions in config repo can refer to further plugins that needs to be installed on executors too.
- Add *Shared Nodes* cloud specifying cloneable git url pointing to config repo and credentials for the rest user.
- Create SSH credentials used to connect the machines. Their ids needs to match those in config repo nodes.
- Configure dedicated automation account to receive incoming REST calls granting it permission named `NodeSharing.Reserve`. No real user should be granted this permission. How to do this is specific to particular authorization strategy used
- Example Jenkins Configuration as Code definition:

```
unclassified:
  location:
    # Needs to be set correctly to executor knows its own url
    url: https://ci.example.com/jenkins

jenkins:
  # Dedicated user needed to receive REST calls from orchestrator
  securityRealm:
    local:
      users:
        - id: "nodesharing"
          password: "nodesharing"
  authorizationStrategy:
    globalMatrix:
      grantedPermissions:
      - "Overall/Read:nodesharing"
      - "Job/Read:nodesharing"
      - "NodeSharing/Reserve:nodesharing"

  # Cloud hooking executor to the pool
  clouds:
    - nodeSharing:
        configRepoUrl: https://git.example.com/team/node-sharing-config-repo.git
        orchestratorCredentialsId: "node-sharing-rest-password-id"

credentials:
  system:
    # Any credentials we are referring to from node definitions
    domainCredentials:
    - credentials:
      - usernamePassword:
          scope: SYSTEM
          id: "node-sharing-rest-password-id"
          username: "nodesharing"
          password: "secret"
          description: "Rest credential for node sharing"
```

(Note it requires `configuration-as-code` and `configuration-as-code-support` plugins installed to execute, plus all the plugins needed by the declaration itself)