Lab: Distributed Jenkins builds

The goal of this exercise is to create the nodes and agents required that implement a scalable and secure distributed Jenkins instance.

The target is:

- Two ssh agents that handle builds.
- Each agent has one executor.
    - We have two CPUs available (a common guideline is that this can usually support two executors).
- The controller does not have any executors, so no builds run on the controller.
- Security must follow up.

Start by selecting the *Jenkins LTS (Long-Term Support) link from you lab's Home Page. Log into your Jenkins controller using the following credentials:

- Username: **butler**
- Password: **butler**

Deleting existing agents

Your lab instance is pre-configured with two agents. We will first delete those.

- Go to **Manage Jenkins Manage Nodes and Clouds**.
- You will see two agents — **jdk7-node** and **jdk-8-node**.

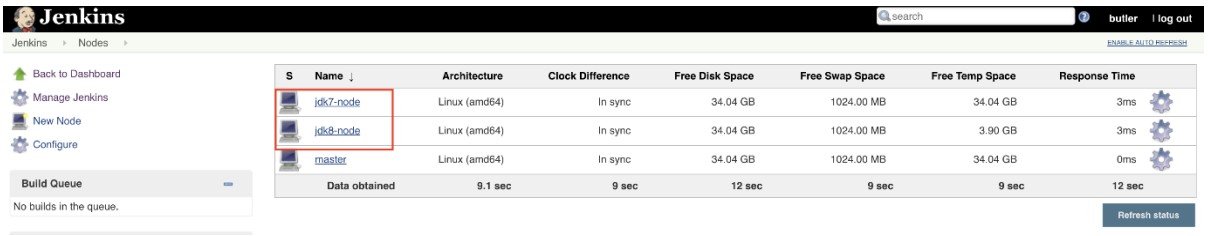


Figure 1. Manage Nodes and Clouds

- Click the down arrow next to **jdk7-node** and select **Delete Agent**.
- Click the **yes** button to confirm that you really want to delete the agent.

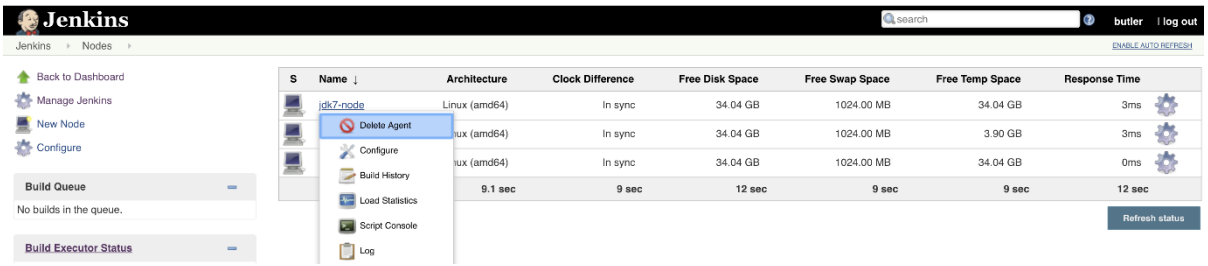


Figure 2. Delete jdk7-node from nodes list

- Repeat the steps above to delete the **jdk8-node** agent.

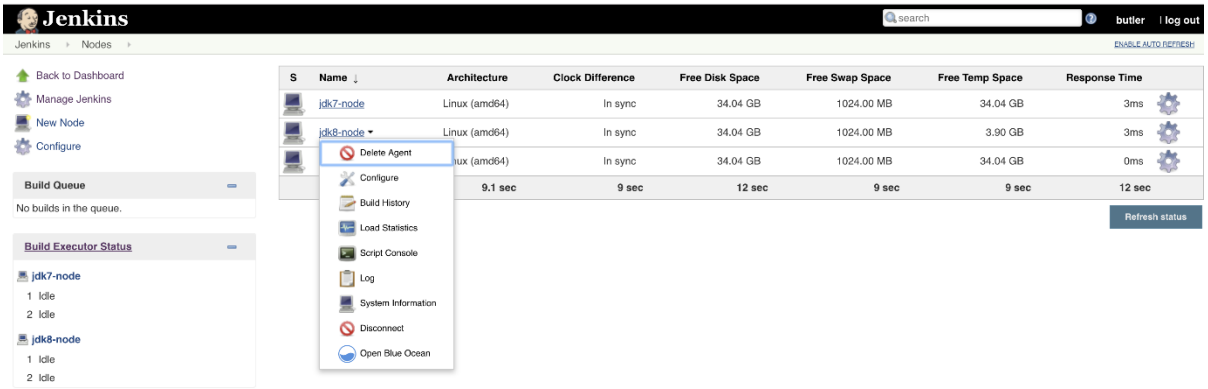


Figure 3. Delete jdk8-node from nodes list

Adding SSH agents

The Lab instance has an SSH "nodes" already running with these properties:

- Hostname: `jdk8-ssh-agent` and `jdk7-ssh-agent`
- Port: `22`
- Username: `jenkins`
- User authentication: RSA private Key
  - Key Passphrase: **No Passphrase**
  - Private Key location: In **Devbox**, in the file `/ssh-keys/vagrant_insecure_key`

A passphrase is different from a password. Passphrases are typically longer than passwords (for added security) and are most often used to control access to cryptographic systems.

Validating SSH

We are going to "validate" the SSH connection first:

- Spawn a shell in the Devbox.
- Use the **ssh** command to connect to the SSH node using previous settings.
- `$ docker exec -ti jenkins /bin/bash`
-
- `$ ssh -i /ssh-keys/vagrant_insecure_key -p22 \`
- `  jenkins@jdk8-ssh-agent \`
- `  echo "-- Hello from jdk8-ssh-agent"`
- `...`
- `Are you sure you want to continue connecting (yes/no)? yes`
- `...`
- `-- Hello from jdk8-ssh-agent`
-
- `$ ssh -i /ssh-keys/vagrant_insecure_key -p22 \`
- `  jenkins@jdk7-ssh-agent \`
- `  echo "-- Hello from jdk7-ssh-agent"`

- `...`
- `Are you sure you want to continue connecting (yes/no)? yes`
- `...`
  `-- Hello from jdk7-ssh-agent`

- Copy the contents of `/ssh-keys/vagrant_insecure_key` somewhere; we will use it later.

Adding the jdk8-ssh-agent to Jenkins

Sign in to Jenkins as `butler/butler` and go to the **Manage Jenkins Manage Nodes and Clouds** page.

You see that your Jenkins instance always has at least one node created. This is the built-in node used for the Jenkins controller itself.

Create a **New Node** by selecting **New Node** in the left frame.

Identify it with the following values:

- Name: `ssh-agent-1`
- Type: 'Permanent Agent'

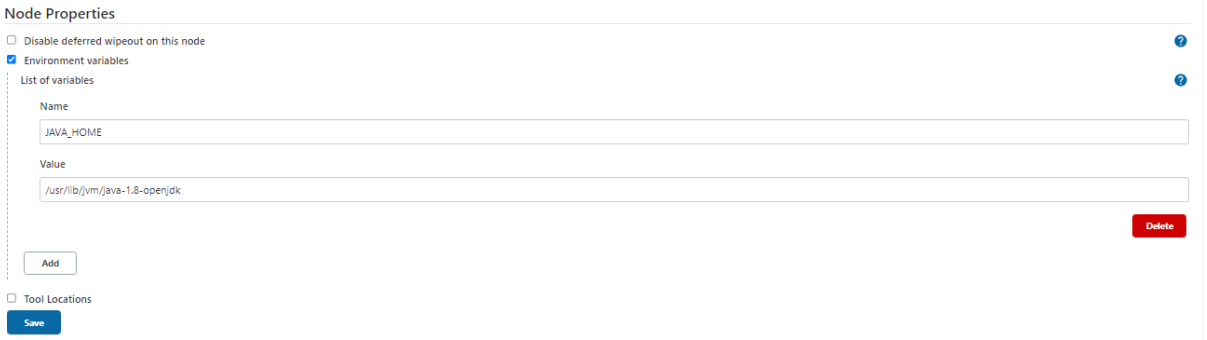
Select **OK**.

Configure it with these properties:

- # of executors: **2**
- Remote root directory: `/home/jenkins`
- No Labels
- Usage: **Use this node as much as possible**
  - Host is `jdk8-ssh-agent`
  - Set the Credentials field dropdown to "Jenkins (SSH Key for the Agent)"
  - Click the button **Advanced** to access the **Port** field, and set it to `22`
- **Host Key Verification Strategy**: `Non verifying Verification Strategy`
  - Under the Node Properties
    - Check the toggle for **Environment Variables**
    - Click **Add** to add a new environment variable
    - Enter `JAVA_HOME` for **Name** and
    - `/usr/lib/jvm/java-1.8-openjdk` for **Value**

In case of error and retry, use the **Trust SSH Host Key** button if it has changed

- Figure 4. SSH Agent 1 Configuration



- Figure 5. SSH Agent 1 Configuration

You can see that your node now appears in the node list, but it may have a little red cross because the agent is not yet started on this node or is experiencing troubles.

Browse to the Log Console of the node to see what is happening:

- Click on the Node `ssh-agent-1` on the list
- On the left-menu, click on the ***Log**
- See that, if you read the log, the Agent is now online

The agent took a few seconds to start. This is why you may have experienced the red cross.

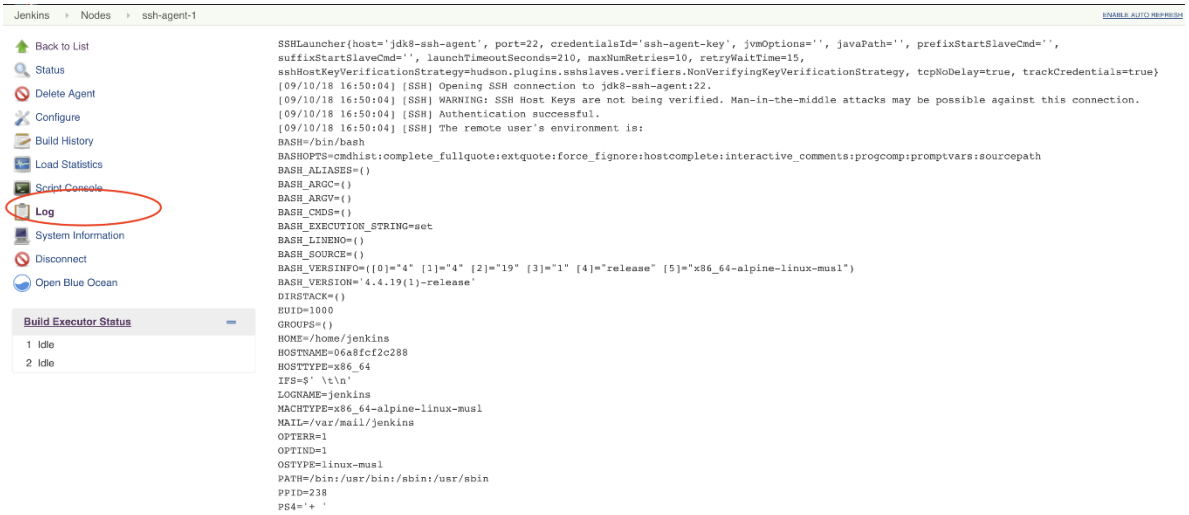


Figure 6. Jenkins SSH Agent 1 Online

Browse back to the Node list and see that the Agent is now fully online:

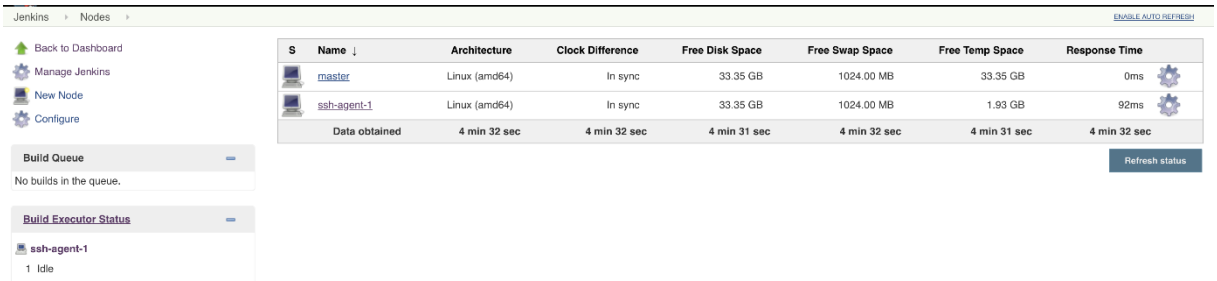


Figure 7. Jenkins SSH Agent 1 Online

Adding the jdk7-ssh-agent to Jenkins

Repeat the steps above to add the second ssh agent:

Create another new node by selecting **New Node** in the left frame.

Identify the new node with the following values:

- Name: `ssh-agent-2`
- Type: **Permanent Agent**

Configure it with these properties:

- # of executors: **2**
- Remote root directory: `/home/jenkins`
- No Labels
- Usage: **Utilize as much as possible**
- Launch method: **Launch agents via SSH** and configure it with previous properties
    - Host is `jdk7-ssh-agent`
    - Click the button **Advanced** to access the **Port** field, and set it to `22`
- **Host Key Verification Strategy**: `Non verifying Verification Strategy`
- Under the Node Properties:
    - Check the toggle for `Environment Variables`
    - Click **Add** to add a new environment variable
    - Enter `JAVA_HOME` for **Name** and
    - `/usr/lib/jvm/java-1.7-openjdk` for **Value**

Select **Save**.

You should now see the two agents when you browse the Node list:

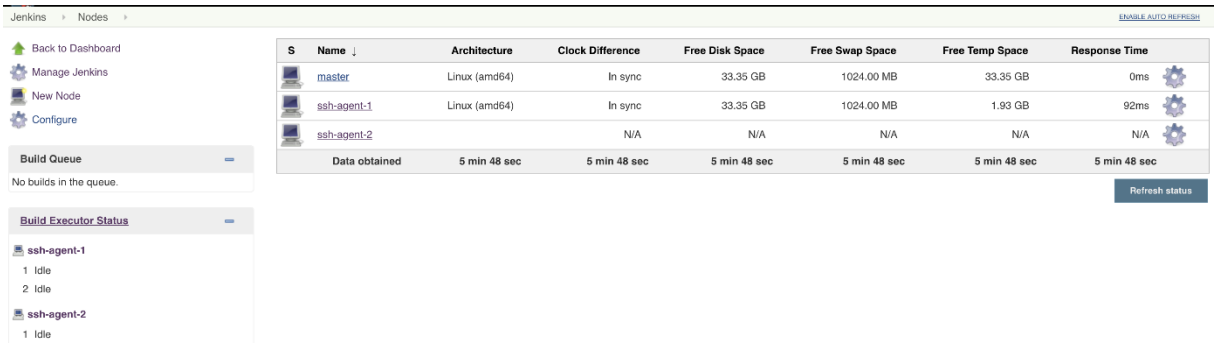


Figure 8. Jenkins SSH Agents

Reconfiguring the controller

It is now time to reconfigure the Jenkins controller, so that all builds run on the agents rather than on the controller.

To do this, go to **Manage Jenkins Configure System** and set the **# of executors** to `0`.

> It is a good practice to avoid having executors on the Jenkins controller. Any job running on the controller can access the JENKINS_HOME where it could leak data (such as credentials) or delete items accidentally.

Also set the following:

- Add a **Label** called `controller`.
- Set the **Usage** to **Only build jobs with label restrictions matching this node**.

Figure 9. Controller Global Configuration

- Save the configuration.

Now, browse to the **Manage Jenkins Configure Global Security** page:

- Ensure that **Enable Agent → Controller Security** is checked.
- Save the configuration.

Go back to the Jenkins dashboard. You should now see a total of four executors, two per agent:

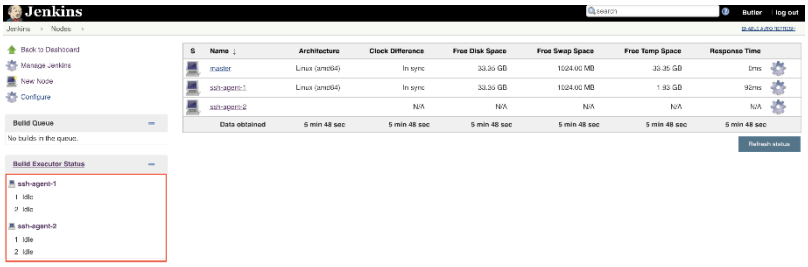


Figure 10. Jenkins Executors Count

Reconfiguring jobs

Now we can reconfigure the `pipeline-job` that was created in a previous exercise to use one of the new agents. You can access the configuration screen in either of two ways:

- Select the arrow to the immediate right of **pipeline-job** on the dashboard and select **Configure** from the drop-down menu.
- Select **pipeline-job** on the dashboard to open the page for the job and select **Configure** from the left frame.

Make the following configuration changes:

- Select the checkbox **Restrict where this project can be run**.
- Enter `ssh-agent-1` in the **Label Expression** field.

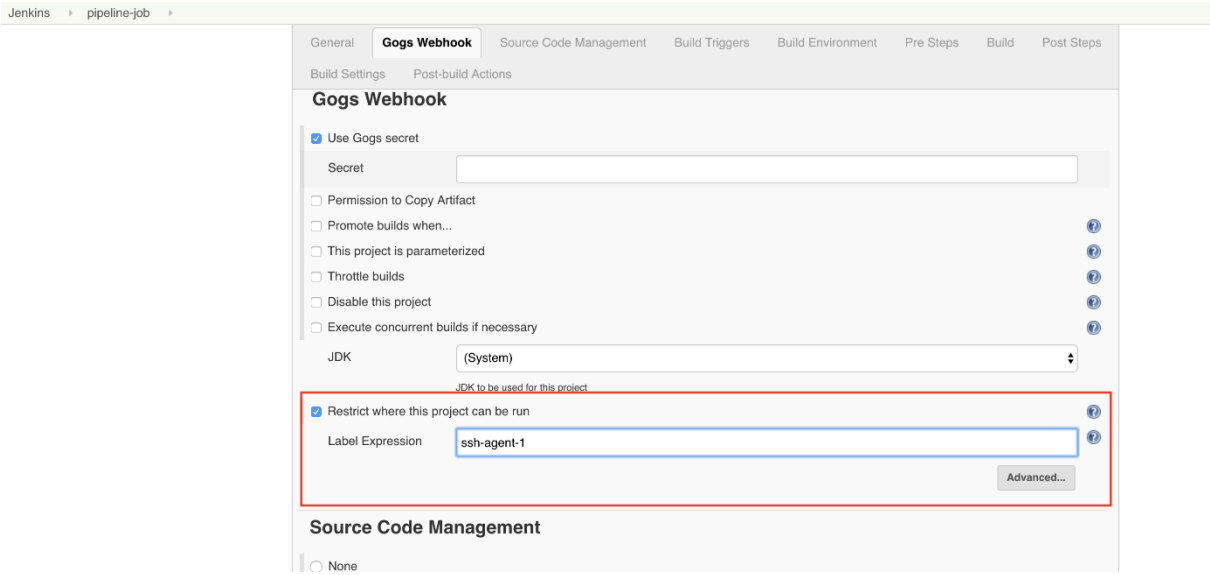


Figure 11. Restricting jobs to agent

- Select **Save** button to save the configuration.

Running distributed builds

It is now time to run a build to use one of the agents we created. Kick off a new build for `pipeline-job` and click **Console Output** to confirm that your build job ran on `ssh-agent-1`:

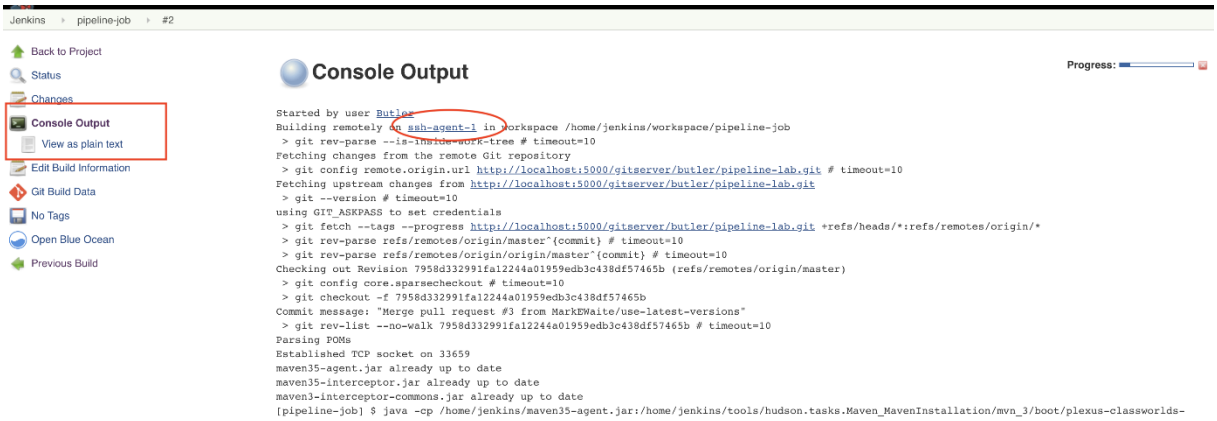


Figure 12. Build job console output

The Node Management system has many admin utilities. To view the node activity for `ssh-agent-1`:

- Go to **Manage Jenkins Manage Node** and click on `ssh-agent-1` node.
- Check the **Load statistics** graph to view the node activity after a few builds.

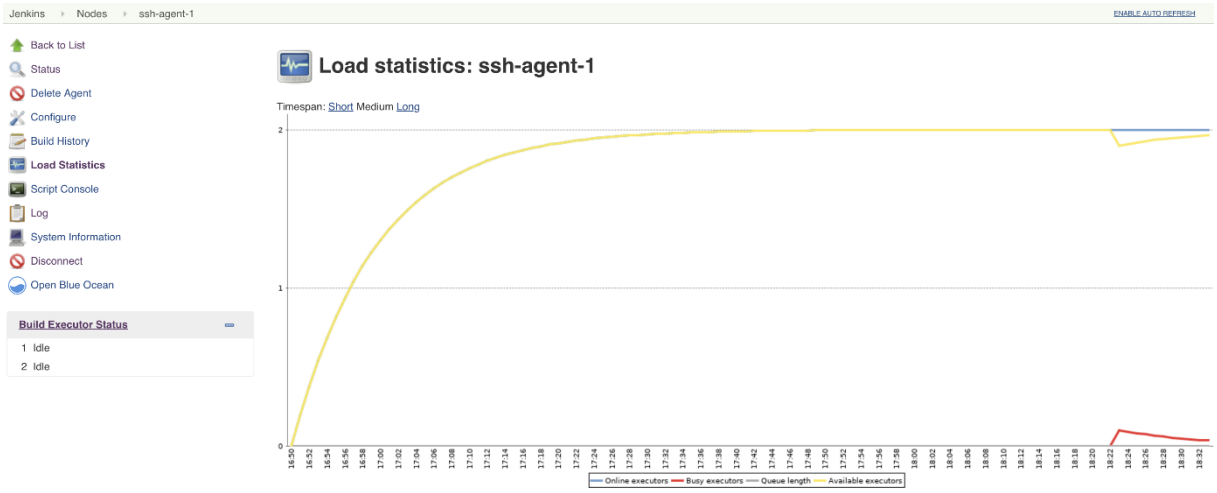


Figure 13. Example of Load Statistics

That's all for this exercise!