Jenkins Tips

- Highlights
- Testing Changes
- Triggering jobs
- Job suffixes
- Job labels
- How to access a test history
- How to delete jobs that are no longer needed
- Jenkins Infrastructure Setup
  - Current installations
  - How to install and upgrade software on Jenkins workers

Highlights

The location ./test-infra/jenkins contains the defined Jenkins jobs are defined in. These jobs are written using the Job DSL using Apache Groovy. Job definitions should be a simple as possible and ideally identify a single Gradle target to execute.

Testing Changes

The following are some Testing Changes tips that could help you:

- Use Local Dockerized Jenkins to test changes locally without breaking the project's job definitions.
- Test a PR that includes Jenkins changes, but it is potentially destructive because the job definitions are a shared resource.
- Use the “Run Seed Job” trigger phrase to update the job definitions on a PR. This will cause the job definitions to be read and parsed, and they will be active until the next scheduled execution on beam_SeedJob or beam_SeedJob_Standalone.

Triggering jobs

Beam committers can trigger a job with the Jenkins UI. Non-committers can trigger a job if there is a trigger phrase.

Job suffixes

Each post-commit and pre-commit job file defines several jobs with different suffixes.

- For pre-commits, there are _Commit, _Phrase, and _Cron suffixes.
- The _Commit job happens with every push to a pull request.
- The _Phrase happens when the trigger phrase is entered as a comment in the pre-commit.
- The _Cron pre-commit happens post-commit on the master branch as a signal of whether the pre-commit would pass without any changes.

Job labels

Consider the following tips regarding Jenkins-jobs-related tags:

- Most Beam Jenkins jobs specify the label beam, which uses beam executors 1-15.
- Performance test jobs specify the label beam-perf, which uses beam executors 16.
- Website publishing job specifies the label git-websites, which allows publishing generated documentation to the asf-site branch.

How to access a test history

To access a test history:

1. Look at this URL:
   https://ci-beam.apache.org/job/beam_PreCommit_Java_Phrase/lastCompletedBuild/testReport/junit/org.apache.beam.runners.fnexecution.data/GrpcDataServiceTest/testMessageReceivedBySingleClientWhenThereAreMultipleClients/history/
2. Go to failed job -> test result -> navigate to failed test -> history

Replace relevant fields in URL. There might be issues getting there for succeeded tests.
How to delete jobs that are no longer needed

To delete jobs that are no longer needed in Jenkins:

1. Remove the `.groovy` file associated with the job. For more information, see

   Removes Python 2 and Python 3.5 Postcommit jobs. #12891.

2. Delete the job history from Jenkins.
   b. Select the job you want to remove.
   c. Click Delete project in the left side panel.

---

### Jenkins Infrastructure Setup

16 GCE instances support Jenkins' continuous integrating jobs. The instance group `apache-ci-beam-jenkins-nodes` manages the compute instances in the Google Cloud project `apache-beam-testing`. Each instance has 16 CPUs and 104GB of memory. We used to run Jenkins nodes using Apache Jenkins Build cluster (https://builds.apache.org/), but we have since migrated to our dedicated cluster: https://ci-beam.apache.org/, and there is an old group of workers `apache-beam-jenkins-jnlp` that is now stopped but not yet deleted.

<table>
<thead>
<tr>
<th>Instance Name</th>
<th>Jenkins agent name</th>
<th>Jenkins Label</th>
</tr>
</thead>
<tbody>
<tr>
<td>apache-ci-beam-jenkins-1</td>
<td>apache-beam-jenkins-1</td>
<td>beam</td>
</tr>
<tr>
<td>apache-ci-beam-jenkins-2</td>
<td>apache-beam-jenkins-2</td>
<td>beam</td>
</tr>
<tr>
<td>apache-ci-beam-jenkins-3</td>
<td>apache-beam-jenkins-3</td>
<td>beam</td>
</tr>
<tr>
<td>apache-ci-beam-jenkins-4</td>
<td>apache-beam-jenkins-4</td>
<td>beam</td>
</tr>
<tr>
<td>apache-ci-beam-jenkins-5</td>
<td>apache-beam-jenkins-5</td>
<td>beam</td>
</tr>
<tr>
<td>apache-ci-beam-jenkins-6</td>
<td>apache-beam-jenkins-6</td>
<td>beam</td>
</tr>
<tr>
<td>apache-ci-beam-jenkins-7</td>
<td>apache-beam-jenkins-7</td>
<td>beam</td>
</tr>
<tr>
<td>apache-ci-beam-jenkins-8</td>
<td>apache-beam-jenkins-8</td>
<td>beam</td>
</tr>
<tr>
<td>apache-ci-beam-jenkins-9</td>
<td>apache-beam-jenkins-9</td>
<td>beam</td>
</tr>
<tr>
<td>apache-ci-beam-jenkins-10</td>
<td>apache-beam-jenkins-10</td>
<td>beam</td>
</tr>
<tr>
<td>apache-ci-beam-jenkins-11</td>
<td>apache-beam-jenkins-11</td>
<td>beam</td>
</tr>
<tr>
<td>apache-ci-beam-jenkins-12</td>
<td>apache-beam-jenkins-12</td>
<td>beam</td>
</tr>
<tr>
<td>apache-ci-beam-jenkins-13</td>
<td>apache-beam-jenkins-13</td>
<td>beam</td>
</tr>
<tr>
<td>apache-ci-beam-jenkins-14</td>
<td>apache-beam-jenkins-14</td>
<td>beam</td>
</tr>
<tr>
<td>apache-ci-beam-jenkins-15</td>
<td>apache-beam-jenkins-15</td>
<td>beam</td>
</tr>
<tr>
<td>apache-ci-beam-jenkins-16</td>
<td>apache-beam-jenkins-16</td>
<td>beam-perf</td>
</tr>
</tbody>
</table>

**Current installations**

- apache-maven:3.5.4
- Docker:18.09.4
- Go:1.16
- kubectl
- open-jdk-11
- open-jdk-1.8
- Pip:8.1.1
- Python:2.7.12
- Python:3.5.2
- Python:3.6.13
- Python:3.7.10
- Python:3.8.9
- Python:3.9.4
- Virtualenv:16.4.3

---

How to install and upgrade software on Jenkins workers
All software updates must be performed and verified in a temporary experimental compute instance. To install or upgrade tools for Jenkins instances:

1. Create a compute instance with the most recently created boot image.

   ```
gcloud compute instances create $USER-jenkins-upgrade --project=apache-beam-testing --zone us-central1-b --image-family=jenkins-worker-boot-image --machine-type=n1-highmem-16
   ```

   ! You can create the instance through VM Instances in Cloud Console or by running the `gcloud` command.

2. Connect using ssh to the instance via Cloud Console or `gcloud` command.

   ```
gcloud compute --project "apache-beam-testing" ssh --zone "us-central1-b" $USER-jenkins-upgrade
   ```

3. Login as `jenkins` user.

   ```
sudo su jenkins
   ```

4. Install or upgrade the required tools.
5. Verify the tool by running corresponding Beam tests.
6. Verify backwards compatibility by running Beam PostCommits against all SDKs.

   - Clone Beam repository from Git:

     ```
git clone https://github.com/apache/beam.git & cd beam
     ```

   - Run your tests by Gradle wrapper. For example:

     ```
./gradlew :PythonPostCommit --continue --<args>
     ```

   - Remove beam repository after verification.

7. Create a new image from the your instance and name it as `jenkins-worker-boot-image-YYYYMMDD`. Put the image in the family of `jenkins-worker-boot-image`, and add a description of your changes.

   ```
gcloud compute instances stop $USER-jenkins-upgrade --project "apache-beam-testing" --zone "us-central1-b"
gcloud compute images create \
   --project apache-beam-testing \
   --source-disk $USER-jenkins-upgrade \
   --source-disk-zone us-central1-b \
   jenkins-worker-boot-image-<YYYYMMDD> \
   --family jenkins-worker-boot-image \
   --description "Updated XYZ, see BEAM-1234."
   ```

To reboot Jenkins executors with a new image, do the following:

1. Go to the page for the Jenkins agent you want to update (for example) and click "Mark this node temporarily offline", leaving a reason such as "Updating X dependency."
2. Wait until there are no more tests running in that agent (under "Build Executor Status" on the left of the page). You can also prioritize the update on agents currently not running any builds.
3. Restart the VM with a new disk:
# Select the workers currently offline and not processing any builds, # that you want to upgrade. # For example: for CI_INSTANCE_NUMBER {1..4} {10..12} 16; do

for CI_INSTANCE_NUMBER in {1..4}; do
    CI_INSTANCE=apache-ci-beam-jenkins-${CI_INSTANCE_NUMBER}
gcloud compute instances stop $CI_INSTANCE --project $PROJECT --zone $ZONE
    gcloud compute instances detach-disk $CI_INSTANCE --project $PROJECT --zone $ZONE --disk $(CI_INSTANCE)-${OLD_IMAGE_DATE}
gcloud compute disks create $(CI_INSTANCE)-${NEW_IMAGE_DATE} --project $PROJECT --zone $ZONE --image=jenkins-worker-boot-image-${NEW_IMAGE_DATE}
gcloud compute instances attach-disk $CI_INSTANCE --project $PROJECT --zone $ZONE --disk $(CI_INSTANCE)-${NEW_IMAGE_DATE} --device-name=$(CI_INSTANCE)-${NEW_IMAGE_DATE} --boot
    gcloud compute instances start $(CI_INSTANCE) --project $PROJECT --zone $ZONE
done

You could also do these steps via Cloud Console instead of CLI:
   a. Stop the VM, then click edit.
   b. Delete the existing boot disk by clicking the 'X'.
   c. Create a new boot disk by using the newly generated image.
   d. Name disk as 'apache-ci-beam-jenkins-[1,16]-YYYYMMDD'.
   e. Make sure the boot disk 'Model' is 'Boot, read/write'.
   f. Start the VM.

4. Mark the node in Jenkins as online again and click "Launch agent" button if there is one.

5. Verify that inventory job for this node did not start while the worker was offline, sample build for node #10: https://ci-beam.apache.org/job/beam_Inventory_apache-beam-jenkins-10. If there is a stuck execution of the inventory job, cancel it manually. TODO (BEAM-13666): we should cancel inventory stuck jobs automatically.

6. (Optional) If this is the first agent being updated and you would like to test your changes by running a particular Jenkins job, do the following:
   a. Go to the page for the job you want to run (example).
   b. Click "Configure" on the left menu.
   c. Find the checkbox "Restrict where this project can be run" and change the restriction from "beam" to the specific name of the agent (ex. "apache-beam-jenkins-1").
   d. Save and apply that change.
   e. Back on the page for the job, click "Build with Parameters" on the left menu.
   f. Run the build on "master".
   g. Once you're done checking the results, change the restriction for the job back to "beam".

7. Repeat for each worker. You can also do it in batches of workers, but avoid bringing all of the workers offline at once.

8. After confirming that update was successful, we can go to Cloud Console Disks and clean up deprecated Jenkins worker disks, or do this via cli:

    for CI_INSTANCE_NUMBER in {1..16}; do
        CI_INSTANCE=apache-ci-beam-jenkins-${CI_INSTANCE_NUMBER}
gcloud compute disks delete $(CI_INSTANCE)-${OLD_IMAGE_DATE} --project $PROJECT --zone $ZONE --quiet
done