How To Jenkins Sandbox

Creating a Local Jenkins Sandbox

The following describes how to set up a local instance of Jenkins on your laptop using Docker or Vagrant. This can be used for development and testing of Jenkins Jobs.

A Jenkins instance can easily be spun up as a container or virtual machine using Docker of Vagrant on your local machine. A Jenkins Docker container can be found on Docker Hub and documentation for using it can be found here.

```
$ docker pull jenkins/jenkins:lts

$ docker run -d -v jenkins_home:/var/jenkins_home -p 8080:8080 -p 50000:50000 jenkins/jenkins:lts
```

Tip: you can disable the first-run setup wizard with the following environment variable:

```
JAVA_OPTS=-Djenkins.install.runSetupWizard=false
```

You can also set the initial admin password to an arbitrary value by setting the following environment variable:

```
JENKINS_OPTS=--argumentsRealm.roles.user=admin --argumentsRealm.passwd.admin=insecurepassword --
argumentsRealm.roles.admin=admin
```

Various Vagrant boxes can be found readymade on the Vagrant public catalog based on various base images such as Ubuntu and Centos with Jenkins installed ontop. Or you can create one using a Vagrantfile like below. For instruction on installing Jenkins and Jenkins Job Builder see links.

```
Vagrant.configure("2") do |config|

  config.disksize.size = '80GB'

  config.vm.box = "centos/7" # Base image of choice

  config.vm.network "private_network", ip: "192.168.x.x"

  # Set an ip so that you can access Jenkins web gui e.g. http://192.168.50.4:8080/


  config.vm.provider "virtualbox" do |vb|

    vb.memory = "4048"

  end


  config.vm.provision "shell", inline: <<-SHELL

    # INSTALL JENKINS & JJB etc.

    # ...

  SHELL

end
```

After creating your Vagrantfile use the following commands to spinup the vm and login.

```
$ vagrant up

$ vagrant ssh
```


Ensure that your newly-minted Jenkins has both the Environment Injector and MultiJob plugins installed by navigating to the Plugin Manager (accessible under `http://$JENKINS_HOST:$JENKINS_PORT/pluginManager/`)

Adding/Testing Jobs In Jenkins Sandbox

The following describes how to add/modify/run Jenkins Jobs defined using Jenkins Job Builder. Purpose is to demonstrate how to quickly test and validate jobs intended for the likes of Nordix in a sandboxed environemtn, so that you can have extra confidence before committing your job(s) for review.

1. Login to an environment which can access the Jenkins Sandbox.

2. Pull down the job(s) you are looking to modify and test e.g. nordix infra/cicd

```
$ git clone "https://gerrit.nordix.org/infra/cicd"
```

3. Update Jenkins with the jobs by running the following command.

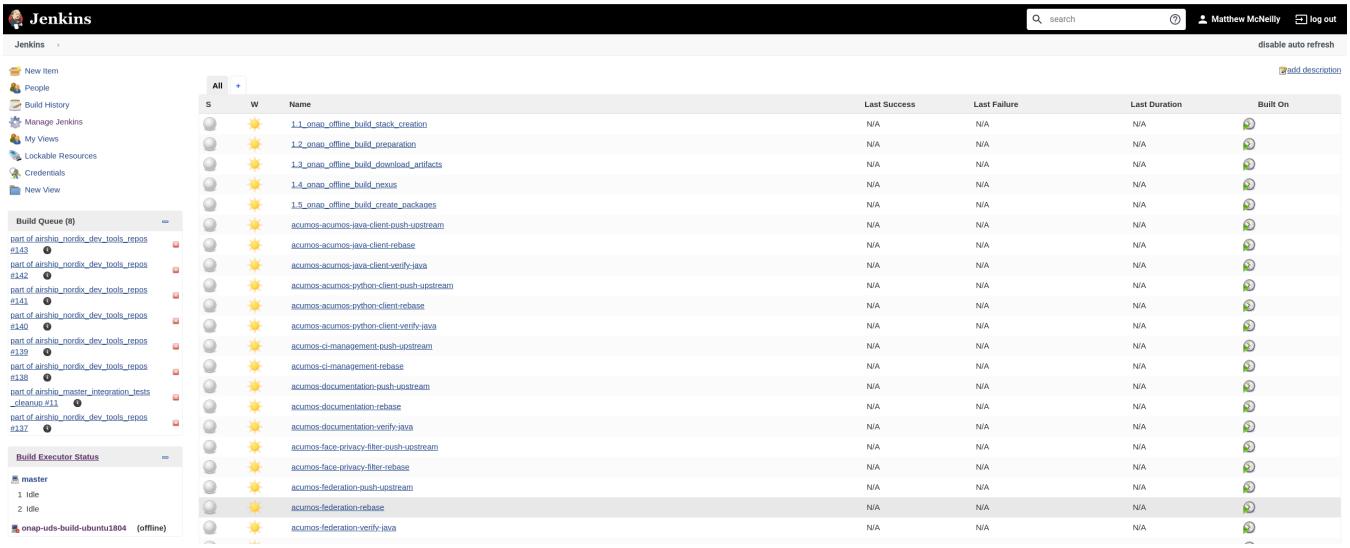
```
$ jenkins-jobs --conf cicd/jjb/global/jenkins_jobs.ini update –-recursive --delete-old ~/cicd/jjb/

If this does not work , then below command line can be used:
jenkins-jobs --conf /<directory_name>/cicd/jjb/global/jenkins_jobs.ini update  -r .
```

Taking note to point to a jjb configuration file, using the –-conf flag. An example of this exists in the infra/cicd repository under cicd/jjb/global/jenkins_jobs.ini . Make a copy of this and modify/add to it as required such as the user, password and url parameters. This can be stored under /etc/jenkins_jobs/jenkins_jobs.ini and will be found automatically without using the –conf flag. The configuration file points to the Jenkins server you are looking to control and specifies parameters for the job builder.

```
[jenkins]

user=<USERNAME>

password=<PASSWORD>

url=<IP/URL OF JENKINS SERVER>     # e.g. http://127.0.0.1:8080

query_plugins_info=True


[job_builder]

ignore_cache=True

keep_descriptions=False

recursive=True

allow_duplicates=False
```

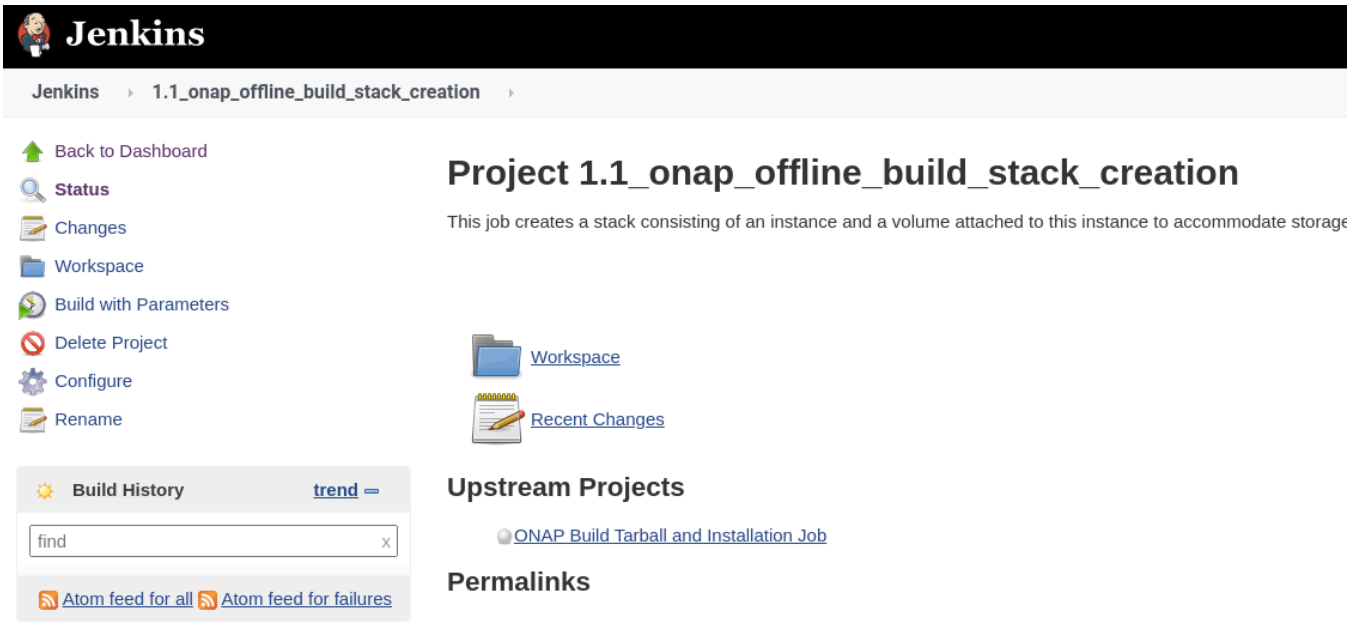
After JJB has updated Jenkins, you should see all the jobs listed.

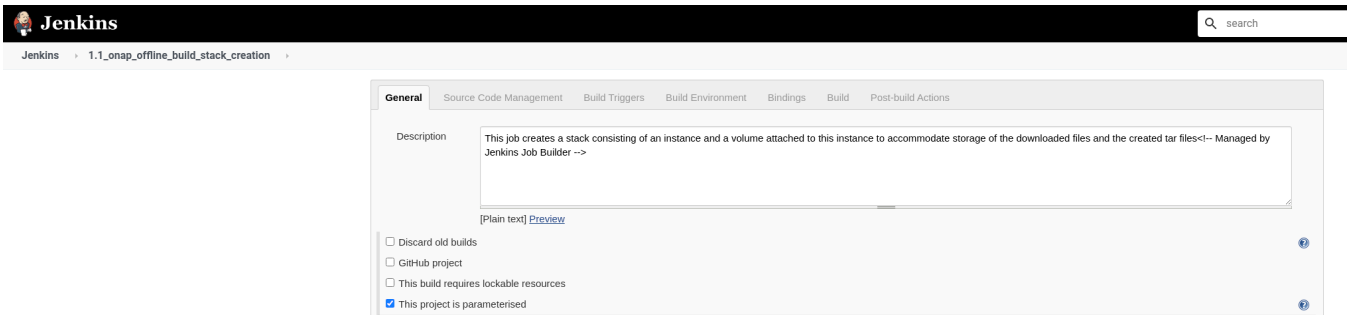
```
INFO:jenkins_jobs.builder:Creating jenkins job oransc-sim-a1-interface-rebase

INFO:jenkins_jobs.builder:Creating jenkins job oransc-sim-e2-interface-push-upstream

INFO:jenkins_jobs.builder:Creating jenkins job oransc-sim-e2-interface-rebase

INFO:jenkins_jobs.builder:Creating jenkins job oransc-sim-o1-interface-push-upstream

INFO:jenkins_jobs.builder:Creating jenkins job oransc-sim-o1-interface-rebase

INFO:jenkins_jobs.builder:Creating jenkins job oransc-sim-push-upstream

INFO:jenkins_jobs.builder:Creating jenkins job oransc-sim-rebase

INFO:jenkins_jobs.builder:Creating jenkins job uds-build

INFO:jenkins_jobs.builder:Creating jenkins job uds-daily-build-package-promote

INFO:jenkins_jobs.builder:Creating jenkins job uds-package-artifacts

INFO:jenkins_jobs.builder:Creating jenkins job uds-package-images

INFO:jenkins_jobs.builder:Creating jenkins job uds-package-repositories

INFO:jenkins_jobs.builder:Creating jenkins job uds-promote-artifacts

INFO:jenkins_jobs.builder:Creating jenkins job uds-promote-images

INFO:jenkins_jobs.builder:Creating jenkins job uds-promote-repositories

INFO:jenkins_jobs.cli.subcommand.update:Number of jobs updated: 973

INFO:jenkins_jobs.builder:Number of views generated:  0

INFO:jenkins_jobs.cli.subcommand.update:Number of views updated: 0

INFO:jenkins_jobs.cli.subcommand.update:Number of jobs deleted: 0
```



4. Access the Jenkins Sandbox web gui and login.

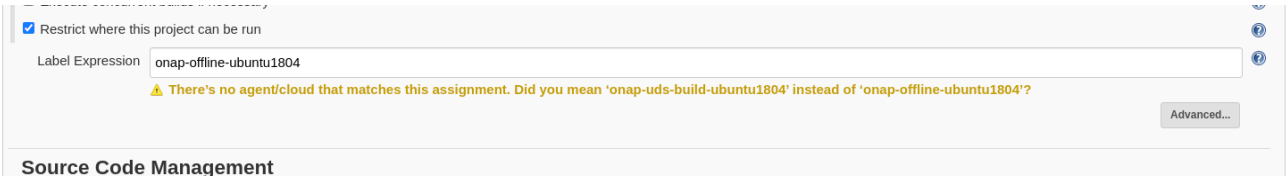
5. Next select the job you are wishing to test. Then select “Configure” to edit the job. You will not see this option unless you are logged in.



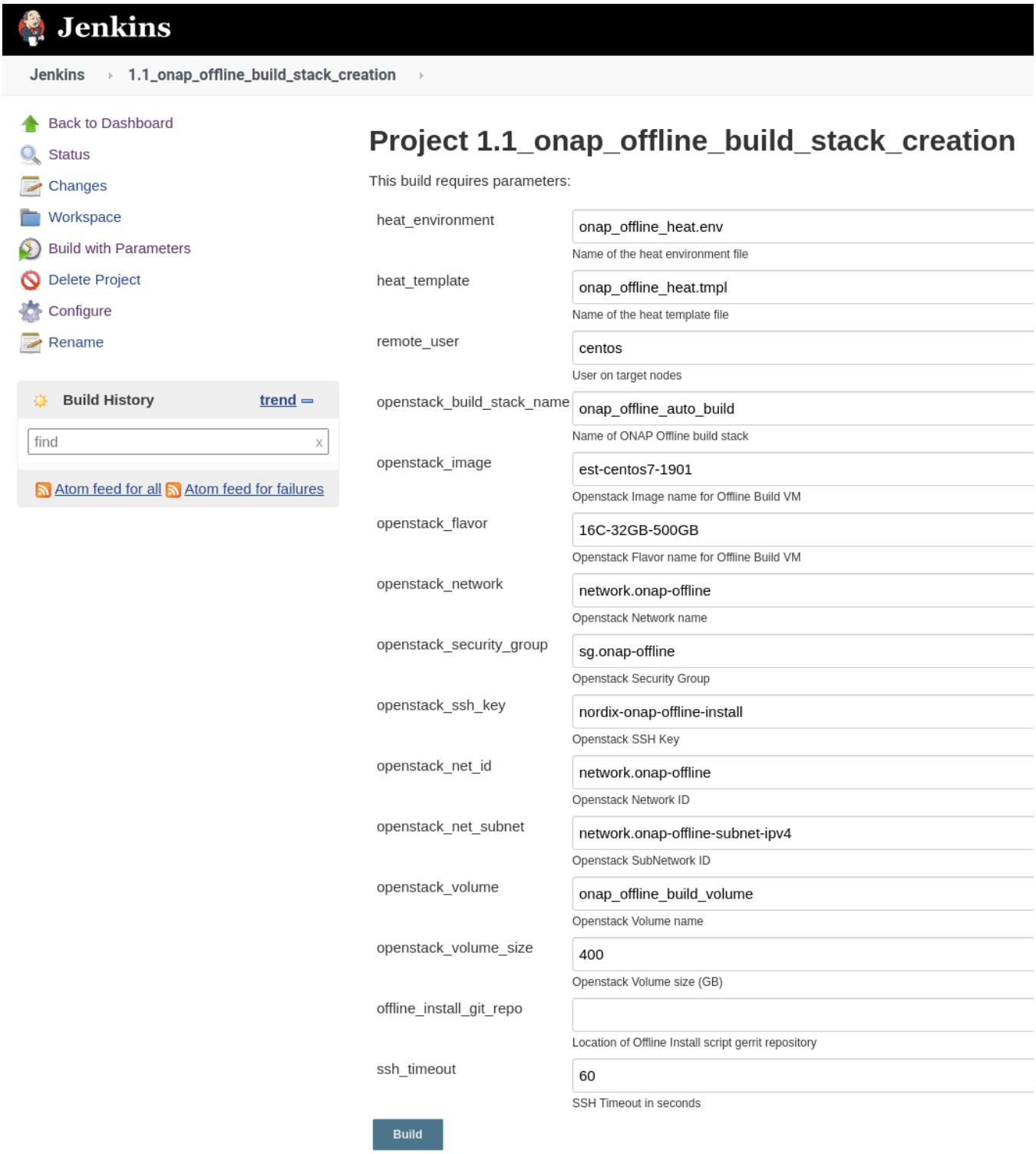


Modifications may be required to get the job to run initially e.g. removing restrictions as to which node(s) the job can run on. Alternatively a docker slave can be created with the same label to fufill this requirement, to do this see “Build Docker Slaves and Testing Job Scripts” chapter. Doing minimal changes manually through Jenkins ensures that the jjb job source code doesn’t change.

6. Click “Save” when finished. Testing the job within the Jenkins sandbox will give you more confidence that the job will run when accepted and merged.



7. Select “Build” or “Build With Parameters” to kick of the job.



8. Once happy with the job, push it for review.

Build Docker Slaves and Testing Job Scripts

The following describes how to quickly spin up docker based slaves for use within Jenkins. Purpose is to further help in developing/testing Jenkins Jobs and their build environments.

1. Firstly create a Dockerfile specifying the environment/software needed to run the required job, examples of Dockerfiles can be found under the infra/tools repository such as infra-tools-docker-slave-ubuntu1804. The following commands can be used to create a Docker image and push it to Docker hub. Alternatively, you can use an existing Docker image such as those hosted at nordixorg on Docker Hub.

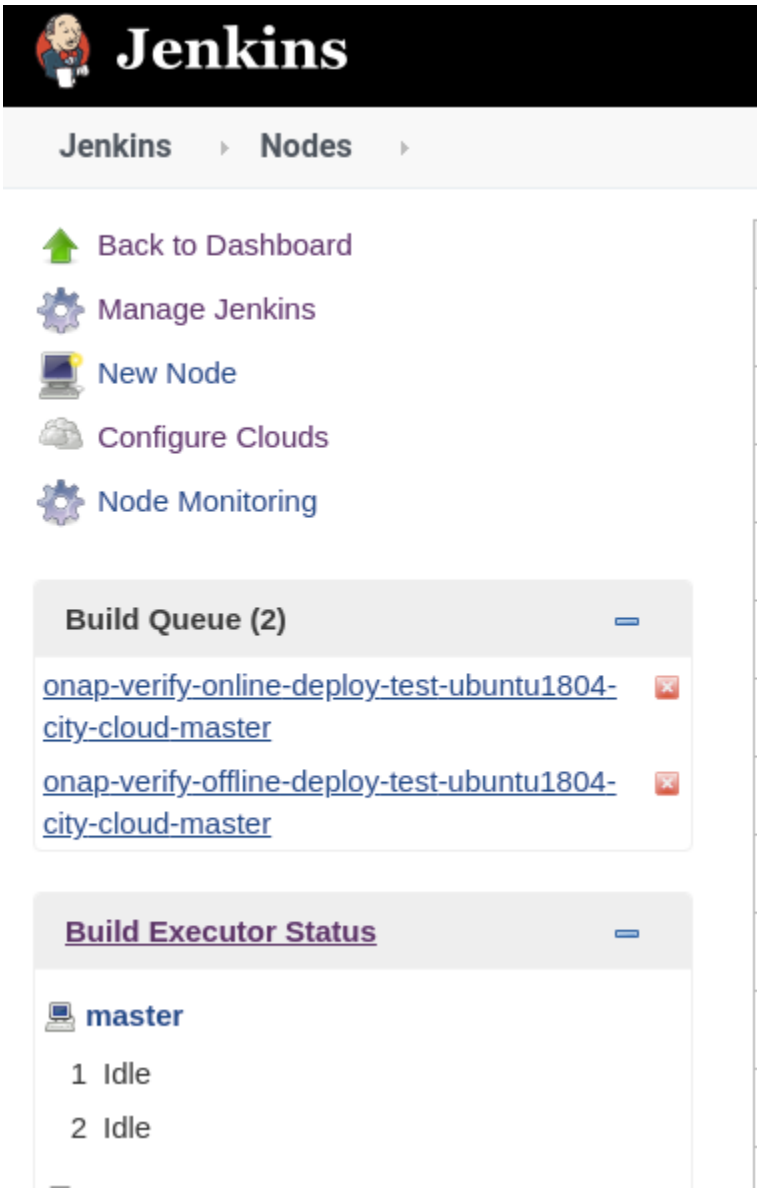
```
$ docker build <directory>

$ docker commit –m “<COMMIT MESSAGE>” -a “<DOCKERHUB_USERNAME>” <IMAGE> <DOCKERHUB_USERNAME>/<IMAGE>:latest

$ docker login

$ docker push <DOCKERHUB_USERNAME>/<IMAGE>
```

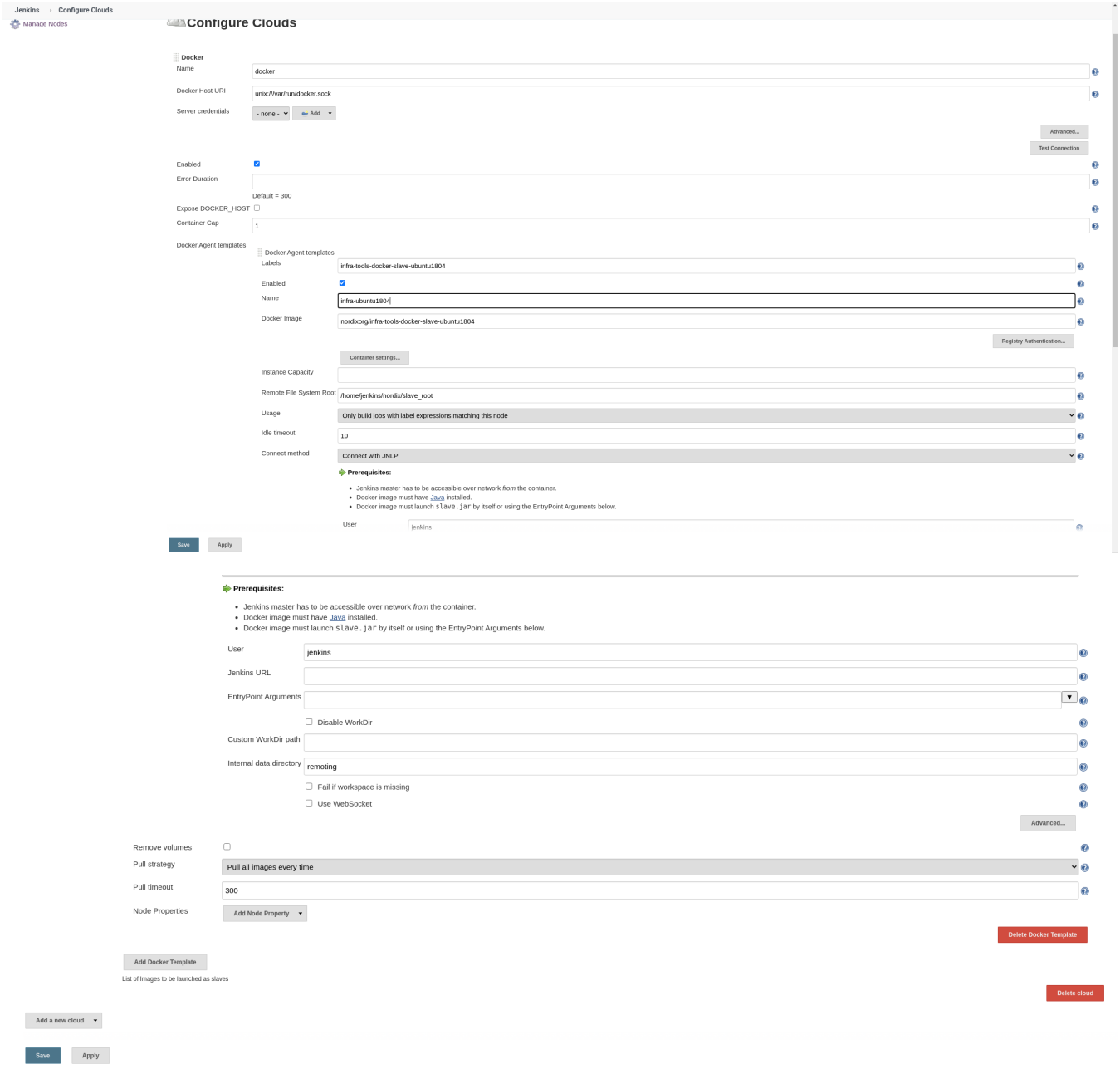
These docker images can be used to then spin up docker based Jenkins slaves within the Jenkins environment using the Jenkins docker-plugin.

2. To add further docker slaves to the Jenkins environment go to “Build Executor Status” > “Configure Cloud”. Then configure a new cloud template specifying the docker image your wishing to use from Docker Hub.

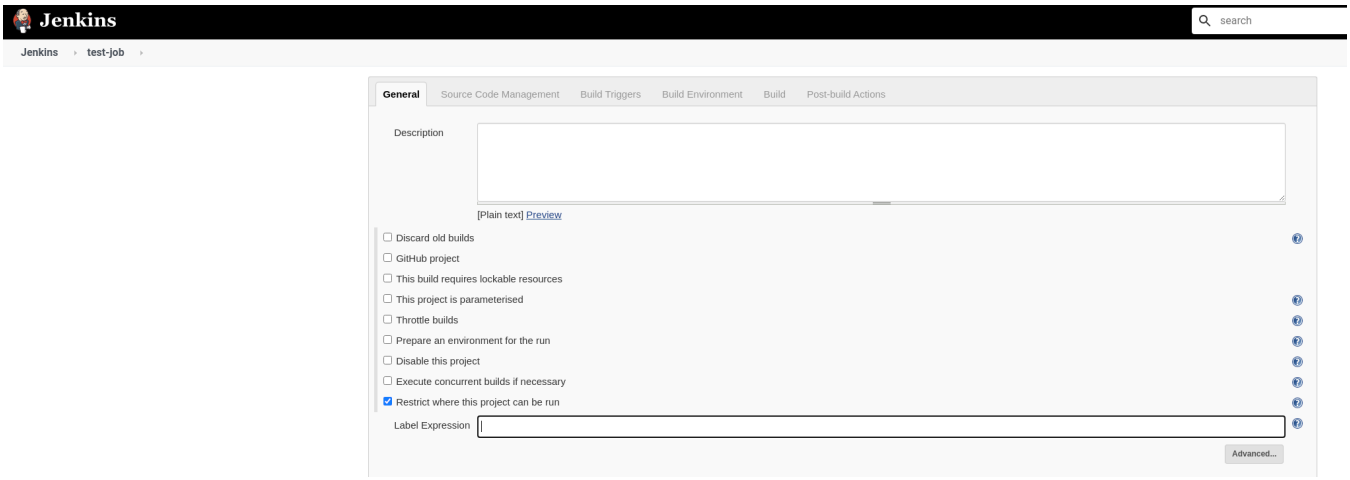
Below is an example of a configured cloud template on a locally hosted Jenkins vm. Some properties you may wish to change include;

- **Docker Host URI** – Machine used to spin up the docker slaves. “unix:///var/run/docker.sock” can be used to specify the same machine i.e. Jenkins master
- **Container Cap** - Maximum number of docker slaves to run
- **Labels** - Label given to the slaves.
- **Name** – Name given to slaves, determines how they appear
- **Docker Image –** Docker image to use to create slaves from

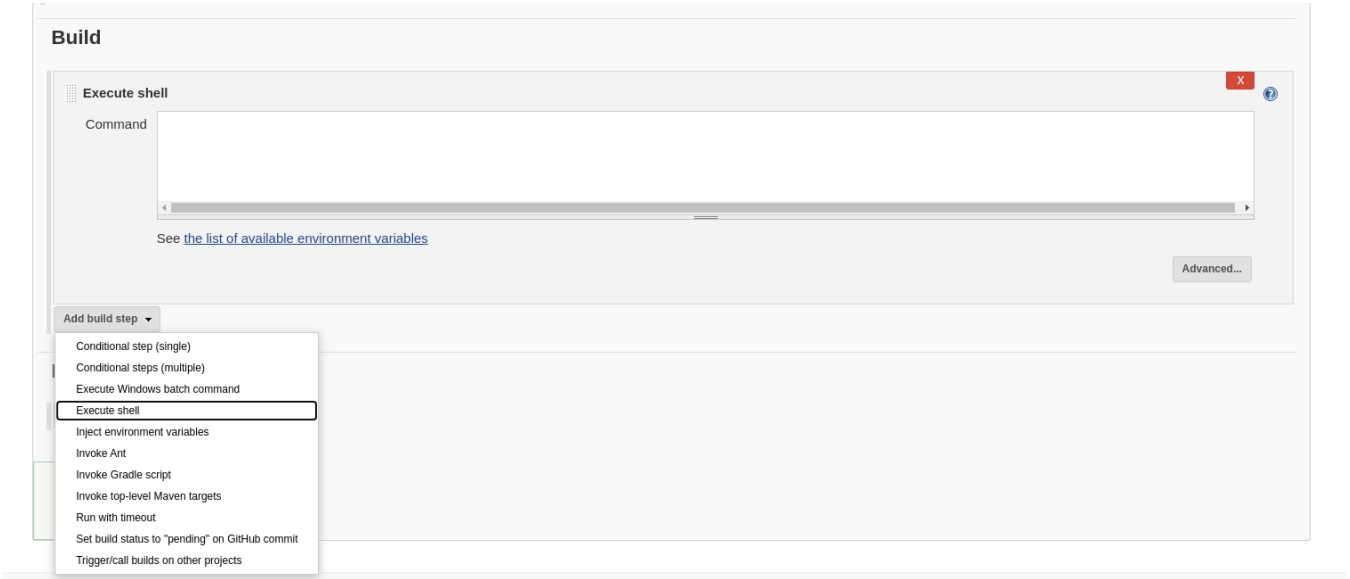

Jenkins
Jenkins
Nodes
Back to Dashboard
Manage Jenkins
New Node
Configure Clouds
Node Monitoring
Build Queue (2)
onap-verify-online-deploy-test-ubuntu1804-city-cloud-master
onap-verify-offline-deploy-test-ubuntu1804-city-cloud-master
Build Executor Status
master
1 Idle
2 Idle



3. Create a new job by selecting “New Item” > “Freestyle project”, give the job a name then press “OK”. Assign the job to the new docker slave by selecting “Restrict where this project can be run” and entering the name of the docker slave. This will be the same as the label you assigned during creating the cloud template.



4. Test your scripts by adding a build step, by selecting “Add Build Step” > “Execute Shell”. Also add in other configuration if needed e.g. parameters



5. When finished select “Save”. Then “Build” or “Build with Parameters” to kick of the job.