

# 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 or 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/](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 environment, 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-al-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-ol-interface-push-upstream
INFO:jenkins_jobs.builder:Creating jenkins job oransc-sim-ol-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

```

The screenshot shows the Jenkins dashboard with the following sections:

- Left Sidebar:**
  - New Item
  - People
  - Build History
  - Manage Jenkins
  - My Views
  - Lockable Resources
  - Credentials
  - New View
- Build Queue (8):**

#	Name	Last Success	Last Failure	Last Duration	Built On
#143	part_of_airship_nordix_dev_tools_repos	N/A	N/A	N/A	
#142	part_of_airship_nordix_dev_tools_repos	N/A	N/A	N/A	
#141	part_of_airship_nordix_dev_tools_repos	N/A	N/A	N/A	
#140	part_of_airship_nordix_dev_tools_repos	N/A	N/A	N/A	
#139	part_of_airship_nordix_dev_tools_repos	N/A	N/A	N/A	
#138	part_of_airship_nordix_dev_tools_repos	N/A	N/A	N/A	
#137	part_of_airship_nordix_integration_tests	N/A	N/A	N/A	
#132	part_of_airship_nordix_dev_tools_repos	N/A	N/A	N/A	
- Build Executor Status:**

Status	Label	Count
master	1 idle	1
onap-uds-build-ubuntu1804	(offline)	1

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.

The screenshot shows the Jenkins project page for '1.1\_onap\_offline\_build\_stack\_creation'. On the left, there's a sidebar with links: Back to Dashboard, Status, Changes, Workspace, Build with Parameters, Delete Project, Configure, and Rename. Below that is a search bar with 'Build History' and 'trend' dropdown, and a 'find' input field. At the bottom of the sidebar are Atom feed links. The main content area has a title 'Project 1.1\_onap\_offline\_build\_stack\_creation' and a description: 'This job creates a stack consisting of an instance and a volume attached to this instance to accommodate storage'. It also includes 'Workspace' and 'Recent Changes' sections. A 'Upstream Projects' section lists 'ONAP Build Tarball and Installation Job'. A 'Permalinks' section is also present.

The screenshot shows the 'General' configuration tab for the Jenkins project. The 'Description' field contains the text: 'This job creates a stack consisting of an instance and a volume attached to this instance to accommodate storage of the downloaded files and the created tar files<br>Managed by Jenkins Job Builder'. Below the description are several checkboxes: 'Discard old builds', 'GitHub project', 'This build requires lockable resources', and 'This project is parameterised'. The 'This project is parameterised' checkbox is checked. There are also 'Plain text' and 'Preview' buttons. The top right corner has a search bar.

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 fulfill 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.

The screenshot shows the 'Source Code Management' configuration tab. It has a 'Label Expression' field containing 'onap-offline-ubuntu1804'. A warning message below it says: '⚠ There's no agent/cloud that matches this assignment. Did you mean 'onap-uds-build-ubuntu1804' instead of 'onap-offline-ubuntu1804'?' There is an 'Advanced...' button. The top right corner has a search bar.

7. Select "Build" or "Build With Parameters" to kick off the job.

[Back to Dashboard](#)[Status](#)[Changes](#)[Workspace](#)[Build with Parameters](#)[Delete Project](#)[Configure](#)[Rename](#)**Build History**[trend](#)  [Atom feed for all](#) [Atom feed for failures](#)

## Project 1.1\_onap\_offline\_build\_stack\_creation

This build requires parameters:

heat_environment	onap_offline_heat.env
	Name of the heat environment file
heat_template	onap_offline_heat tmpl
	Name of the heat template file
remote_user	centos
	User on target nodes
openstack_build_stack_name	onap_offline_auto_build
	Name of ONAP Offline build stack
openstack_image	est-centos7-1901
	Openstack Image name for Offline Build VM
openstack_flavor	16C-32GB-500GB
	Openstack Flavor name for Offline Build VM
openstack_network	network.onap-offline
	Openstack Network name
openstack_security_group	sg.onap-offline
	Openstack Security Group
openstack_ssh_key	nordix-onap-offline-install
	Openstack SSH Key
openstack_net_id	network.onap-offline
	Openstack Network ID
openstack_net_subnet	network.onap-offline-subnet-ipv4
	Openstack SubNetwork ID
openstack_volume	onap_offline_build_volume
	Openstack Volume name
openstack_volume_size	400
	Openstack Volume size (GB)
offline_install_git_repo	
	Location of Offline Install script gerrit repository
ssh_timeout	60
	SSH Timeout in seconds

[Build](#)

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

Jenkins > Configure Clouds

## Configure Clouds

**Docker**

Name: docker

Docker Host URI: unix://var/run/docker.sock

Server credentials: - none -

Enabled:

Error Duration: Default = 300

Expose DOCKER\_HOST:

Container Cap: 1

Docker Agent templates:

- Labels: infra-tools-docker-slave-ubuntu1804
- Enabled:
- Name: infra-ubuntu1804
- Docker Image: nordxorg/infra-tools-docker-slave-ubuntu1804

Container settings...

Instance Capacity:

Remote File System Root: /home/jenkins/nordx/slave\_root

Usage: Only build jobs with label expressions matching this node

Idle timeout: 10

Connect method: Connect with JNLP

**Prerequisites:**

- Jenkins master has to be accessible over network from the container.
- Docker image must have `java` installed.
- Docker image must launch `$java -jar` by itself or using the EntryPoint Arguments below.

User: jenkins

---

**EntryPoint Arguments:**

User: jenkins

Jenkins URL:

Custom WorkDir path:

Internal data directory: remoting

Disable WorkDir

Fall if workspace is missing

Use WebSocket

Remove volumes:

Pull strategy: Pull all images every time

Pull timeout: 300

Node Properties:

Add Docker Template

List of Images to be launched as slaves

Add a new cloud

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.

The screenshot shows the Jenkins 'General' configuration page for a job named 'test-job'. The 'Restrict where this project can be run' checkbox is checked. Other checkboxes shown include 'Discard old builds', 'GitHub project', 'This build requires lockable resources', 'This project is parameterized', 'Throttle builds', 'Prepare an environment for the run', 'Disable this project', and 'Execute concurrent builds if necessary'. A 'Label Expression' field is also present.

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

The screenshot shows the Jenkins 'Build' configuration page. An 'Execute shell' build step is selected. The 'Command' field is empty. A tooltip indicates 'See the list of available environment variables'. A dropdown menu for 'Add build step' is open, showing options like 'Conditional step (single)', 'Conditional steps (multiple)', 'Execute Windows batch command', 'Execute shell' (which is highlighted), 'Inject environment variables', 'Invoke Ant', 'Invoke Gradle script', 'Invoke top-level Maven targets', 'Run with timeout', 'Set build status to "pending" on GitHub commit', and 'Trigger/call builds on other projects'.

5. When finished select "Save". Then "Build" or "Build with Parameters" to kick off the job.