Abstract:

We want to set up Jenkins in a setting that is highly available and reliable. In order to do that we

need to set up 4 + 1 machines as detailed below.

1. Lauch 4 instances in AWS:

Ubuntu

T2. Micro

Security group: default

10 GB

Number of instances: 4

1. Create a helper file in terminal which includes public IPs and private Ips of each machine.
2. Lauch 1 instance in AWS with the above specs, but add another storage to it. Each storage will be 10GB. This will be the storage instance. Name it.
3. Connect to swarm1

ssh -i "C:\Users\User\Desktop\AWS\_key\RoyG\_key.pem" ubuntu@ENTER IP HERE

sudo -i

edit host file so that each machine will recognize other machines by name:

cat /etc/hosts, as follows:  
  
sudo hostnamectl set-hostname swarm1

sudo apt -y update

sudo apt -y install docker.io

cat<<EOF>>/etc/hosts

172.31.27.207 swarm1 #private IP

172.31.19.12 swarm2 #private IP

172.31.22.236 swarm3 #private IP

172.31.21.235 storage #private IP

172.31.22.69 agent #private IP

EOF

1. From swarm1:

sudo docker swarm init

copy to helper file:

docker swarm join --token SWMTKN-1-50bl5dz4ukfq9tq5thx5ttz8rbiotnf8ph67dnimtux3eiu9ww-514dbprs4qjz4i6qglcl87jun 172.31.27.207:2377

change IP address to swarm1:

docker swarm join --token SWMTKN-1-50bl5dz4ukfq9tq5thx5ttz8rbiotnf8ph67dnimtux3eiu9ww-514dbprs4qjz4i6qglcl87jun swarm1:2377

1. Connect to swarm2 and paste the script from 4. Remember to change “swarm1” here: “sudo hostnamectl set-hostname swarm1” to swarm2. The following output should be received:

This node joined a swarm as a manager.

1. Perform section for swarm3 as well.
2. Connect to storage.
3. Hostnamectl set-hostname storage
4. apt -y update &>/dev/null 🡪 don’t show me the log. Not mandatory.
5. sudo apt update
6. sudo apt install nfs-kernel-server
7. sudo apt install nfs-common
8. sudo systemctl start nfs-kernel-server
9. sudo systemctl enable nfs-kernel-server
10. verify: sudo systemctl status nfs-kernel-server
11. We want to create a storage disk on the storage instance and share it. We don’t want to use the default disk because it contains an OS.
12. cat /etc/fstab. All available disks and partitions
13. blkid. All block devices.
14. We want to create a new partition and give it all the space in the disk.
15. fdisk /dev/xvdb. Click n (new)
16. Partition number 1. Primary. First sector 1. Last sector: we can give default 8191. Now we have a partition but we don’t have a file system yet. So the OS can’t work with it yet.
17. We want to put a file system in there. So Make FileSystem: run mkfs.ext4 /dev/xvdb1
18. Blkid should display all disks and partitions. Now we can see that xvdb1 had a file system, a block size etc. So it’s ready to go.
19. mkdir /mnt/export
20. Now we need to edit ftstab: vi /etc/fstab. Unlike a mount command, editing this file is permanent. That’s what we want.
21. Add the 4th line to the file:

**LABEL=cloudimg-rootfs / ext4 discard,commit=30,errors=remount-ro 0 1**

**LABEL=BOOT /boot ext4 defaults 0 2**

**LABEL=UEFI /boot/efi vfat umask=0077 0 1**

**/dev/xvdb1 /mnt/export ext4 defaults 0 0**

To save: Esc -> :wq

1. systemctl daemon-reload, and mount -a. meaning: this will read and apply all the changes in ftstab.
2. Final check that we can write to this location. We’ll create a test file. touch /mnt/export/test.
3. Ls /mnt/export/ and we should see the test file.
4. Now we want to start sharing this folder. So vi /etc/export. Make sure you’re sudo.
5. Add this line to share and give permissions:

/mnt/export (one tab here) swarm1(rw,sync,no\_subtree\_check)

1. systemctl restart nfs-server
2. Connect to swarm1
3. Apt -y install nfs-common/
4. vi /etc/ftstab
5. mount -a
6. in the storage machine: chmod -R 777 /mnt/export/
7. in swarm1:

to check if permission is ok: touch /mnt/testswarm1

mkdir /mnt/jenkins\_home.

vi /etc/fstab and add:

storage:/mnt/export /mnt/jenkins\_home nfs defaults 0 0

systemctl daemon-reload

mount -a

df -h to check

reboot and log in again (new ip)

to check: touch /mnt/jenkins\_home/swarm1

1. in swarm2:

sudo -i

make sure ips of all machines are correct in /etc/hosts

apt -y install nfs-common

mkdir /mnt/jenkins\_home

vi /etc/fstab and add:

storage:/mnt/export /mnt/jenkins\_home nfs defaults 0 0

systemctl daemon-reload

in storage: vi /etc/exports add swarm2 and swarm3 after swarm1

systemctl restart nfs-server

back to swarm2:

mount -a

check:

touch /mnt/jenkins\_home/swarm2

also possible to run in the storage machine: ls /mnt/export

1. Do the same for swarm3
2. Before we start building with jenkins, we need to give user 1000 owenership of jenkins\_home

So in the storage machine:

chown -R 1000:1000 /mnt/export/jenkins\_home/

1. We need to create a network, so in swarm3: docker network create --driver overlay jenkins (jenkins is the name we chose).

We can see it was created: docker network ls

1. docker service create --name jenkins --publish published=8080,target=8080 --mount type=bind,source=/mnt/jenkins\_home,target=/var/jenkins\_home jenkins/jenkins
2. We can check everything is ok: docker service inspect. Look specifically at “mounts”.
3. Check that we have jenkins ls /mnt/jenkins\_home/
4. Now we should be able to connect to jenkins via public ip of any swarm machine.
5. Adding external and internal load balancers and target groups:
6. External:

Go to AWS load balancers

Create new, application

First we’ll create internet facing and give an appropriate name (with “external”).

Availability zone: should be like machines

http port 8080

create target group

instances

name: like load balancer

check the three swarm machines

include as pending

create target group

(Remember to delete load balancer, target groups, instances after finishing)

now refresh the load balancer target group box and choose the newly created target group

create load balancer

copy DNS name to helper file and add :8080

after a few minutes- paste in browser to see if it gets to jenkins

1. Internal (for connecting the agent):

Create new, application

Internal

In listeners: change port to 50,000 (jenkins agent port)

Create target group, instances, 50,000

Choose the swarm machines etc

1. In storage machine:

cat /mnt/export/secrets/initialAdminPassword

copy the password to “unlock jenkins” screen.

Install suggested plugins

Admin for everything. Email [admin@admin.com](mailto:admin@admin.com)

Manage jenkins -> nodes

Nodes are all the jenkins agents. Built-in node is jenkins itself.

New node. We can call it docker. Permanent agent.

Enter description such as:

This agent is for docker related operations e.g docker build, docker run etc.  
Remote root directory – very important: /var/jenkins\_home

Labels: docker

Usage: in our case we want the agent to run only specific jobs, not all jobs. So choose second option.

Launch method: we can control either SSH or JAR file. In our case Launch agent by connecting to the controller. SSH.

Availability: keep online as much as possible.

Save

Marked with “X” because it’s unavailable.

Click the name.

1. These are actually “instructions” on how to configure our linux to Jenkin’s agent.

First line in Unix (any linux actually) is where he should take the JAR file from. It’s in our load balancer (LB)

curl -sO http://Lesson-23-external-R-1928713263.us-west-2.elb.amazonaws.com:8080/jnlpJars/agent.jar

java -jar agent.jar -url http://lesson23-external-roy-1769587461.us-west-2.elb.amazonaws.com:8080/ -secret

ba8ea6ab1e27260226741c0303c5b5ad1f01e42d236ddbc400686c9b9d334c60 -name docker -webSocket -workDir "/var/jenkins\_home"

1. Connet to agent machine
2. In agent:

Sudo -i

Hostnamectl set-hostname agent

apt -y update && apt -y install docker.io

Take the curl command from jenkins and run:

curl -sO http://Lesson-23-external-R-1928713263.us-west-2.elb.amazonaws.com:8080/jnlpJars/agent.jar

ls. You should see the JAR file now.

mkdir /var/jenkins\_home

mv agent.jar /var/jenkins\_home

chmod -R 777 /var/jenkins\_home

now connect using the second line from jenkins box:

java -jar agent.jar -url http://Lesson-23-external-R-1928713263.us-west-2.elb.amazonaws.com:8080/ -secret ba8ea6ab1e27260226741c0303c5b5ad1f01e42d236ddbc400686c9b9d334c60 -name docker -webSocket -workDir "/var/jenkins\_home"

but he says JAVA not installed and gives me options to install. We need to choose openjdk (open source), not java (because oracle wants payment for it).

apt install openjdk-17-jre-headless

java --version

It should say INFO: Connected

Now go back to jenkins, and there’s no more X on the docker. And it’s in-sync and has disk space. i.e our agent is connected and we can start using it. But as soon as we disconnect from it in the terminal (^C), it will disconnect. If we want it to work permanently, we need to create a service for it.

1. vi /var/jenkins\_home/service.sh
2. Paste the command to service.sh:

#!/bin/bash

java -jar agent.jar -url http://Lesson-23-external-R-1928713263.us-west-2.elb.amazonaws.com:8080/ -secret ba8ea6ab1e27260226741c0303c5b5ad1f01e42d236ddbc400686c9b9d334c60 -name docker -webSocket -workDir "/var/jenkins\_home"

1. chmod 755 /var/jenkins\_home/service.sh
2. vi /etc/systemd/system/jenkins\_agent.service

[Unit]

Description=Jenkins agent for docker related operations

After=network.target

[Service]

User=root

WorkingDirectory=/var/jenkins\_home

ExecStart=/bin/bash /var/jenkins\_home/service.sh

Type=forking

Restart=on-failure

[Install]

WantedBy=default.target

1. systemctl daemon-reload
2. systemctl status jenkins\_agent
3. Now to check the availability of the whole operation.
4. Connect to swarm1
5. We can check where jenkins is running with this command docker service ps jenkins
6. Shut jenkins service off to see how it restarts and available.
7. sudo -i
8. docker service ps jenkins
9. docker service scale jenkins=1 (shut off)
10. We can always look at the jenkins page to see if docker if available or has an “X”.