


Deployment 6

By following the task we created 3 EC2 instances 1-Master and 2-Agent EC2s

1. First EC2 is Master with ports ssh anywhere-22, Custom TCP 8080 and 500

**Amazon Linux**
Free tier eligible

Amazon Linux 2 AMI (HVM), SSD Volume Type - ami-087c17d1fe0178315 (64-bit x86) / ami-029c64b3c205e6cce (64-bit Arm)
Amazon Linux 2 comes with five years support. It provides Linux kernel 4.14 tuned for optimal performance on Amazon EC2, systemd 219, GCC 7.3, Glibc 2.26, Binutils 2.29.1, and the latest

[Select](#)
☒ 64-bit (x86)
☐ 64-bit (Arm)

1 to 44 of 44 AMIs

Subnet: Default in us-east-1a

Step 3: Configure Instance Details

Configure the instance to suit your requirements. You can launch multiple instances from the same AMI, request Spot instances to take advantage of, assign an access management role to the instance, and more.

Number of instances ⓘ [Launch into Auto Scaling Group](#) ⓘ

Purchasing option ⓘ ☐ Request Spot instances

Network ⓘ [Create new VPC](#)

Subnet ⓘ [Create new subnet](#)
4090 IP Addresses available

Add Tags:

Name Master

SSH	TCP	22	Anywhere	0.0.0.0/0, ::/0	e.g. S3
Custom TCP F	TCP	5000	Anywhere	0.0.0.0/0, ::/0	e.g. S3
Custom TCP F	TCP	8080	Custom	0.0.0.0/0, ::/0	e.g. S3

Add Tags

Finally, select keypair.pem and launch an instance

In the terminal ssh into that EC2 Master

- ssh -i keypair.pem ec2-user@PublicIPv4
- nano 8.sh
- Copy and past
User data script

```
sudo amazon-linux-extras install java-openjdk11
```

```
sudo amazon-linux-extras install epel
```

```
sudo wget -O /etc/yum.repos.d/jenkins.repo \
```

```
https://pkg.jenkins.io/redhat-stable/jenkins.repo
sudo rpm --import https://pkg.jenkins.io/redhat-stable/jenkins.io.key
sudo yum upgrade
sudo yum install epel-release java-11-openjdk-devel
sudo yum install jenkins
sudo systemctl start jenkins
```

- save and exit
- bash 8.sh (should start installation process all dependencies)

2. Second EC2 is Agent1 will use Ubuntu, AMI and port ssh -22, Custom TCP 5000

Selected: ubuntu

Step 1: Choose an Amazon Machine Image (AMI)

[Cancel and Exit](#)

An AMI is a template that contains the software configuration (operating system, application server, and applications) required to launch your instance. You can select an AMI provided by AWS, our user community, or the AWS Marketplace; or you can select one of your own AMIs.

Search by Systems Manager parameter

Quick Start (8)

- My AMIs (0)
- AWS Marketplace (883)
- Community AMIs (37847)

Ubuntu Server 20.04 LTS (HVM), SSD Volume Type - ami-09e67e426f25ce0d7 (64-bit x86) / ami-00d1ab6b335f217cf (64-bit Arm)

Select

☒ 64-bit (x86)
☐ 64-bit (Arm)

Free tier eligible Ubuntu Server 20.04 LTS (HVM),EBS General Purpose (SSD) Volume Type. Support available from Canonical (<http://www.ubuntu.com/cloud/services>).

Selected: t2.micro

Configure Instance: by default

Add tag: Name Agent1

Security group: ports ssh - 22 and Custom TCP - 5000

Step 6: Configure Security Group

A security group is a set of firewall rules that control the traffic for your instance. On this page, you can add rules to allow specific traffic to reach your instance, add rules that allow unrestricted access to the HTTP and HTTPS ports. You can create a new security group or select an existing one.

Assign a security group: ☒ Create a **new** security group
☐ Select an **existing** security group

Security group name:

Description:

Type	Protocol	Port Range	Source
SSH	TCP	22	Custom 0.0.0.0/0
Custom TCP F	TCP	5000	Custom CIDR, IP or Security Group

Add Rule

Finally, select keypair.pem and launch an instance

In the terminal ssh into that EC2 Agent1

- ls: 8.sh
- nano keypair.pem
- ls: 8.sh keypair.pem
- ssh -i keypair.pem ec2-user@Agent1 Public IPv4
- chmod 4000 keypair.pem
- ls: 8.sh keypair.pem
- ssh -i keypair.pem ec2-user@18.233.98.167 (Permission denied (publickey))
- cat keypair.pem
- copy keypair.pem (open with TextEdit) and past
- ssh -i keypair.pem ubuntu@Agent1 Public IPv4
- nano a.sh
- Copy and past
User data script

```
#!/bin/bash
```

```
sudo apt-get update && sudo apt-get upgrade -y
```

```
sudo apt-get install -y \
```

```
default-jre \
```

```
git \
```

```
nodejs -y \
```

```
npm -y
```

```
sudo apt install default-jre git nodejs npm
```

- Save and exit
- bash a.sh (should start installation process all dependencies)
- node --version (Command 'node' not found)
- sudo apt install default-jre git nodejs npm
- Exit
- Then check
- maven --version (command not found)
- java --version (already installed)
- Open new tab and go to the Jenkins using Master Public IPv4 and port 8080:
Ipv4:8080
- Copy the path and paste to terminal using cat command:
sudo cat /var/lib/jenkins/secrets/initialAdminPassword

```
93cbd89e096b4d008952bbfe0099f18c
```

- Log in to the Jenkins
- Install suggested plugins
- Create an account
- We need to download plugins go to Manage Jenkins
- Manage Plugins

Manage Jenkins

Building on the controller node can be a security issue. You should set the number of executors on the controller to 0. See [the documentation](#).

[Manage](#)
[Dismiss](#)

System Configuration



Configure System

Configure global settings and paths.



Global Tool Configuration

Configure tools, their locations and automatic installers.



Manage Plugins

Add, remove, disable or enable plugins that can extend the functionality of Jenkins.

- Install: NodeJs, Amazon EC2, Maven Integration
- Select: Download now and install after restart

3. Third EC2 is Agent2 will use Ubuntu, AMI and port ssh -22

- Select ubuntu
- t2 micro
- Add tag: name Agent2
- Configure SG: SSH port 22
- Select keypair and launch it.
- SSH third EC2 go to terminal
- ssh -i keypair.pem ubuntu@Agent2 Public IPv4
- nano a.sh
- Copy and paste

```
#!/bin/bash
```

```
sudo apt-get update && sudo apt-get upgrade -y
```

```
sudo apt-get install -y \
```

```
default-jre \
```

```
git \
```

```
nodejs -y \
```

```
npm -y \
```

```
maven \
```

```
libgtk2.0-0 \
```

```
libgtk-3-0 \
```

```
libgbm-dev \
```

```
libnotify-dev \
```

```
libgconf-2-4 \
```

libnss3 \
libxss1 \
libasound2 \
libxtst6 \
xauth \
xvfb

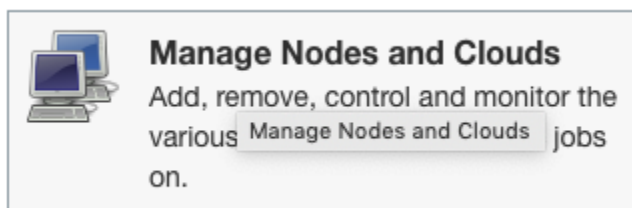
- Save and exit
- bash a.sh
- nano b.sh
- Copy and paste

sudo apt install default-jre git nodejs npm maven libgtk2.0-0 libgtk-3-0 libgbm-dev
libnotify-dev libgconf-2-4 libnss3 libxss1 libasound2 libxtst6 xauth xvfb

- Save and exit
- bash b.sh (should start installation process all dependencies)
- sudo apt install libnotify-dev
- Exit
- sudo yum install git
- Exit
- sudo apt install xvfb

Go to Jenkins

- Dashboard
- Manage Jenkins
- Manage Nodes and Clouds



- New Node
- Node name: anything my Ubuntu1
- Select: Permanent Agent and OK

Name

Ubuntu1

Description

Deployment6 - ubuntu1

Number of executors

2

Remote root directory

/home/ubuntu/jenkins/app

Labels

agent-ubuntu1

Usage

Use this node as much as possible

Launch method

Launch agents via SSH

The host should be the private IPv4 of the Agent1

Host

172.31.95.74

Credentials

- none - ▾

 Add ▾

 **The selected credentials cannot be found**

Add Credentials



Add Credentials

Domain

Global credentials (unrestricted)

Kind

SSH Username with private key

Scope

Global (Jenkins, nodes, items, all child items, etc)

ID

agent1

- User name: ubuntu
- Select: Enter directly
- Key: copy keypair.pem (open with TextEdit) and past
- Add

- Credentials -none-: select GitHub acc name

^

Labels

agent-ubuntu1

Usage

Use this node as much as possible

Launch method


Launch agents via SSH

Host

172.31.95.74

Credentials

ubuntu ▾

 Add ▾

Host Key Verification Strategy

Non verifying Verification Strategy

Availability

Keep this agent online as much as possible

Node Properties

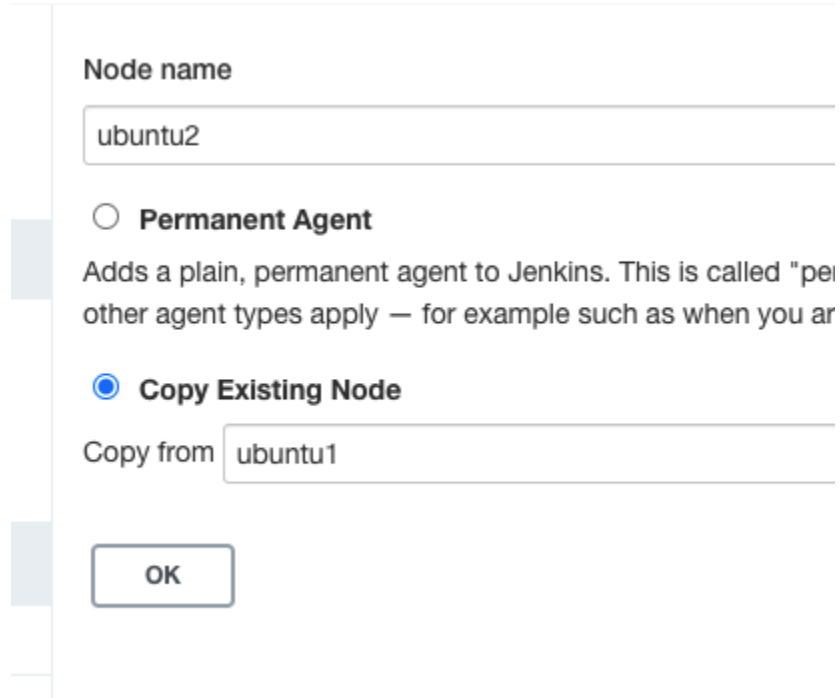
☐ Disable deferred wipeout on this node

☐ Environment variables

☐ Tool Locations

Save

- Save
- Create the second node
- New Node
- Node name: anything my Ubuntu2
- Select: Permanent Agent and OK



Node name

ubuntu2

☐ **Permanent Agent**

Adds a plain, permanent agent to Jenkins. This is called "permanent agent" and other agent types apply — for example such as when you are using a cloud provider.

☒ **Copy Existing Node**

Copy from

OK

-
- Host: Host should be the private IPv4 of the Agent2
- Select ssh credentials
- Username: Ubuntu
- Key used to log in

Name

ubuntu2

Description

Number of executors

2

Remote root directory

/home/ubuntu/jenkins/app

Labels

agent-ubuntu2

Usage

Use this node as much as possible

Launch method

Launch agents via SSH

Host

172.31.86.182

 **The Host must be specified**

Credentials

ubuntu ▾

 Add ▾

Host Key Verification Strategy

Save

- Save it

- Got to the master
-

Jenkins Dashboard > Nodes

S	Name	Architecture	Clock Difference	Free Disk Space	Free Swap Space	Free Temp Space	Response Time
	master	Linux (amd64)	In sync	5.21 GB	0 B	5.21 GB	0ms
	Ubuntu1	Linux (amd64)	In sync	5.48 GB	0 B	5.48 GB	57ms
	ubuntu2	Linux (amd64)	In sync	4.94 GB	0 B	4.94 GB	44ms
Data obtained			3.1 sec	3.1 sec	3 sec	3.1 sec	3.1 sec

Build Queue
No builds in the queue.

Refresh status

Download the repository from Kura GitHub https://github.com/kura-labs-org/DEPLOY6_FE - Code - Download ZIP

Go to file Add file Code

Clone

HTTPS SSH GitHub CLI

`https://github.com/kura-labs-org/DEPLO`

Use Git or checkout with SVN using the web URL.

Open with GitHub Desktop

Download ZIP

Go to the Jenkinsfile and edit it, enter your label name for your first agent1

```
1  pipeline {
2    agent {
3      label 'agent-ubuntu1'
4    }
5    stages {
6      stage ('Build') {
7        steps {
8          sh 'rm -rf ./kura_test_repo/cypress2'
9          sh '''
10             npm install
11             npm run build
12             sudo npm install -g serve
13             serve -s build &
14             '''
15        }
16      }
17      stage ('Second') {

17      stage ('Second') {
18        agent {
19          label 'agent-ubuntu2'
20        }
21        steps {
22          sh '''
23            npm install cypress
24            npm install mocha
25            npx cypress run --spec ./cypress/integration/test.spec.js
26            '''
27          }
28          post {
29            always {
30              junit 'results/cypress-report.xml'
31            }
32          }
33        }
34      }
35    }
36  }
```

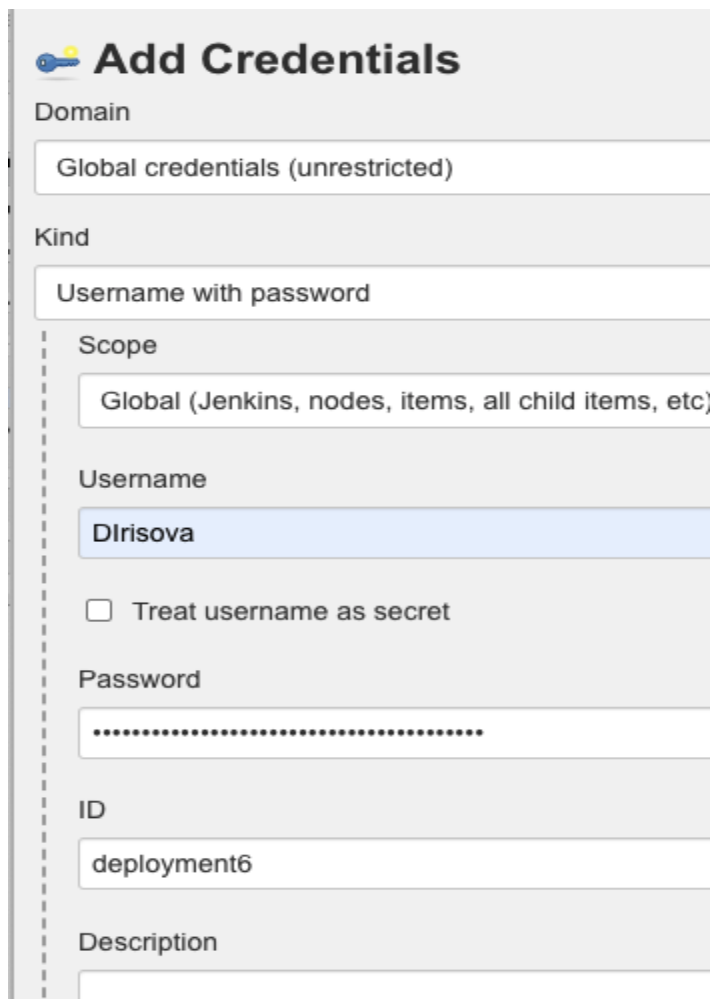
Go to the cypress/integration/test.spec.js and edit line 3, change the IP address to the private IP agent1

```

1  describe('Heading', () => {
2      it('has the right title', () => {
3          cy.visit('http://172.31.95.74:5000/example-1')
4
5          cy.get('h1')
6              .invoke('text')
7              .should("equal", "My Awesome Web Application")
8          Cypress.Screenshot.defaults({
9              capture: 'runner',
10          })
11          cy.screenshot();
12      });
13
14  });

```

Go to the Jenkins to run the build
 Create new item
 Select Multibranch pipeline project -Deploy6



The image shows the 'Add Credentials' form in Jenkins. It has a key icon and the title 'Add Credentials'. The form is divided into sections by dashed lines. The 'Domain' section has a dropdown menu with 'Global credentials (unrestricted)' selected. The 'Kind' section has a dropdown menu with 'Username with password' selected. The 'Scope' section has a dropdown menu with 'Global (Jenkins, nodes, items, all child items, etc)' selected. The 'Username' section has a text input field with 'Dlrisova' entered. Below it is a checkbox labeled 'Treat username as secret' which is unchecked. The 'Password' section has a password input field with dots. The 'ID' section has a text input field with 'deployment6' entered. The 'Description' section is empty.

The username should be your GitHub username

For passwords, you can create new or use already existed your GitHub Personal Access Token

To create a new password go to the GitHub/Settings/Developer settings/Personal access tokens

Add

Select project repository

Select the credentials

General Branch Sources Build Configuration Scan Multibranch Pipeline Triggers Orphaned Item Strat

Properties Pipeline Libraries

Display Name

Deploy6

Description

[Plain text] [Preview](#)

Disable

☐ (No new builds within this Multibranch Pipeline will be executed until it is re-enabled)

Branch Sources

Git

Project Repository ?

https://github.com/DIrisova/DEPLOY6_FE.git

Credentials ?

DIrisova/***** [Add](#)

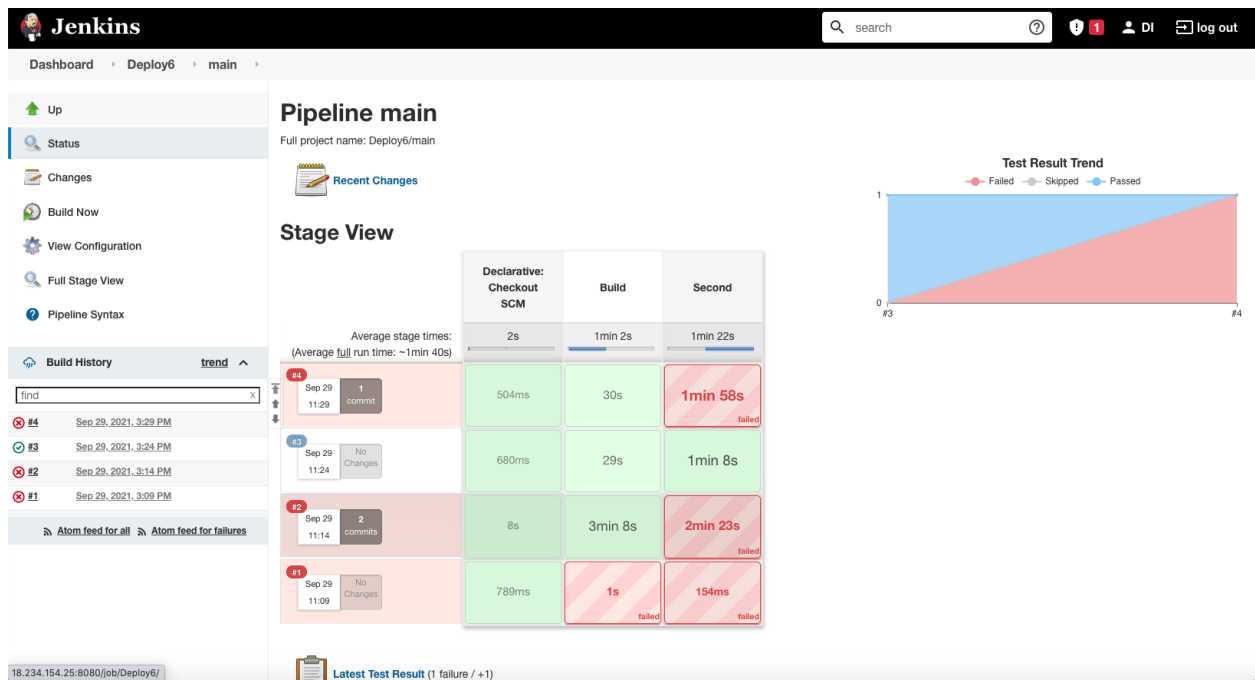
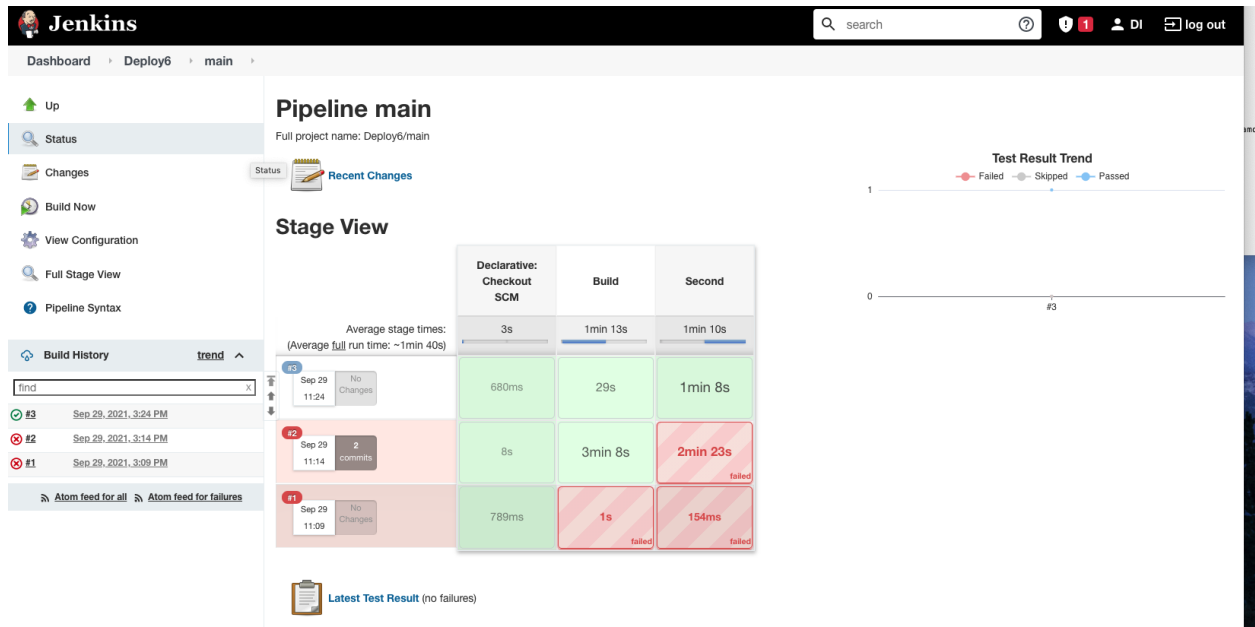
Behaviors

Discover branches ?

[Add](#) [Save](#) [Apply](#)

Save

Finally, build and test



Next, we need to find the video that cypress created
Go to the terminal:

- find /home -name *.mp4

/home/ubuntu/jenkins/app/workspace/Deploy6_main/cypress/videos/test.spec.js.mp4

- cd /home/ubuntu/jenkins/app/workspace/Deploy6_main/cypress
- ls
- mv videos screenshots ~
- cd ~
- ls
- ssh-keygen
- ls
- cat video1.pub
- eval `ssh-agent -s`
- ssh-add video1.pub
- ssh-add video1
- git clone git@github.com:DIrisova/DEPLOY6_FE.git
- mv screenshots videos DEPLOY6_FE
- cd DEPLOY6_FE
- git status
- git add .
- git commit -m "add"
- git push origin main