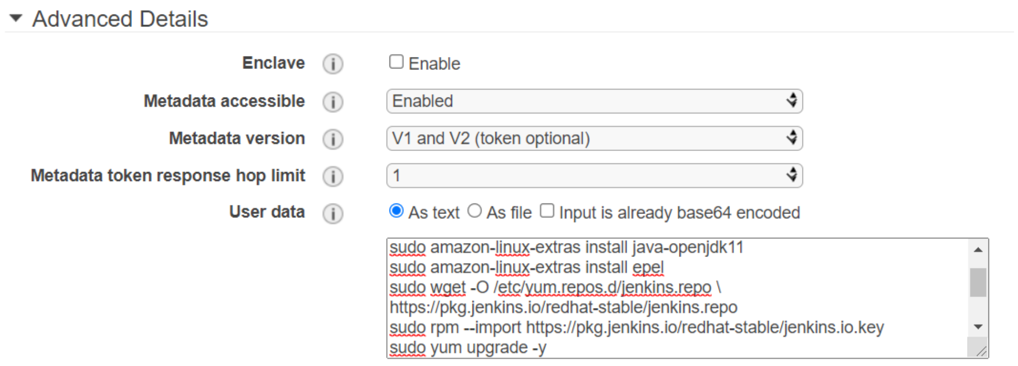
Steps:

1. Create your Jenkins master ec2 using the following script

```
#!/bin/bash

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 -y

sudo yum install epel-release java-11-openjdk-devel -y

sudo yum install jenkins -y

sudo systemctl start jenkins

sudo yum install git -y
```

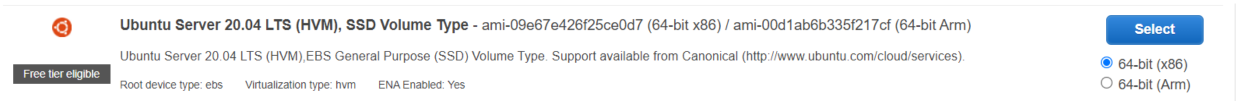


Advanced Details

Enclave	☐ Enable
Metadata accessible	Enabled
Metadata version	V1 and V2 (token optional)
Metadata token response hop limit	1
User data	● As text ○ As file ☐ Input is already base64 encoded

```
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 -y
```

2. Once done ssh into your instance using (ssh -I key.pem ec2-user@PublicIPv4).
3. Create the ec2 using the Ubuntu Ami.



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

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

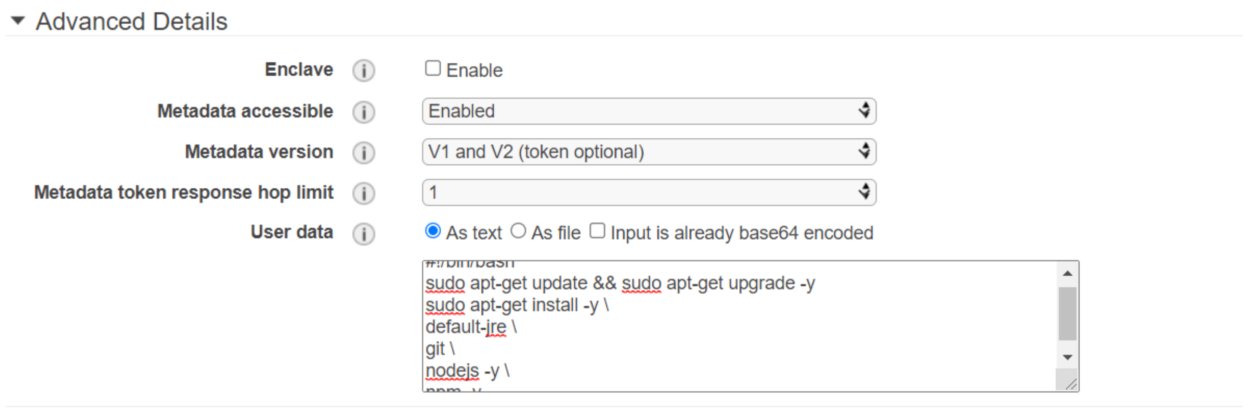
Root device type: ebs   Virtualization type: hvm   ENA Enabled: Yes

Free tier eligible

● 64-bit (x86)
○ 64-bit (Arm)

4. This time for the first agent, use this script:

```
#!/bin/bash

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

sudo apt-get install -y \

default-jre \

git \

nodejs -y \

npm -y
```

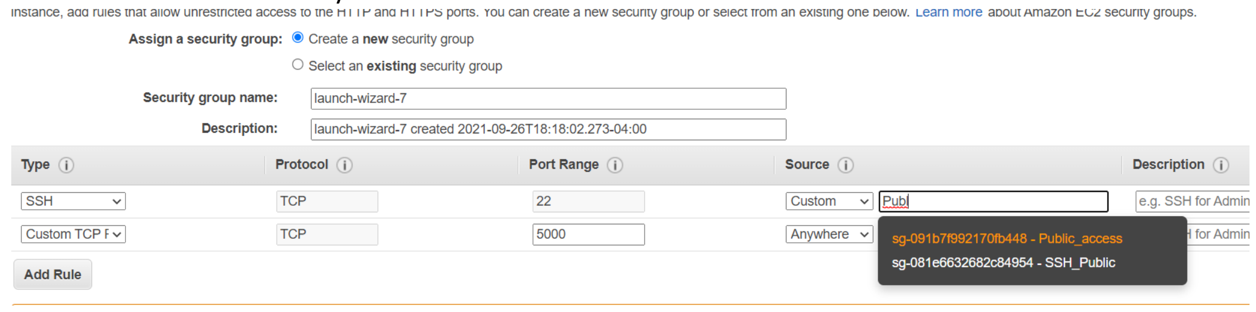


Advanced Details

Enclave	☐ Enable
Metadata accessible	Enabled
Metadata version	V1 and V2 (token optional)
Metadata token response hop limit	1
User data	[✓ As text] ○ As file ☐ Input is already base64 encoded

```
#!/bin/bash
sudo apt-get update && sudo apt-get upgrade -y
sudo apt-get install -y \
default-jre \
git \
nodejs -y \
npm -y
```

5. Give your instance a name. Then configure the security group using the public security made in for the Jenkins master. In my case "Public access".



instance, add rules that allow unrestricted access to the HTTP and HTTPS ports. You can create a new security group or select from an existing one below. Learn more about Amazon EC2 security groups.

Assign a security group: [✓ Create a **new** security group] ○ Select an **existing** security group

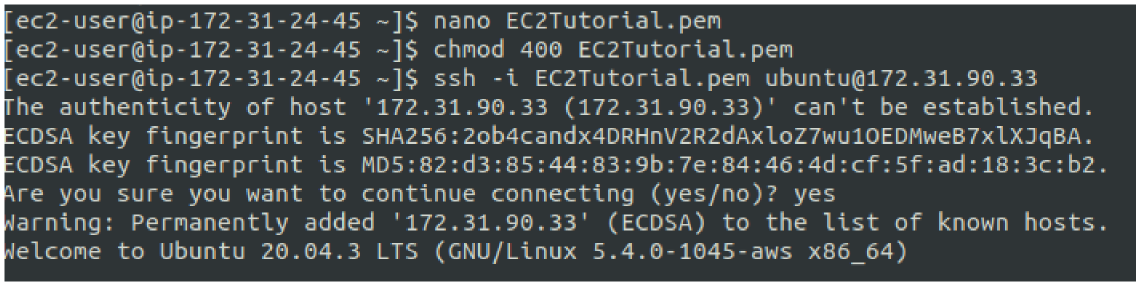
Security group name: launch-wizard-7

Description: launch-wizard-7 created 2021-09-26T18:18:02.273-04:00

Type	Protocol	Port Range	Source		Description
SSH	TCP	22	Custom	Publ	e.g. SSH for Admin
Custom TCP F	TCP	5000	Anywhere		for Admin

sg-091b7f992170fb448 - Public_access
sg-081e6632682c84954 - SSH_Public

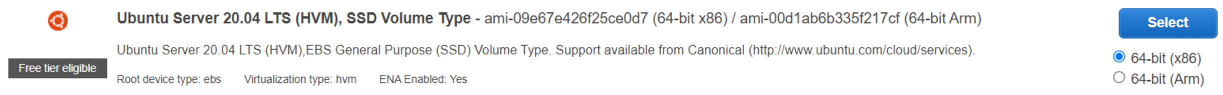
6. Now ssh into your ec2 instance from your Jenkins instance
7. Create a new key.pem by using nano ((name-of-key).pem).
8. Paste the RSA key into that file
9. Change permissions using the chmod 400. Example chmod 400 EC2Tutorial.pem
10. Then use the command, ssh -i EC2Tutorial.pem **ubuntu**@privateIPv4.
    NOTE: the privateIPv4 of the first agent required here. This is shown below.



```
[ec2-user@ip-172-31-24-45 ~]$ nano EC2Tutorial.pem
[ec2-user@ip-172-31-24-45 ~]$ chmod 400 EC2Tutorial.pem
[ec2-user@ip-172-31-24-45 ~]$ ssh -i EC2Tutorial.pem ubuntu@172.31.90.33
The authenticity of host '172.31.90.33 (172.31.90.33)' can't be established.
ECDSA key fingerprint is SHA256:2ob4candx4DRHnV2R2dAxloZ7wu1OEDMweB7xlXJqBA.
ECDSA key fingerprint is MD5:82:d3:85:44:83:9b:7e:84:46:4d:cf:5f:ad:18:3c:b2.
Are you sure you want to continue connecting (yes/no)? yes
Warning: Permanently added '172.31.90.33' (ECDSA) to the list of known hosts.
Welcome to Ubuntu 20.04.3 LTS (GNU/Linux 5.4.0-1045-aws x86_64)
```

11. Using the command “nano script.sh”, paste the bash script into script created
12. Use the command “bash script.sh”. This allows you to download the dependencies from the script created.

13. Create a third instance using the Ubuntu AMI.



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

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

Root device type: ebs   Virtualization type: hvm   ENA Enabled: Yes

14. This time for the second agent, use this script:

#!/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 \

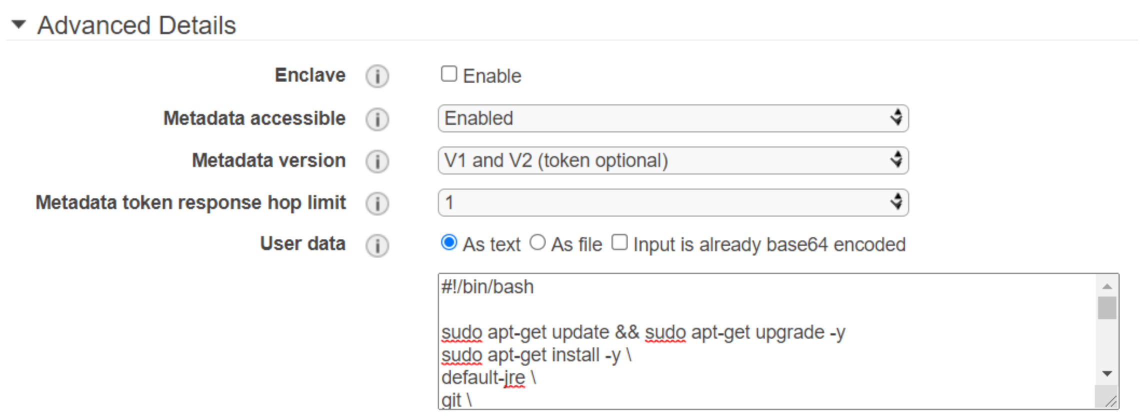
libxss1 \

libasound2 \

libxtst6 \

xauth \

xvfb

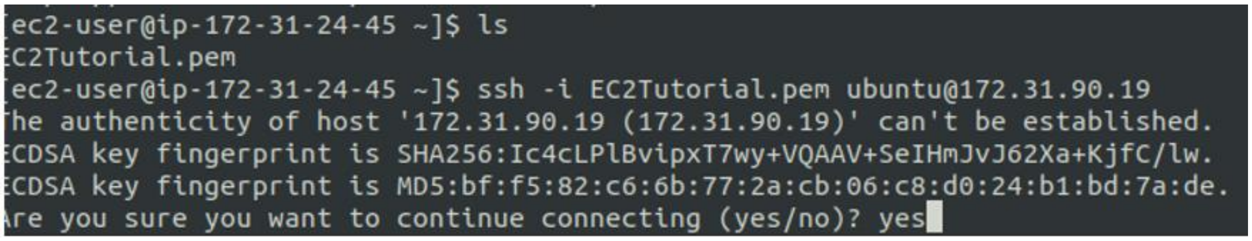


Advanced Details

Enclave	☐ Enable
Metadata accessible	Enabled
Metadata version	V1 and V2 (token optional)
Metadata token response hop limit	1
User data	● As text ○ As file ☐ Input is already base64 encoded

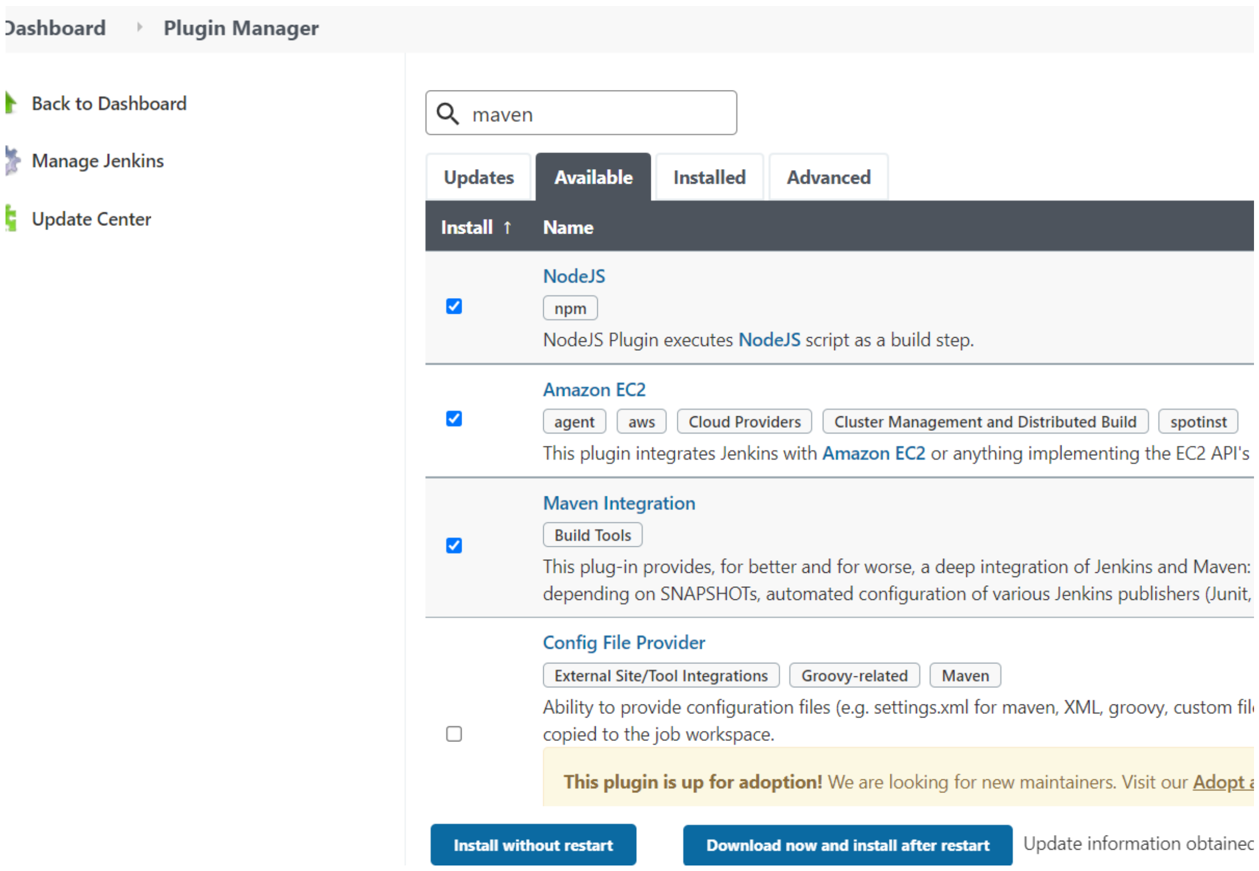
```
#!/bin/bash

sudo apt-get update && sudo apt-get upgrade -y
sudo apt-get install -y \
default-jre \
git \
```

15. Name the instance and set the security group.
16. Then ssh into the Jenkins master and repeat steps 6 to 10.
    NOTE: the privateIPv4 of the second agent required here. This is shown below.



```
[ec2-user@ip-172-31-24-45 ~]$ ls
EC2Tutorial.pem
[ec2-user@ip-172-31-24-45 ~]$ ssh -i EC2Tutorial.pem ubuntu@172.31.90.19
The authenticity of host '172.31.90.19 (172.31.90.19)' can't be established.
ECDSA key fingerprint is SHA256:Ic4cLPlBvipxT7wy+VQAAV+SeIHmJvJ62Xa+KjfC/lw.
ECDSA key fingerprint is MD5:bf:f5:82:c6:6b:77:2a:cb:06:c8:d0:24:b1:bd:7a:de.
Are you sure you want to continue connecting (yes/no)? yes
```

17. Using the command “nano script.sh”, paste the bash script into script created
18. Use the command “bash script.sh”. This allows you to download the dependencies from the script created.
19. After successfully setting up all the EC2s, connect to your Jenkins application using the PublicIPv4 of the Jenkins Master and add port 8080 (PublicIPv4:8080).
20. Once successfully logged in, install the recommended plugins.
21. After creating your account, download the plugins for Nodejs, Amazone EC2 and Maven as shown below:


Dashboard › Plugin Manager

Back to Dashboard
Manage Jenkins
Update Center

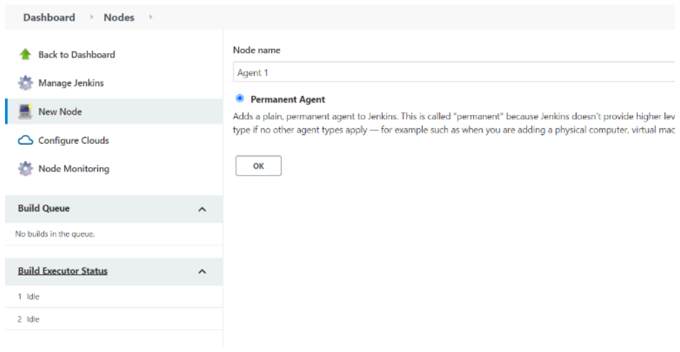
maven

Updates | Available | Installed | Advanced

Install ↑	Name
✓	NodeJS npm NodeJS Plugin executes NodeJS script as a build step.
✓	Amazon EC2 agent · aws · Cloud Providers · Cluster Management and Distributed Build · spotinst This plugin integrates Jenkins with Amazon EC2 or anything implementing the EC2 API's
✓	Maven Integration Build Tools This plug-in provides, for better and for worse, a deep integration of Jenkins and Maven: depending on SNAPSHOTs, automated configuration of various Jenkins publishers (Junit,
☐	Config File Provider External Site/Tool Integrations · Groovy-related · Maven Ability to provide configuration files (e.g. settings.xml for maven, XML, groovy, custom fil copied to the job workspace. **This plugin is up for adoption!** We are looking for new maintainers. Visit our Adopt a

Install without restart | Download now and install after restart | Update information obtainec

22. Once selected choose the option to “Download now and install after reset”.
23. Once done, create two different Agents on Jenkins using Nodes.
24. Go to manage Jenkins. There you will select Manage Nodes and Clouds.
25. Click “New Node” to create the first agent. Give it a name as shown below:


Dashboard › Nodes ›

Back to Dashboard
Manage Jenkins
New Node
Configure Clouds
Node Monitoring

Build Queue
No builds in the queue.

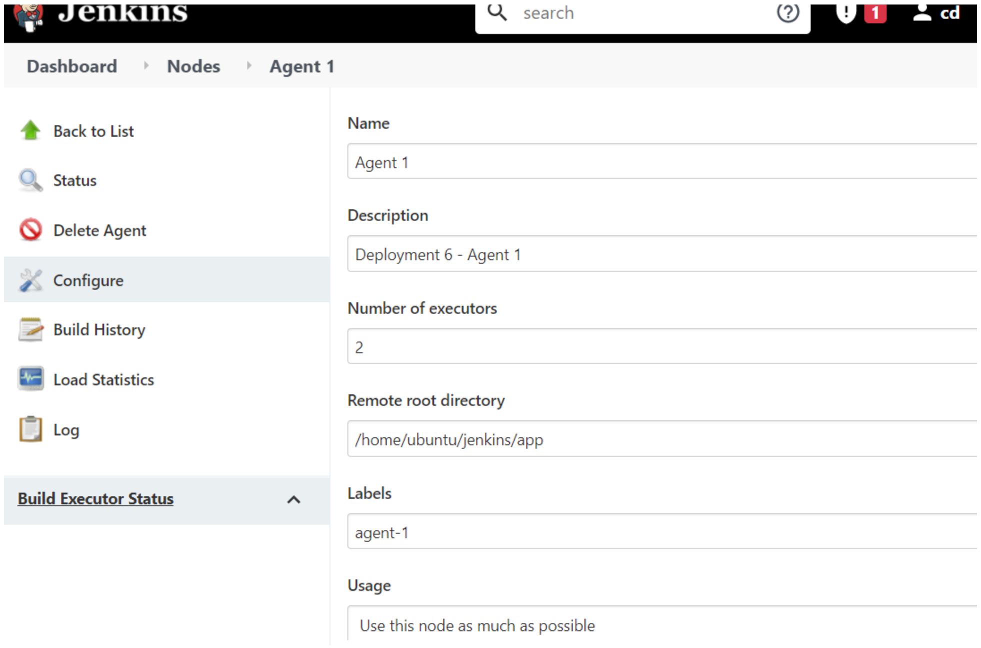
Build Executor Status
1 Idle
2 Idle

Node name
Agent 1

● Permanent Agent
Adds a plain, permanent agent to Jenkins. This is called "permanent" because Jenkins doesn't provide higher lev
type if no other agent types apply — for example such as when you are adding a physical computer, virtual mac

OK

26. Give your first agent the following setting:



Name: Agent 1

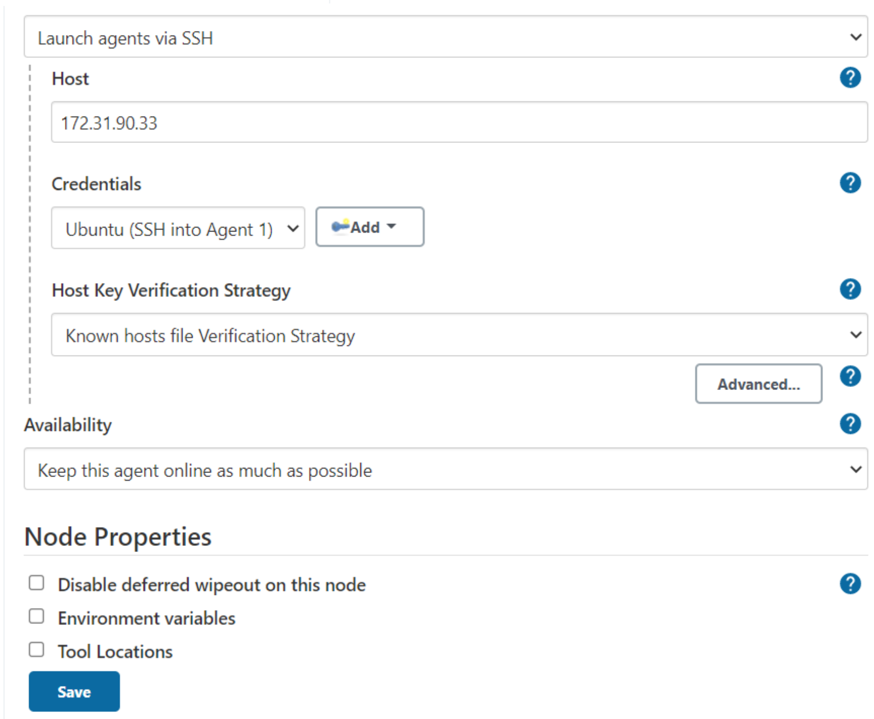
Description: Deployment 6 - Agent 1

Number of executors: 2

Remote root directory: /home/ubuntu/jenkins/app

Labels: agent-1

Usage: Use this node as much as possible



Launch agents via SSH

Host: 172.31.90.33

Credentials: Ubuntu (SSH into Agent 1)

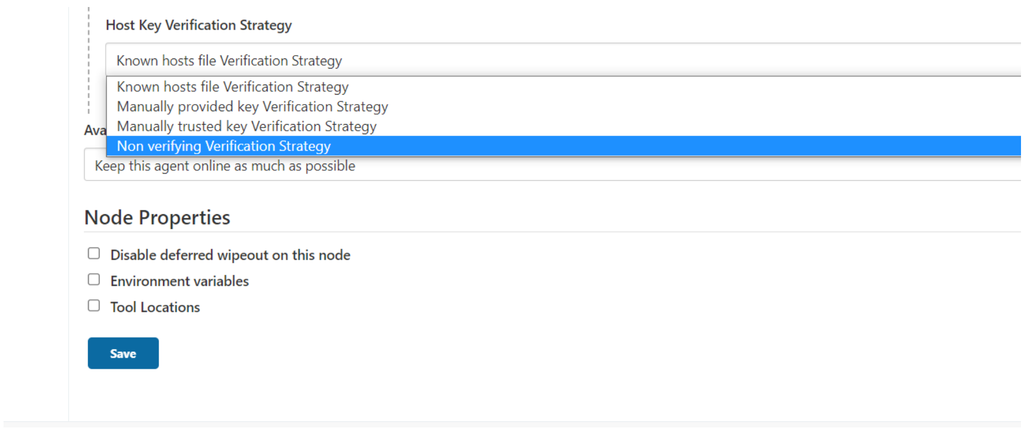
Host Key Verification Strategy: Known hosts file Verification Strategy

Availability: Keep this agent online as much as possible

Node Properties

- [ ] Disable deferred wipeout on this node
- [ ] Environment variables
- [ ] Tool Locations

NOTE: wrong Host Key verification Strategy which resulted in my agent unable to connect to my Jenkins master. Ensure you have this selected:


Host Key Verification Strategy

Known hosts file Verification Strategy

- Known hosts file Verification Strategy
- Manually provided key Verification Strategy
- Manually trusted key Verification Strategy
- Non verifying Verification Strategy

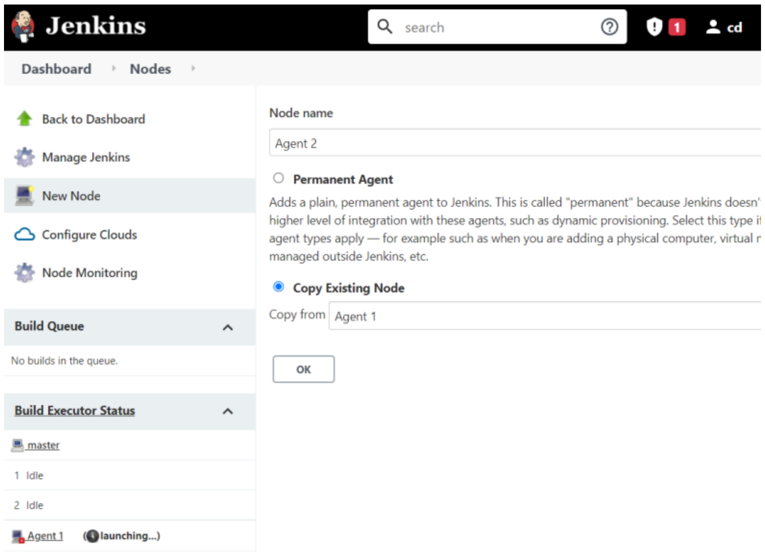
Ava

Keep this agent online as much as possible

Node Properties

- [ ] Disable deferred wipeout on this node
- [ ] Environment variables
- [ ] Tool Locations

27. Once done save changes.
28. After configure Agent 2 as shown:



Node name

Agent 2

○ Permanent Agent

Adds a plain, permanent agent to Jenkins. This is called "permanent" because Jenkins doesn' higher level of integration with these agents, such as dynamic provisioning. Select this type i agent types apply — for example such as when you are adding a physical computer, virtual r managed outside Jenkins, etc.

● Copy Existing Node

Copy from Agent 1

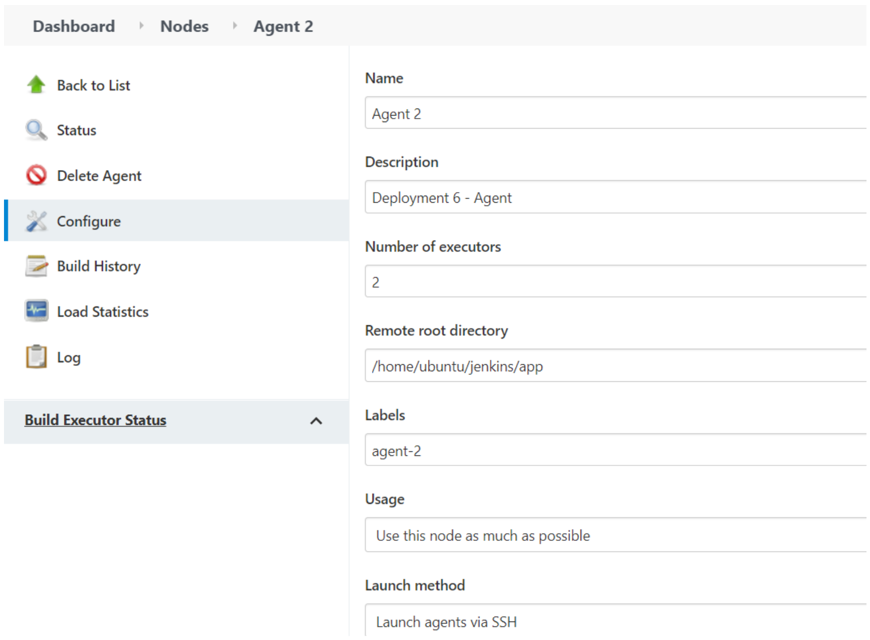
No builds in the queue.

master

1 Idle

2 Idle

Agent 1 (launching...)



Name

Agent 2

Description

Deployment 6 - Agent

Number of executors

2

Remote root directory

/home/ubuntu/jenkins/app

Labels

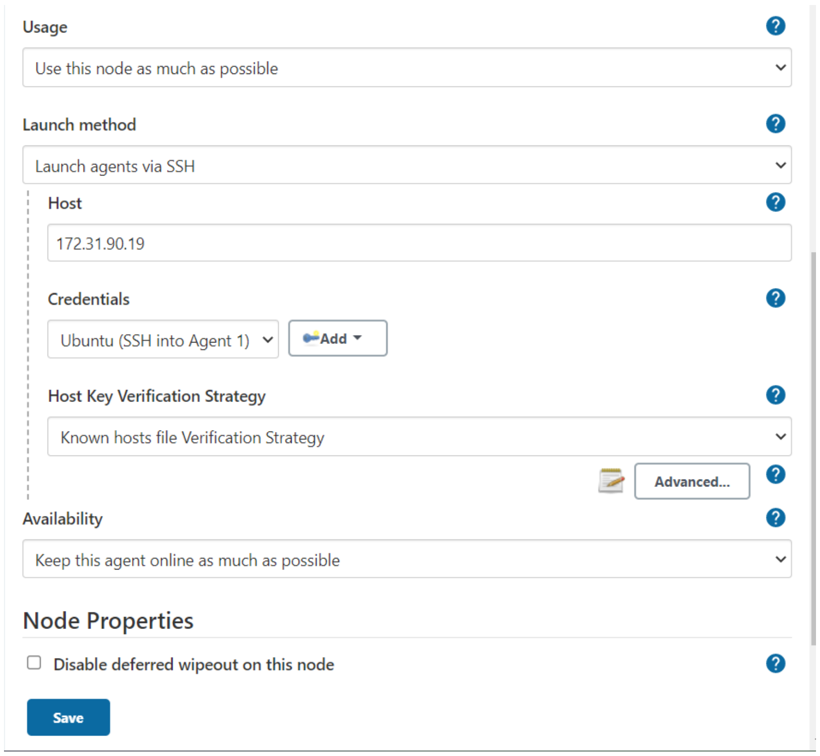
agent-2

Usage

Use this node as much as possible

Launch method

Launch agents via SSH


Usage

Use this node as much as possible

Launch method

Launch agents via SSH

Host

172.31.90.19

Credentials

Ubuntu (SSH into Agent 1)

Host Key Verification Strategy

Known hosts file Verification Strategy

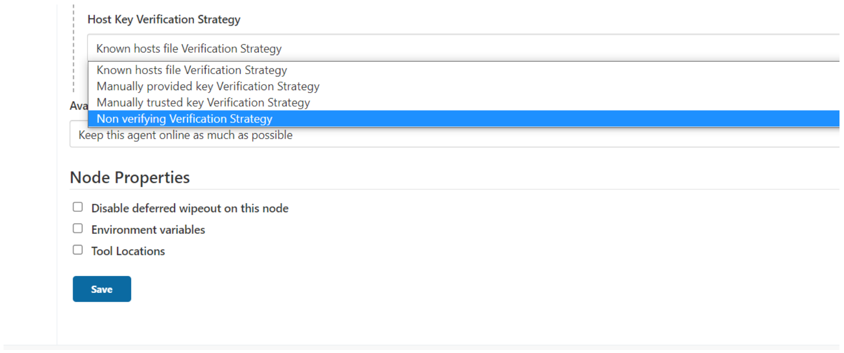
Availability

Keep this agent online as much as possible

Node Properties

- [ ] Disable deferred wipeout on this node

NOTE: wrong Host Key verification Strategy which resulted in my agent unable to connect to my Jenkins master. Ensure you have this selected:


Host Key Verification Strategy

Known hosts file Verification Strategy

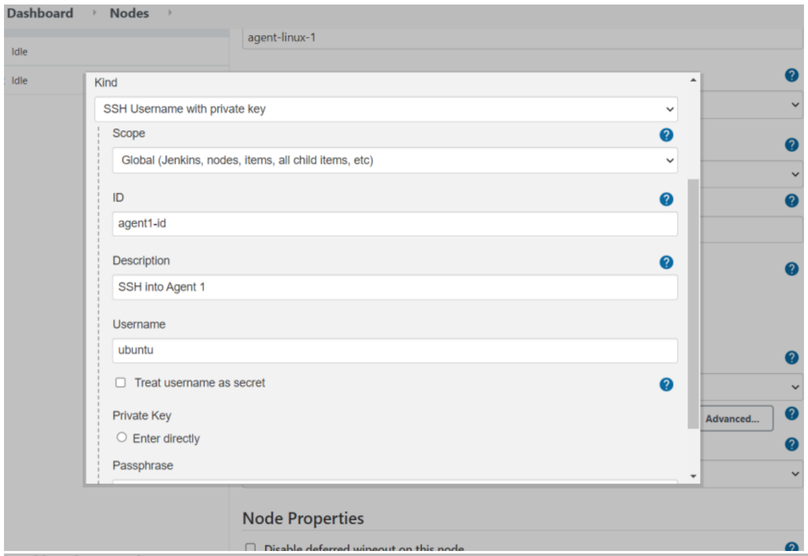
Known hosts file Verification Strategy
Manually provided key Verification Strategy
Manually trusted key Verification Strategy
Non verifying Verification Strategy

Keep this agent online as much as possible

Node Properties

- [ ] Disable deferred wipeout on this node
- [ ] Environment variables
- [ ] Tool Locations

29. Once done saves the changes.
30. Now add your credentials as shown:



Dashboard › Nodes ›

agent-linux-1

Kind: SSH Username with private key

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

ID: agent1-id

Description: SSH into Agent 1

Username: ubuntu

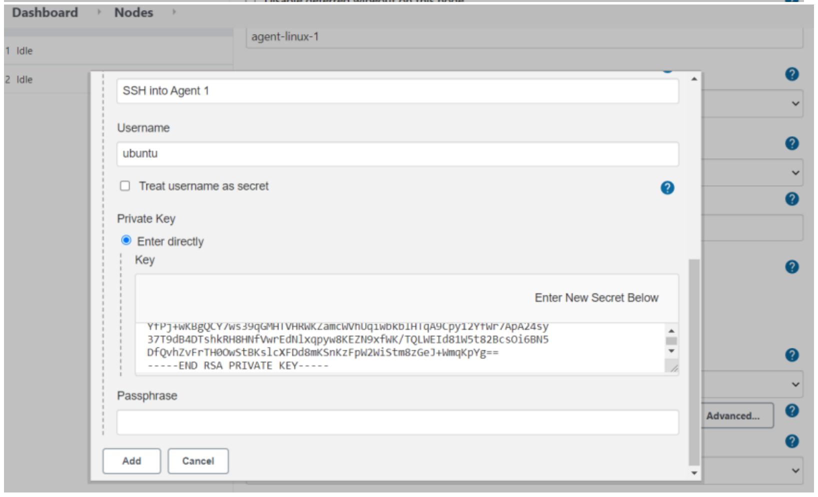
☐ Treat username as secret

Private Key

○ Enter directly

Passphrase

Node Properties



Dashboard › Nodes ›

agent-linux-1

SSH into Agent 1

Username: ubuntu

☐ Treat username as secret

Private Key

● Enter directly

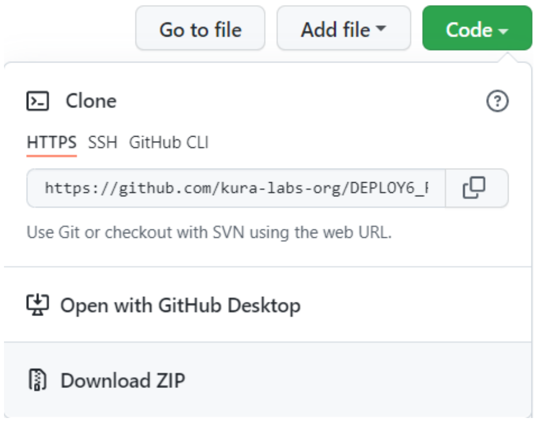
Key

Enter New Secret Below

```
YfPj+wKBgQCY/ws39qGMHTVHRWKZamcWVhUq1WbKblHTqA9Cpy12YfWr7ApA24sy
37T9dB4DTshkRH8HNfVwrEdNlxqpyw8KEZN9xfWK/TQLWEId81W5t82BcsOi6BN5
DfQvhZvFrTH0OwStBKslcXFDd8mKSnKzFpW2WiStm8zGeJ+WmqKpYg==
-----END RSA PRIVATE KEY-----
```

Passphrase

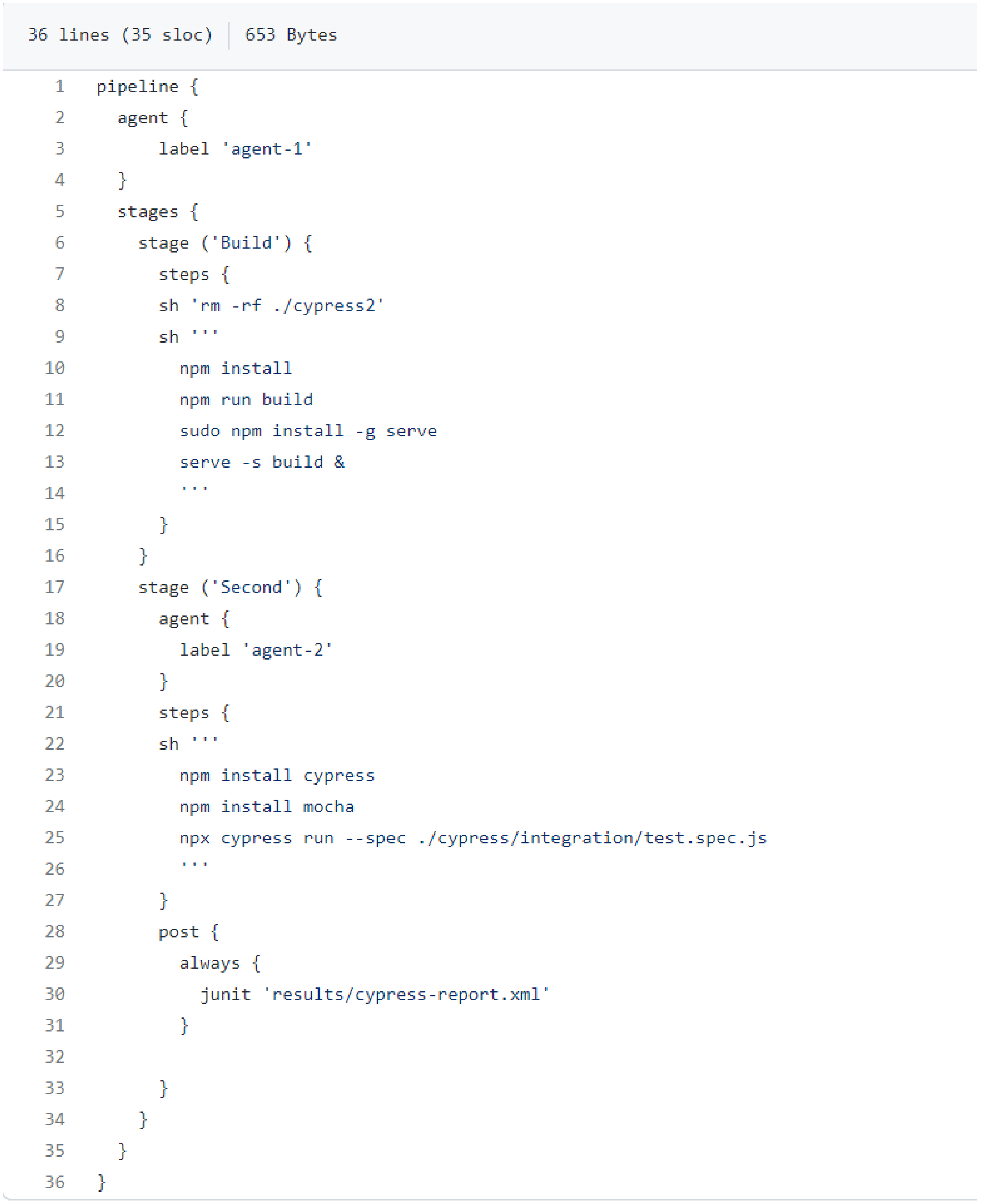
31. Once done, click add and select the credentials you created.
32. Then go to the repository and download it by clicking “Download ZIP”.



https://github.com/kura-labs-org/DEPLOY6_

Use Git or checkout with SVN using the web URL.

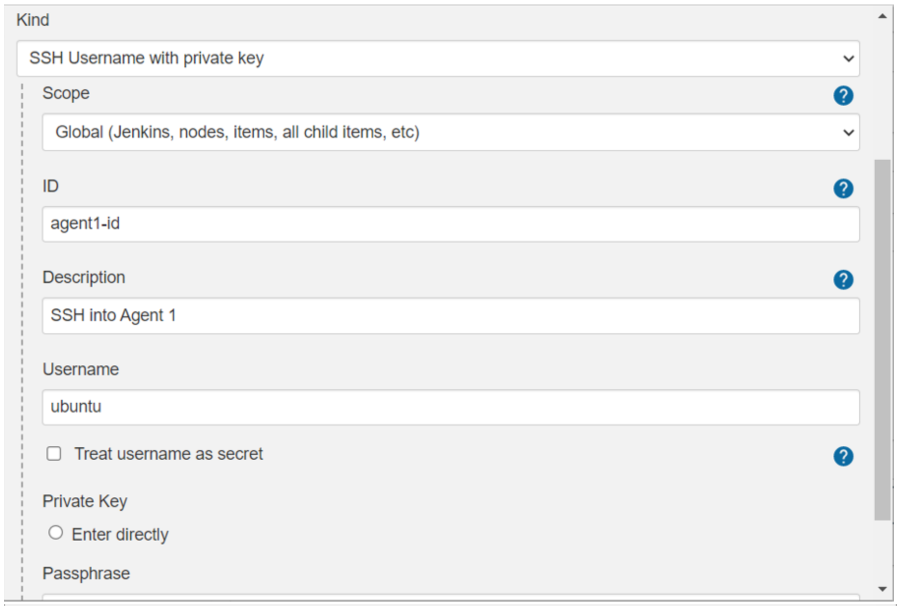
33. After it's downloaded, extract the files and delete both the README.md and Document#6.pdf.
34. Create a new repository on Github and upload the contents from the folder you downloaded.
35. Then edit the Jenkinsfile ensuring that the label names given are those you given to the Node(Jenkins) in label setting as shown:



36 lines (35 sloc) | 653 Bytes

```
pipeline {
  agent {
      label 'agent-1'
  }
  stages {
    stage ('Build') {
      steps {
      sh 'rm -rf ./cypress2'
      sh '''
        npm install
        npm run build
        sudo npm install -g serve
        serve -s build &
        '''
      }
    }
    stage ('Second') {
      agent {
        label 'agent-2'
      }
      steps {
      sh '''
        npm install cypress
        npm install mocha
        npx cypress run --spec ./cypress/integration/test.spec.js
        '''
      }
      post {
        always {
          junit 'results/cypress-report.xml'
        }

      }
    }
  }
}
```

36. Go back to Jenkins and create a new item.
37. Give it a name then select Multibranch pipeline
38. Add the link to your Git Repository then create your credentials using Jenkins.
39. For username, use your Github username and for password use your Github Personal Access Token.



Kind: SSH Username with private key

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

ID: agent1-id

Description: SSH into Agent 1

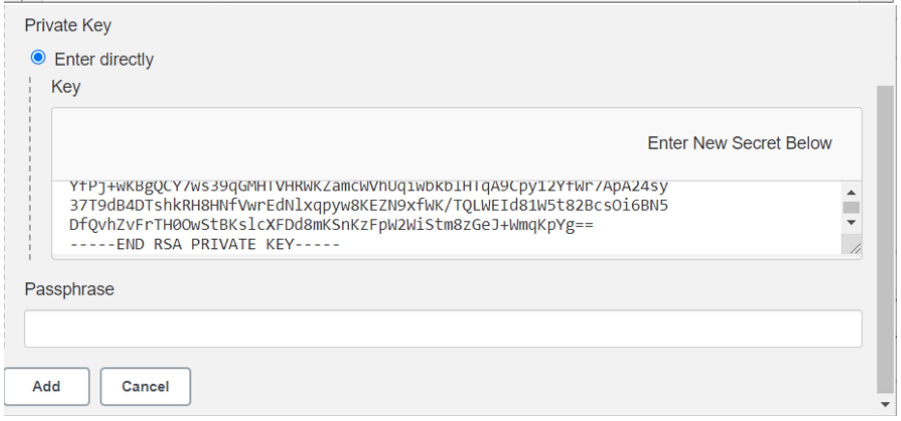
Username: ubuntu

☐ Treat username as secret

Private Key

○ Enter directly

Passphrase



Private Key

● Enter directly

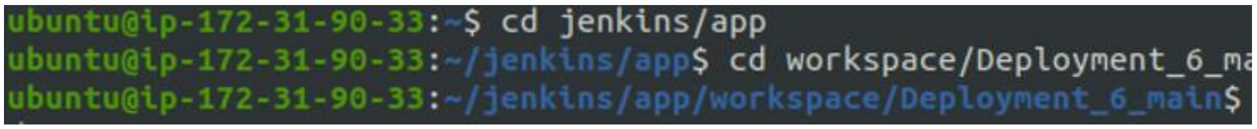
Key

Enter New Secret Below

```
YfPj+wKBgQCY7ws39qGMHTVHRWKZamcWVhUq1wbkblHTqA9Cpy12YfWr7ApA24sy
37T9dB4DTshkRH8HNfVwrEdNlxqpyw8KEZN9xfWK/TQLWEId81W5t82BcsOi6BN5
DfQvhZvFrTH0OwStBKslcXFDd8mKSnKzFpW2WiStm8zGeJ+WmqKpYg==
-----END RSA PRIVATE KEY-----
```

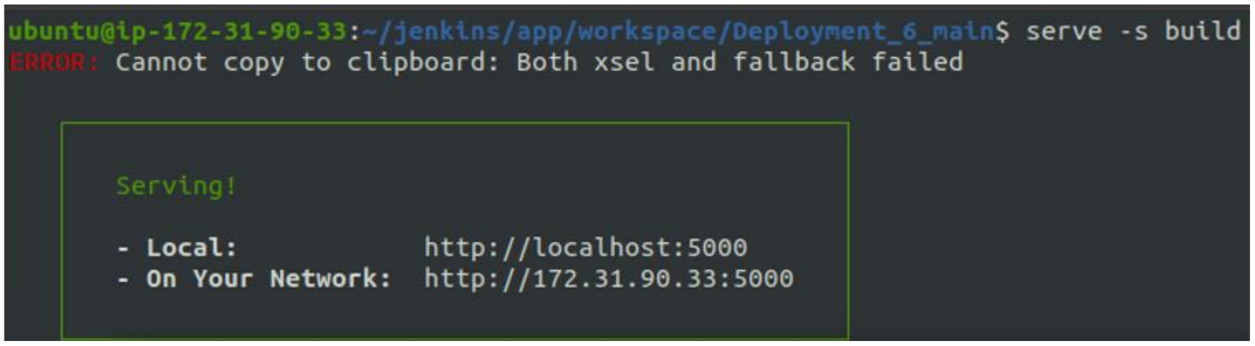
Passphrase

40. Once done, add and select the credentials you created.
41. Save all setting made to the Multipipeline and then build it.
42. After testing head to your terminal and ssh into your first agent.
43. Once inside your agent you can type “ls” to see what’s in there. Then head into your Jenkins directory.
44. From the Jenkins directory make your way to the Deployment_6_main directory using the following pathing “cd ./jenkins/app/workspace/Deployment_6_main” as shown below:



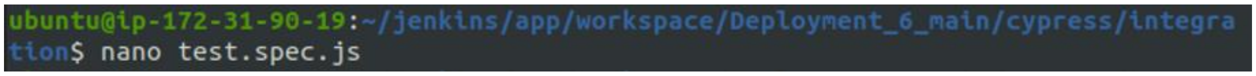
```
ubuntu@ip-172-31-90-33:~$ cd jenkins/app
ubuntu@ip-172-31-90-33:~/jenkins/app$ cd workspace/Deployment_6_ma
ubuntu@ip-172-31-90-33:~/jenkins/app/workspace/Deployment_6_main$
```

45. Once in the Deployment_6_main directory, use the command “serve -s build”. The result should look as shown:



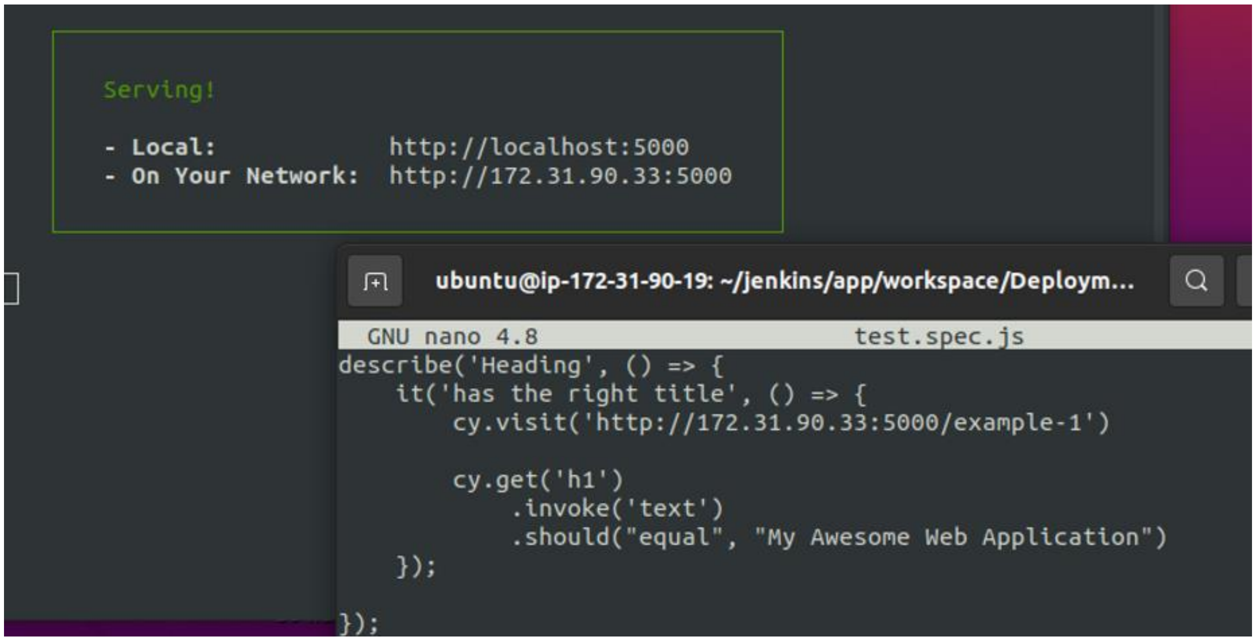
```
ubuntu@ip-172-31-90-33:~/jenkins/app/workspace/Deployment_6_main$ serve -s build
ERROR: Cannot copy to clipboard: Both xsel and fallback failed

    Serving!

    - Local:            http://localhost:5000
    - On Your Network:  http://172.31.90.33:5000
```

46. Once done, open a new terminal and ssh into your second agent.
47. Then “ls” to ensure that the dependencies are there.
48. Then change directory using “ cd ./jenkins/app/workspace/Deployment_6_main/cypress/integration” and nano into test.specs.js to edit it.



```
ubuntu@ip-172-31-90-19:~/jenkins/app/workspace/Deployment_6_main/cypress/integra
tion$ nano test.spec.js
```

49. Once inside, ensure that the https link matches your network link in steps 45 as shown below:

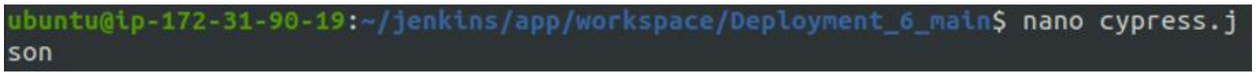


```
    Serving!

    - Local:            http://localhost:5000
    - On Your Network:  http://172.31.90.33:5000
```

ubuntu@ip-172-31-90-19: ~/jenkins/app/workspace/Deploym...

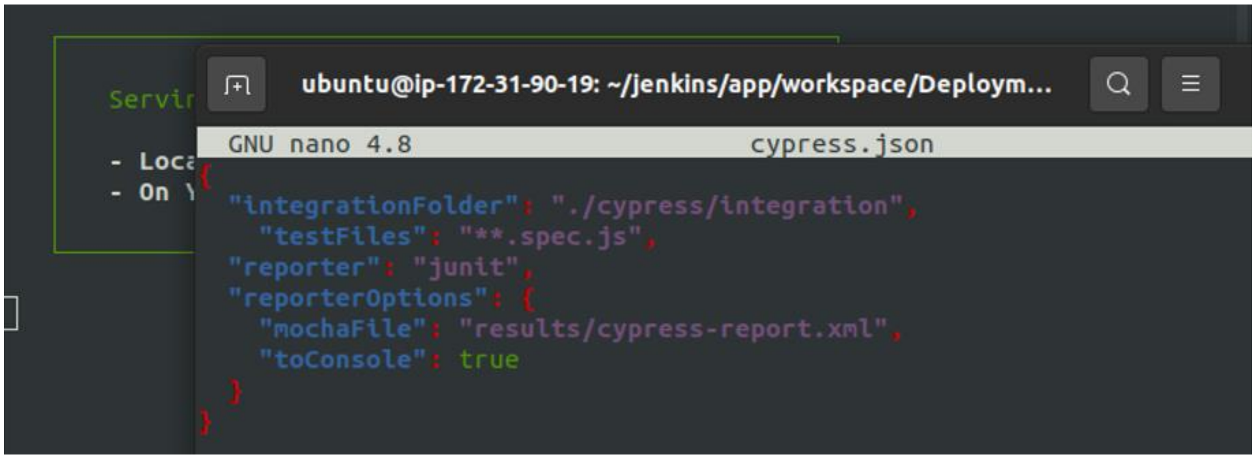
```
  GNU nano 4.8                          test.spec.js
describe('Heading', () => {
    it('has the right title', () => {
        cy.visit('http://172.31.90.33:5000/example-1')

        cy.get('h1')
            .invoke('text')
            .should("equal", "My Awesome Web Application")
    });

});
```

50. Save changes, then nano into your cypress.son located in the Deployment_6_main directory.



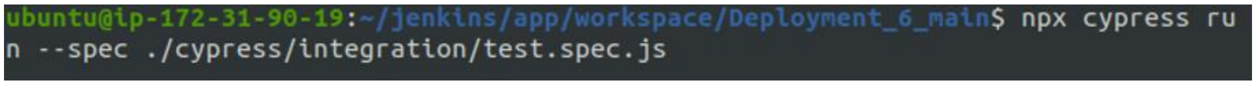
```
ubuntu@ip-172-31-90-19:~/jenkins/app/workspace/Deployment_6_main$ nano cypress.j
son
```

51. Once inside your cypress.json add this to your json:
 "integrationFolder": "./cypress/integration",
 "testFiles": "**.spec.js",



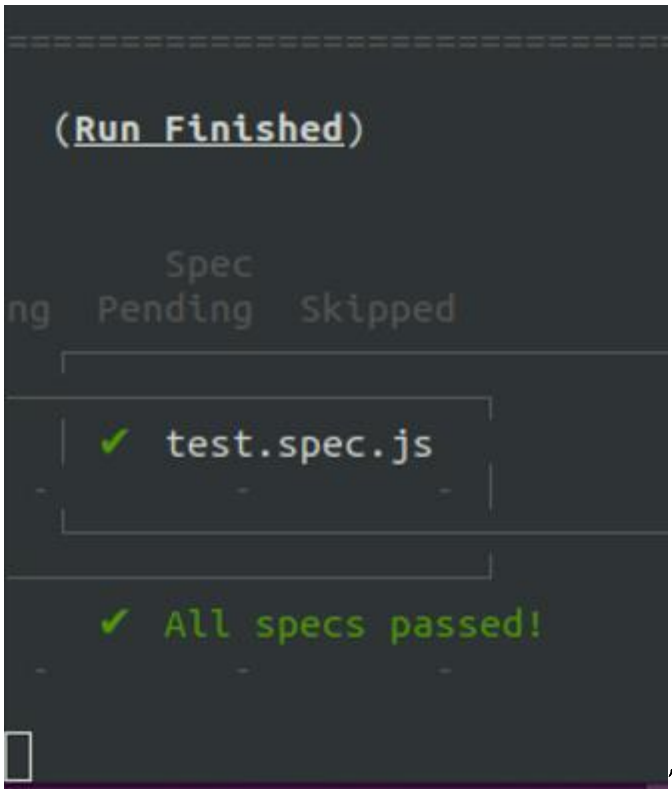
```
GNU nano 4.8                          cypress.json
{
 "integrationFolder": "./cypress/integration",
   "testFiles": "**.spec.js",
 "reporter": "junit",
 "reporterOptions": {
   "mochaFile": "results/cypress-report.xml",
   "toConsole": true
  }
}
```

52. Once done use the following command:



```
ubuntu@ip-172-31-90-19:~/jenkins/app/workspace/Deployment_6_main$ npx cypress ru
n --spec ./cypress/integration/test.spec.js
```

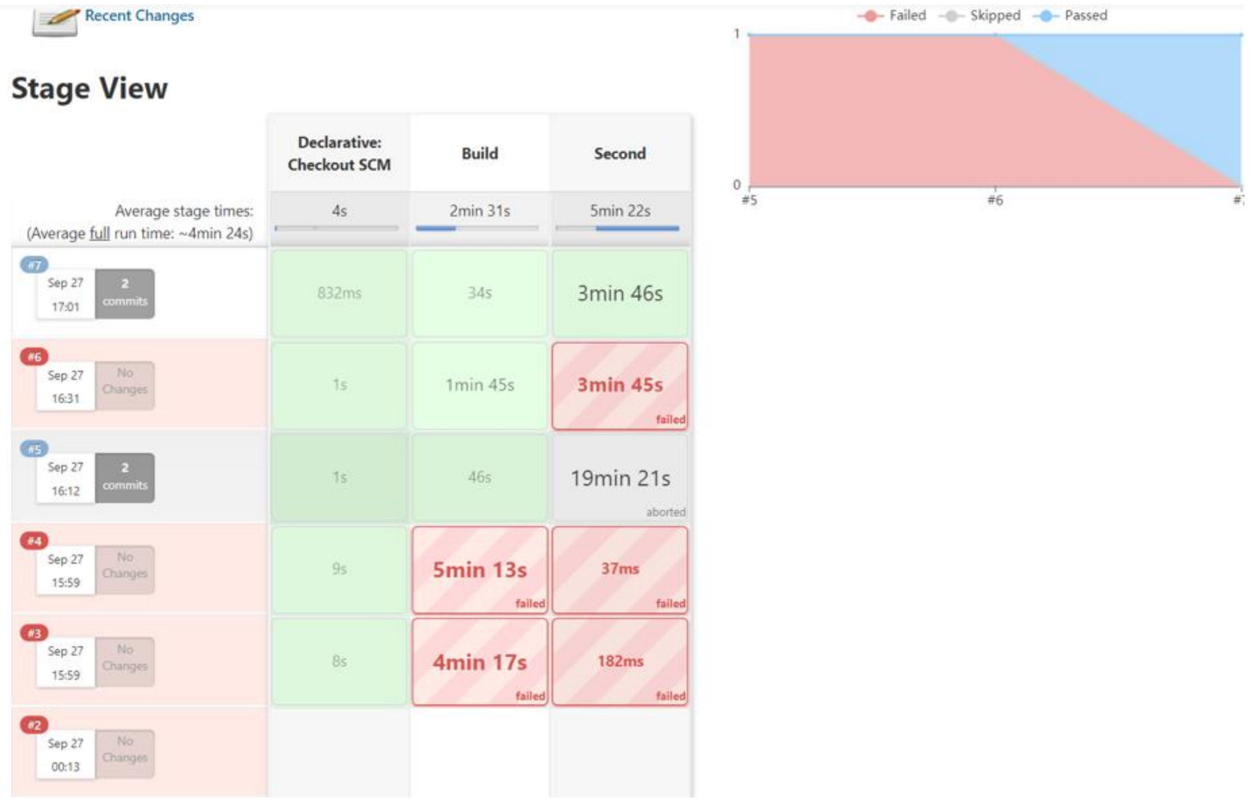
Once successful, the results will look like this:



```
(Run Finished)

      Spec
ng  Pending  Skipped

  ✔  test.spec.js

  ✔  All specs passed!
```

,

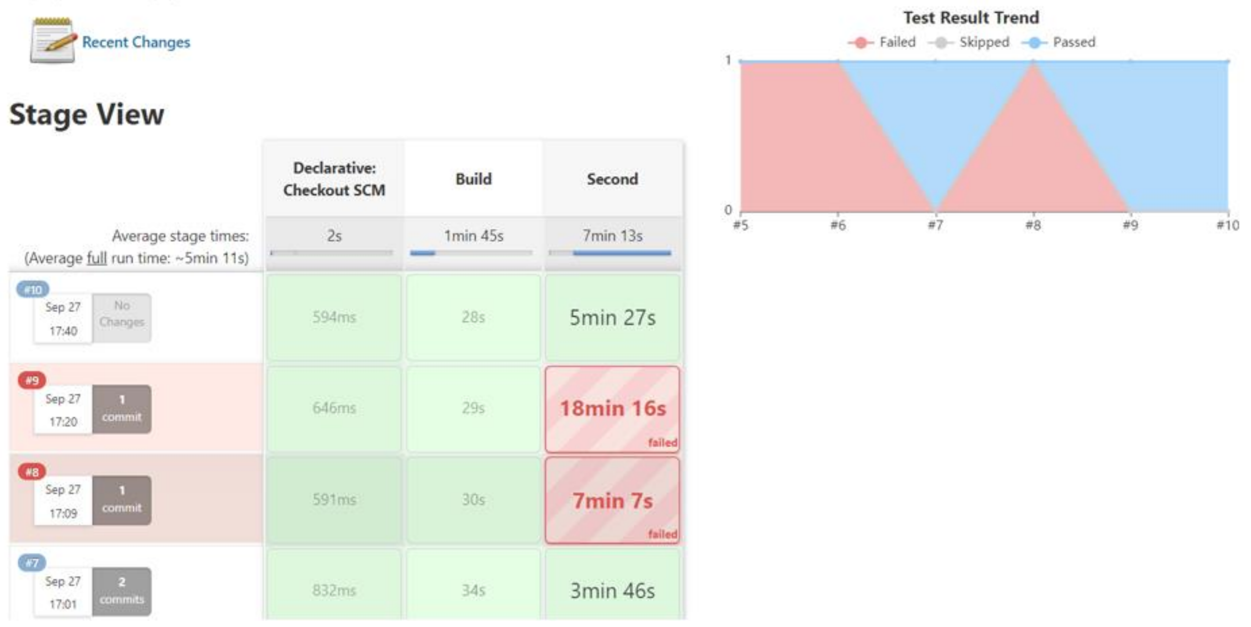
53. Then go to Jenkins and test your build.



Stage View

	Declarative: Checkout SCM	Build	Second
Average stage times: (Average full run time: ~4min 24s)	4s	2min 31s	5min 22s
#7 Sep 27 17:01 2 commits	832ms	34s	3min 46s
#6 Sep 27 16:31 No Changes	1s	1min 45s	3min 45s failed
#5 Sep 27 16:12 2 commits	1s	46s	19min 21s aborted
#4 Sep 27 15:59 No Changes	9s	5min 13s failed	37ms failed
#3 Sep 27 15:59 No Changes	8s	4min 17s failed	182ms failed
#2 Sep 27 00:13 No Changes			

54. Then break it to ensure it fails.

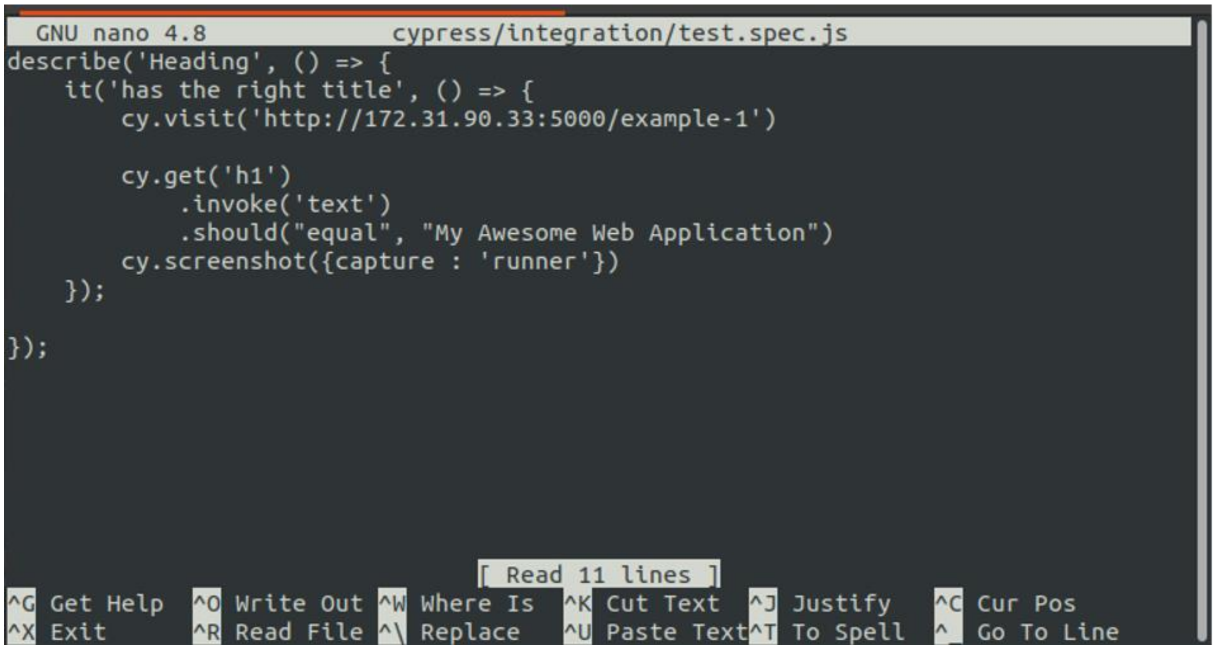


Stage View

	Declarative: Checkout SCM	Build	Second
Average stage times: (Average full run time: ~5min 11s)	2s	1min 45s	7min 13s
#10 Sep 27 17:40 No Changes	594ms	28s	5min 27s
#9 Sep 27 17:20 1 commit	646ms	29s	18min 16s failed
#8 Sep 27 17:09 1 commit	591ms	30s	7min 7s failed
#7 Sep 27 17:01 2 commits	832ms	34s	3min 46s

NOTE: after a successful failure in test 8, my application was successfully built in test 9, however a timeout occurred resulting in a failed result in the end.

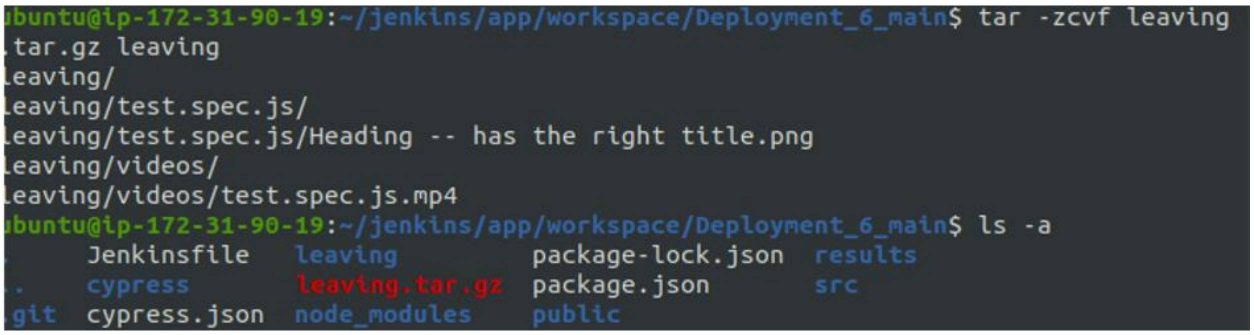
55. Then enter into your test.specs.js and include the line “cy.screenshot({capture : ‘runner’}) as shown below:



```
  GNU nano 4.8              cypress/integration/test.spec.js
describe('Heading', () => {
    it('has the right title', () => {
        cy.visit('http://172.31.90.33:5000/example-1')

        cy.get('h1')
            .invoke('text')
            .should("equal", "My Awesome Web Application")
        cy.screenshot({capture : 'runner'})
    });

});

                                 [ Read 11 lines ]
^G Get Help   ^O Write Out  ^W Where Is   ^K Cut Text   ^J Justify    ^C Cur Pos
^X Exit       ^R Read File  ^\ Replace    ^U Paste Text ^T To Spell   ^_ Go To Line
```

NOTE: I tried “cy.screenshot()” and “cy.screenshot({capture : ‘fullPage’}) but neither worked for getting a screenshot of the full page.

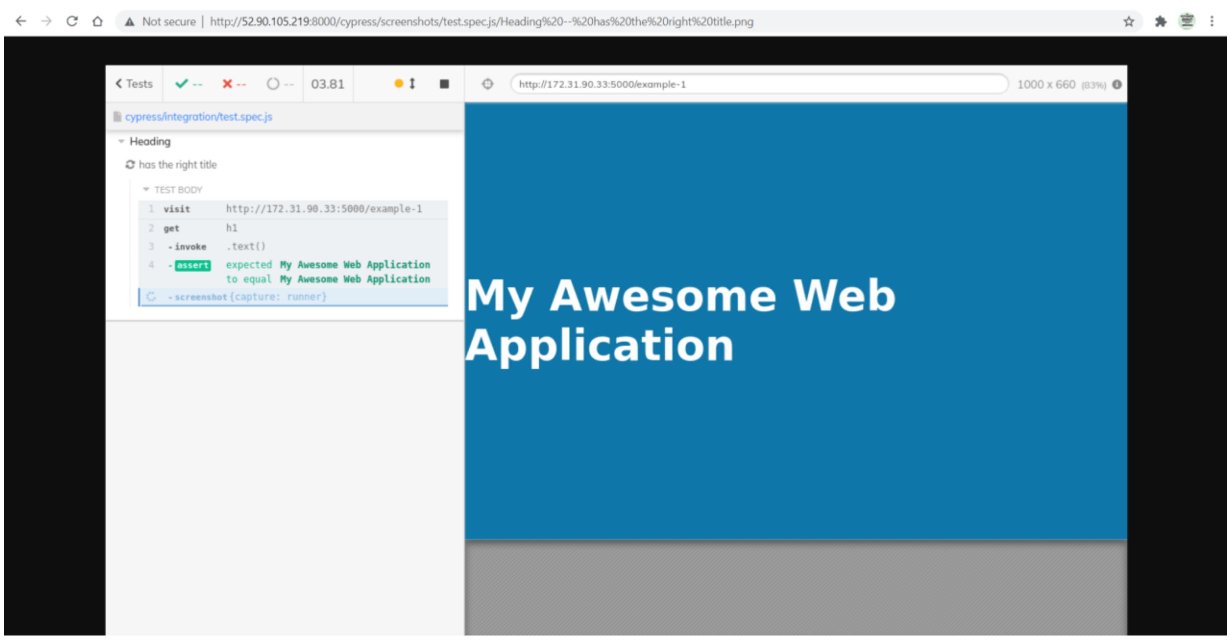
56. Ensure that your first agent is still running, if not restart with steps from 44 and 55. In your terminal running your second agent, enter into the Deployment_6_main directory.
57. Once there use the command “python3 -m http.server”.
58. Once there use the command “cp -R cypress/screenshots” and give it a name. This will create a copy of the screenshots and place them in a file with the new name given. Example “cp -R cypress/screenshots leaving”
59. Then check to see if a copy was made using the command “ls -a leaving/test.specs.js”.
60. Once there use the command “cp -R cypress/videos” and give it a name. This will create a copy of the videos and place them in a file with the new name given. Example “cp -R cypress/videos leaving”
61. Then check to see if a copy was made using the command “ls -a leaving/videos”.
62. Then use the command “tar -zcvf *filename*” in my case “tar -zcvf leaving” to zip the contents of the file as shown below:



```
buntu@ip-172-31-90-19:~/jenkins/app/workspace/Deployment_6_main$ tar -zcvf leaving
.tar.gz leaving
leaving/
leaving/test.spec.js/
leaving/test.spec.js/Heading -- has the right title.png
leaving/videos/
leaving/videos/test.spec.js.mp4
buntu@ip-172-31-90-19:~/jenkins/app/workspace/Deployment_6_main$ ls -a
      Jenkinsfile   leaving          package-lock.json  results
.     cypress       leaving.tar.gz   package.json       src
git   cypress.json  node_modules     public
```

63. Then exit out of the agent and go to your home directory. From there enter into your Downloads directory and use “ls” to see if the components were downloaded.
64. Once successful move them to your home directory using the command “mv leaving.tar.gz ../”.
65. Then use the command “tar -xzvf leaving.tar.gz” to extract the components of that file.

```
~$ tar -xzvf leaving.tar.gz
```



Not secure | http://52.90.105.219:8000/cypress/screenshots/test.spec.js/Heading%20--%20has%20the%20right%20title.png

Tests ✓ -- ✗ -- ○ -- 03.81

http://172.31.90.33:5000/example-1 1000 x 660 (83%)

cypress/integration/test.spec.js

Heading

has the right title

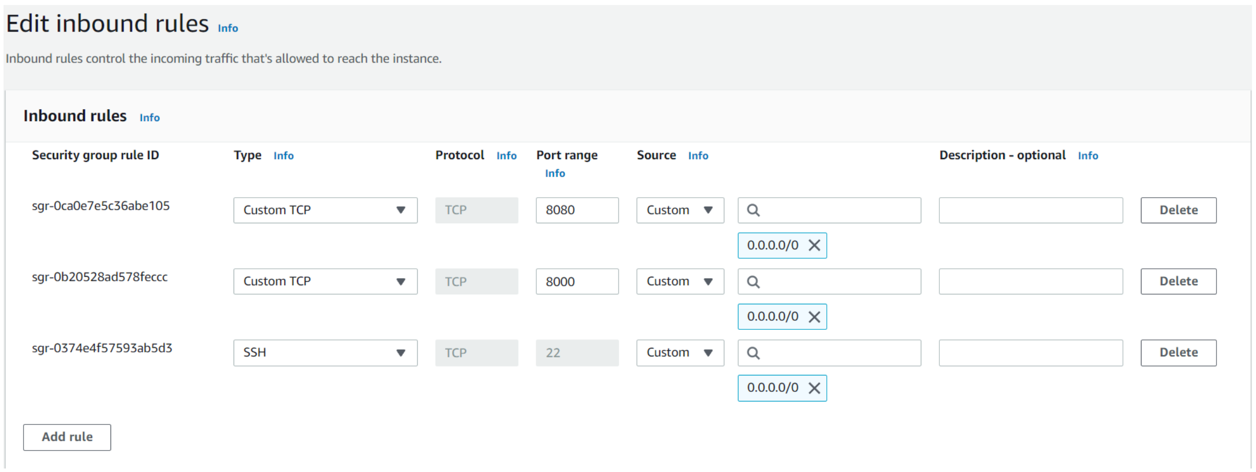
TEST BODY

```
visit       http://172.31.90.33:5000/example-1
get         h1
 - invoke   .text()
 - assert   expected My Awesome Web Application
            to equal My Awesome Web Application
 - screenshot {capture: runner}
```

My Awesome Web Application

Screenshot of application taken

Changes made to Jenkins Master Security group.



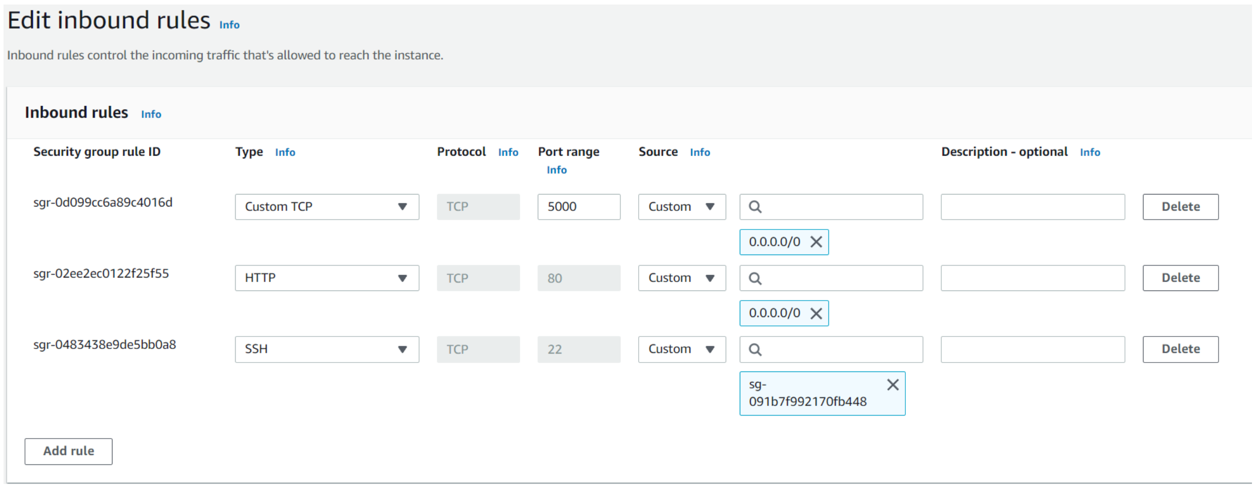
Edit inbound rules Info

Inbound rules control the incoming traffic that's allowed to reach the instance.

Inbound rules Info

Security group rule ID	Type Info	Protocol Info	Port range Info	Source Info		Description - optional Info	
sgr-0ca0e7e5c36abe105	Custom TCP	TCP	8080	Custom	0.0.0.0/0		Delete
sgr-0b20528ad578feccc	Custom TCP	TCP	8000	Custom	0.0.0.0/0		Delete
sgr-0374e4f57593ab5d3	SSH	TCP	22	Custom	0.0.0.0/0		Delete

Changes made to Agent 1 Security group



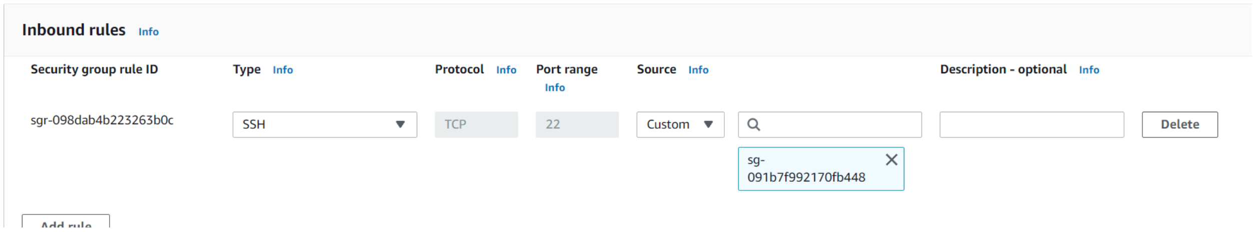
Edit inbound rules

Inbound rules control the incoming traffic that's allowed to reach the instance.

Inbound rules

Security group rule ID	Type	Protocol	Port range	Source		Description - optional
sgr-0d099cc6a89c4016d	Custom TCP	TCP	5000	Custom	0.0.0.0/0	
sgr-02ee2ec0122f25f55	HTTP	TCP	80	Custom	0.0.0.0/0	
sgr-0483438e9de5bb0a8	SSH	TCP	22	Custom	sg-091b7f992170fb448	

Agent 2 Security group



Inbound rules

Security group rule ID	Type	Protocol	Port range	Source		Description - optional
sgr-098dab4b223263b0c	SSH	TCP	22	Custom	sg-091b7f992170fb448	