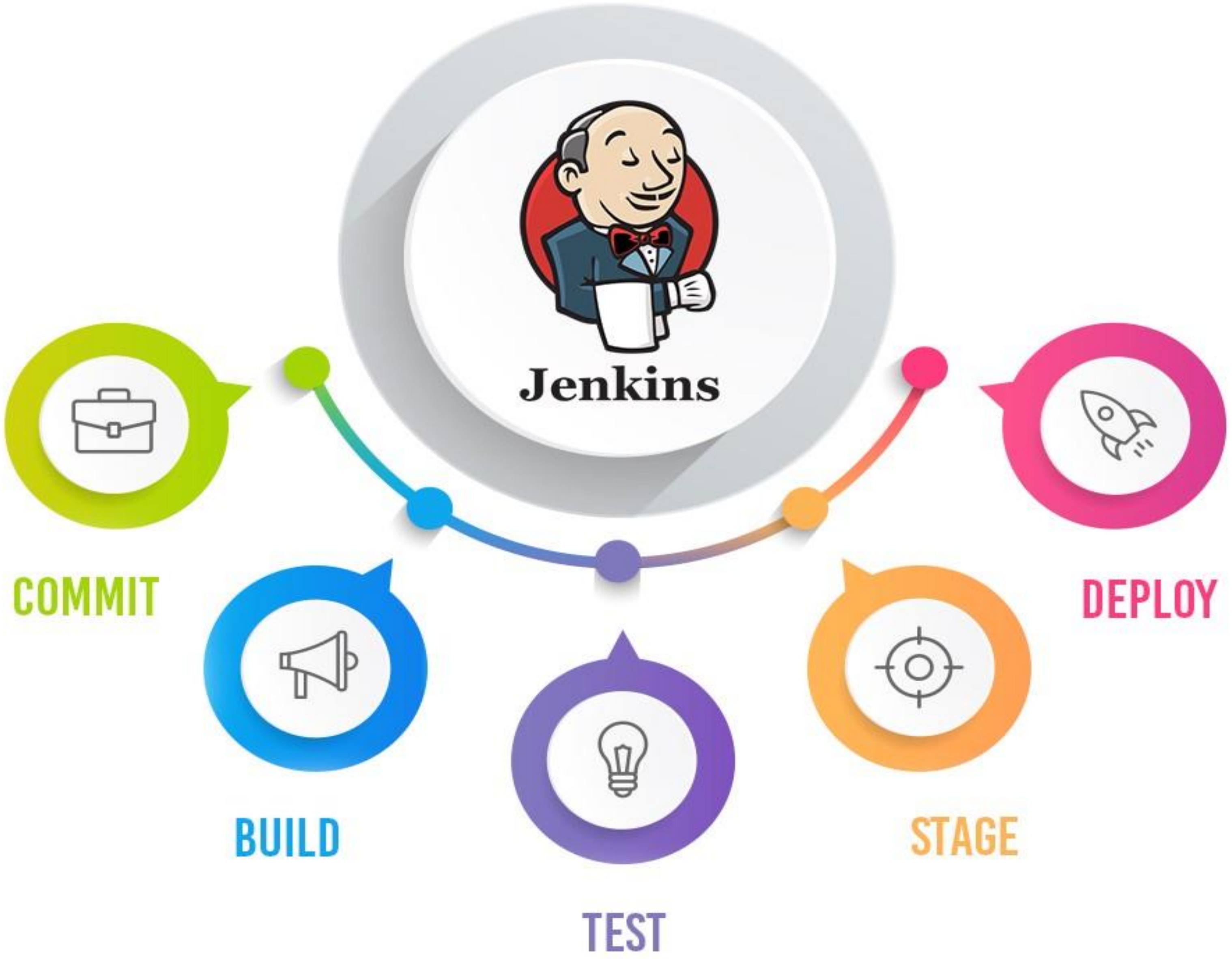
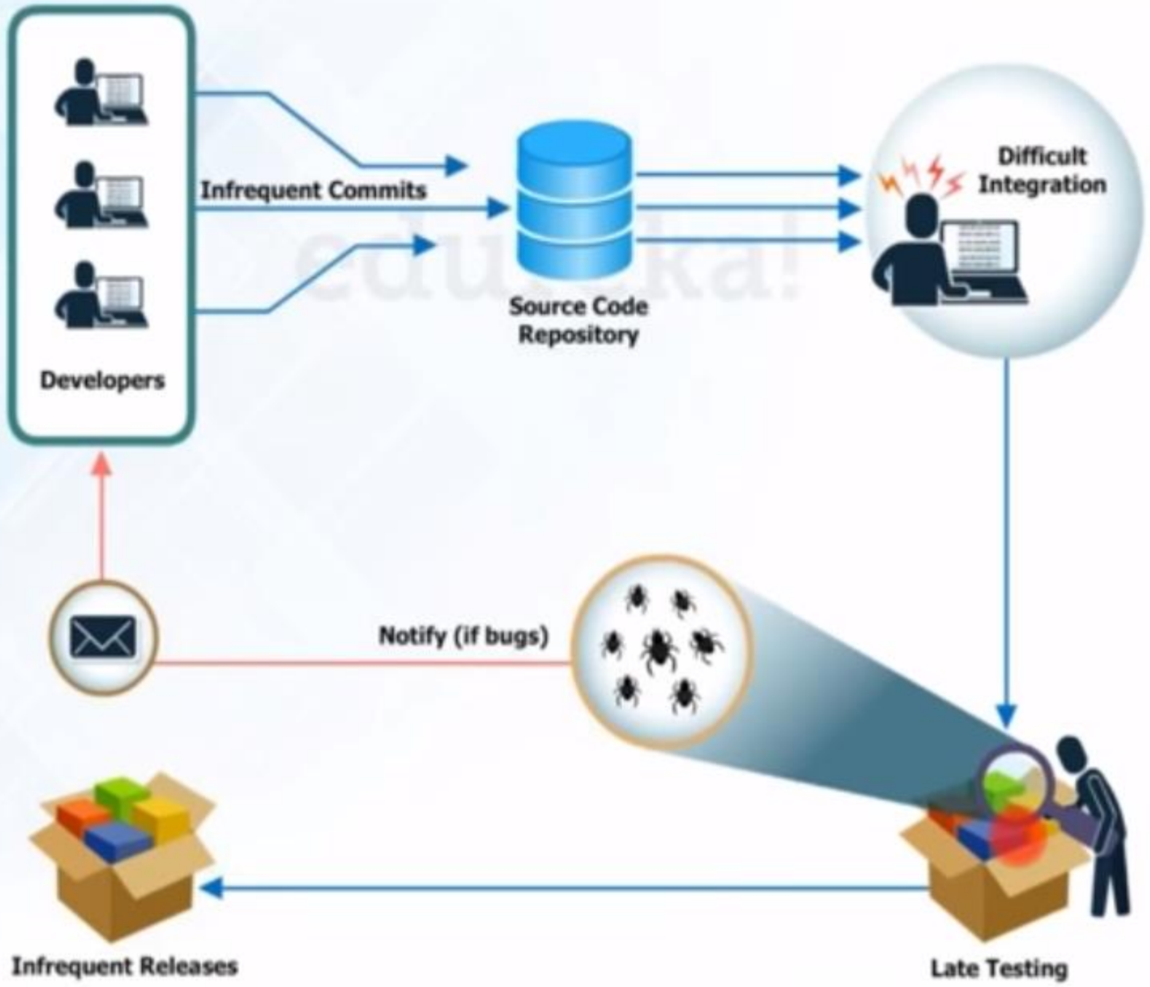
Jenkins


Jenkins
COMMIT
BUILD
TEST
STAGE
DEPLOY

Jenkins

- Jenkins is an open source automation tool written in Java programming language that allows continuous integration.
- Jenkins **builds** and **tests** our software projects which continuously making it easier for developers to integrate changes to the project, and making it easier for users to obtain a fresh build.
- It also allows us to continuously **deliver** our software by integrating with a large number of testing and deployment technologies.
- Jenkins offers a straightforward way to set up a continuous integration or continuous delivery environment for almost any combination of languages and source code repositories using pipelines, as well as automating other routine development tasks.

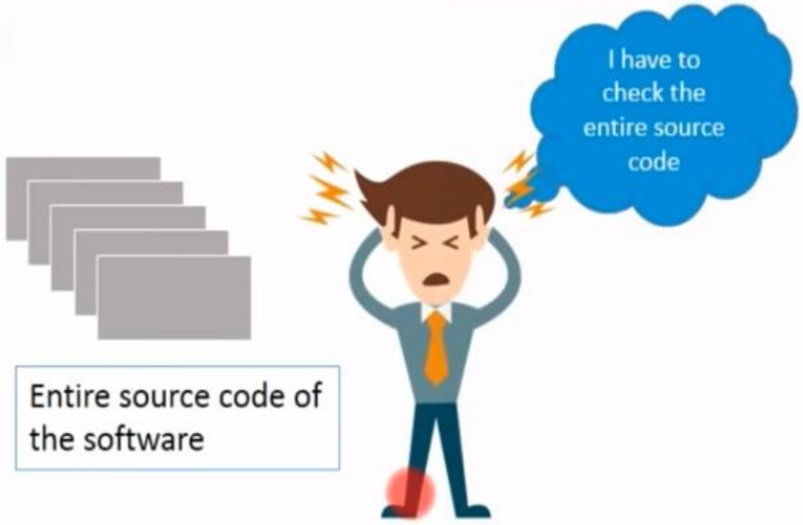
Process Before Continuous Integration



Problems Before Continuous Integration

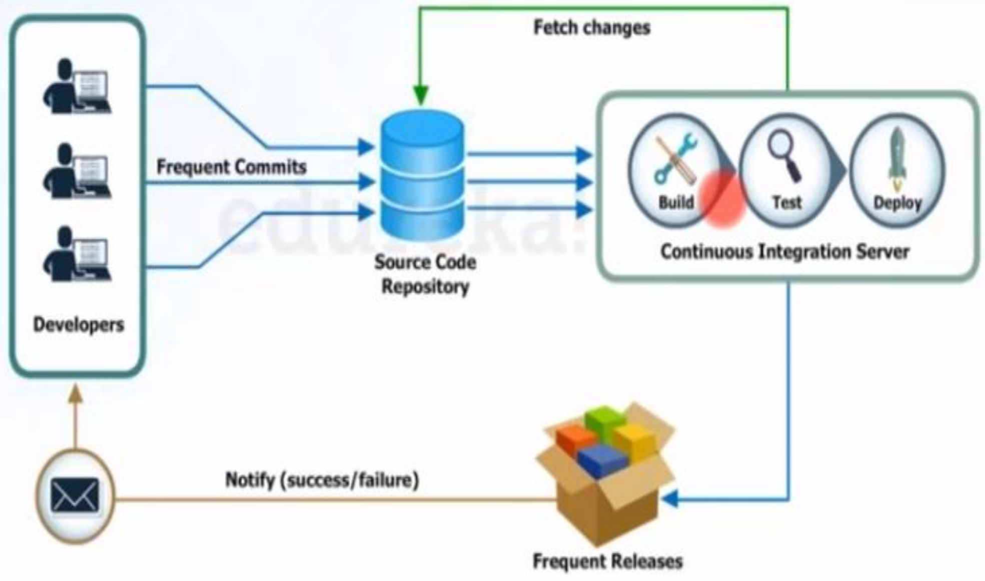
Developers have to wait till the complete software is developed for the test results.

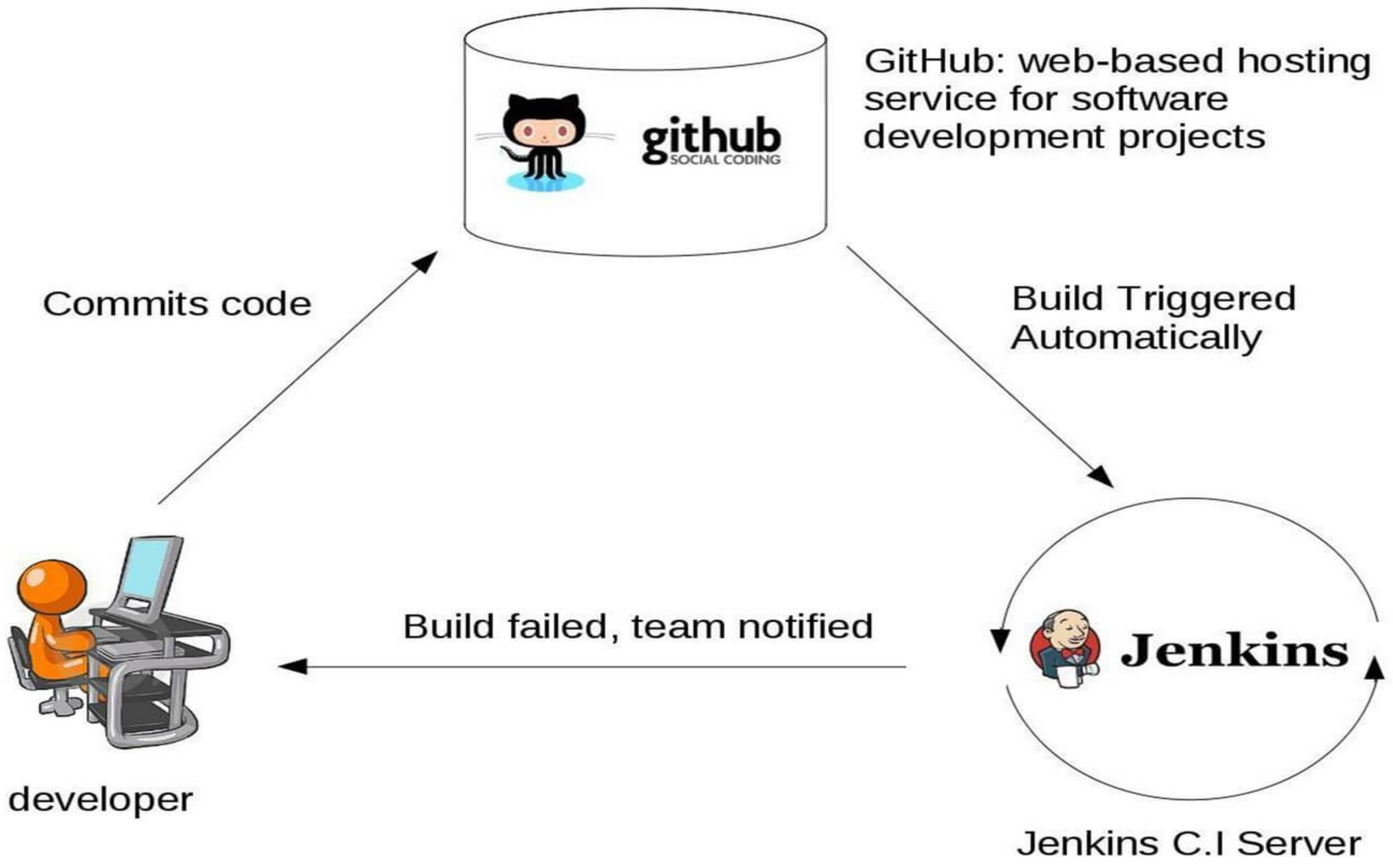
If the test fails then locating and fixing bugs is very difficult. Developers have to check the entire source code of the software.

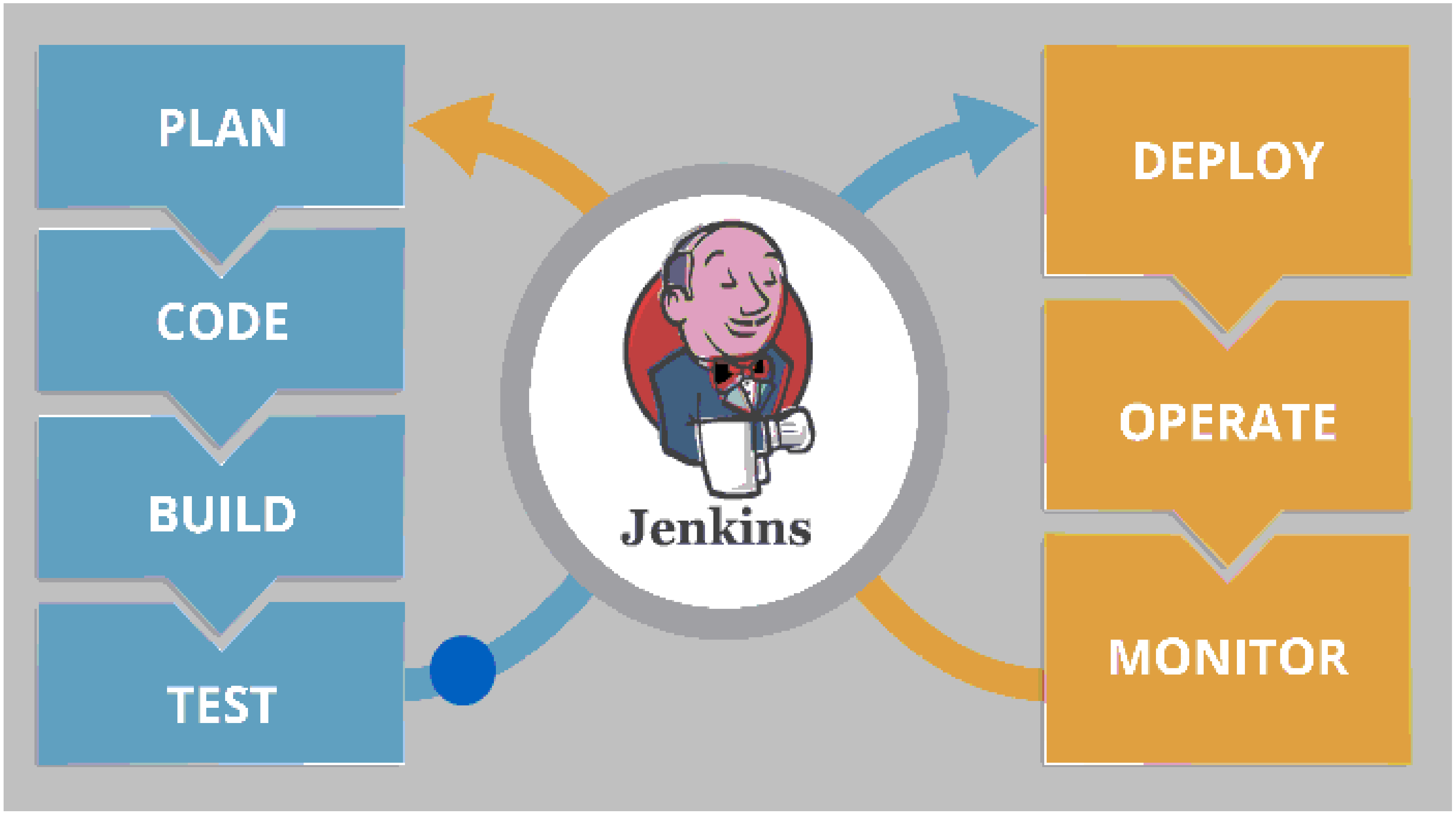

Entire source code of the software
I have to check the entire source code

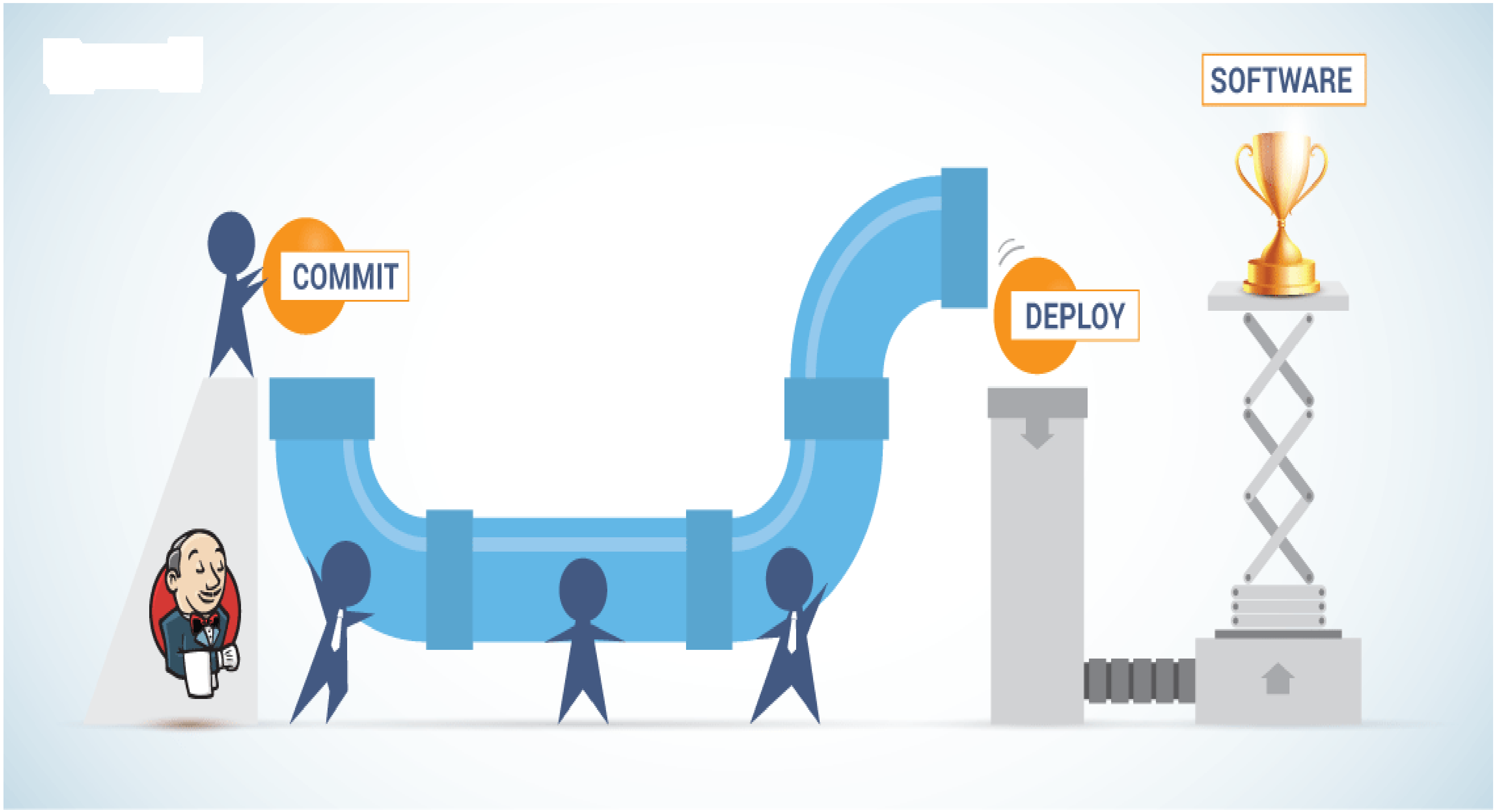
Continuous Integration To The Rescue

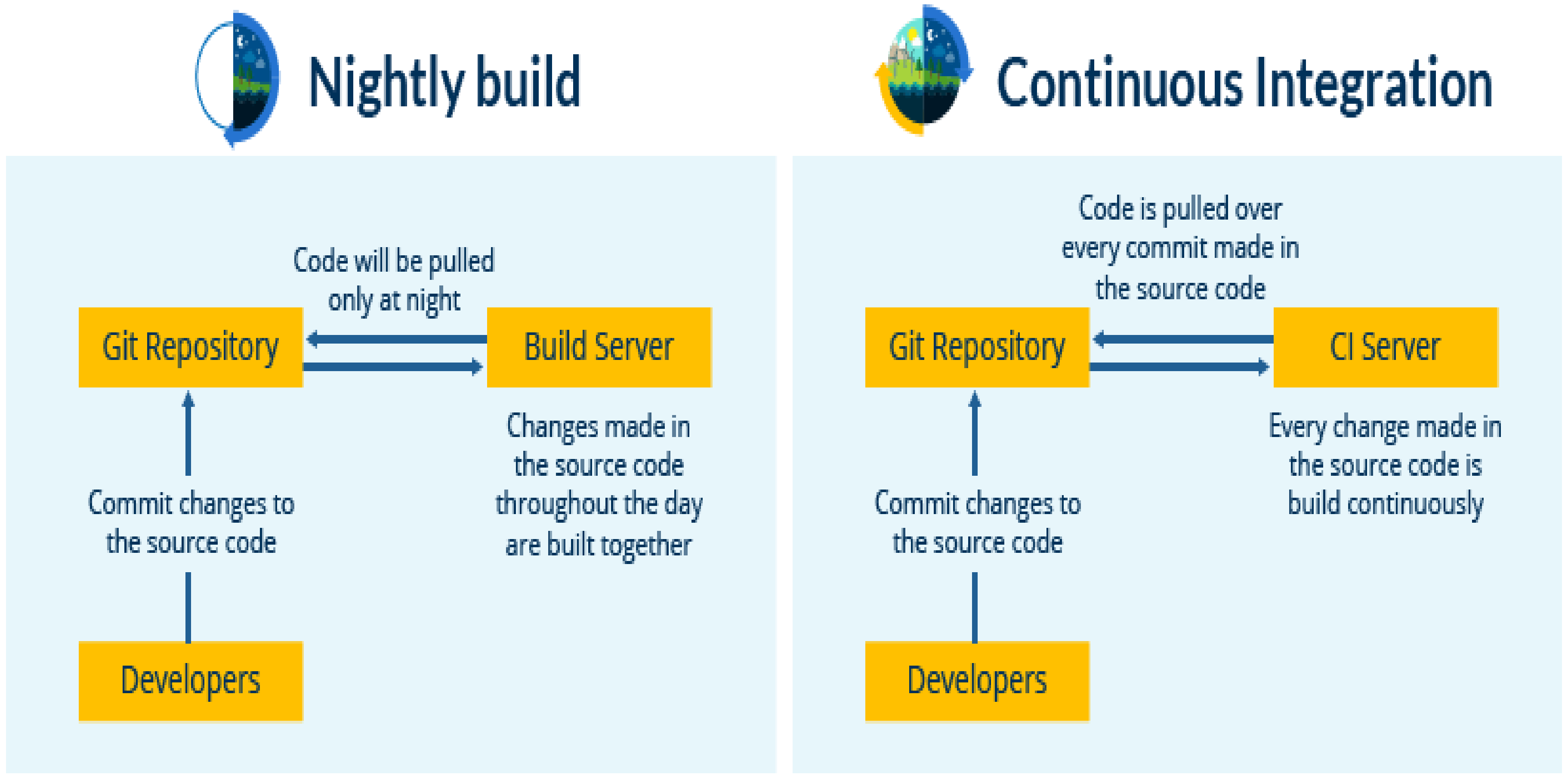
- ❑ Since after every commit to the source code an auto build is triggered and then it is automatically deployed on the test server
- ❑ If the test results shows that there is a bug in the code then the developers only have to check the last commit made to the source code
- ❑ This also increases the frequency of new software releases
- ❑ The concerned teams are always provided with the relevant feedback

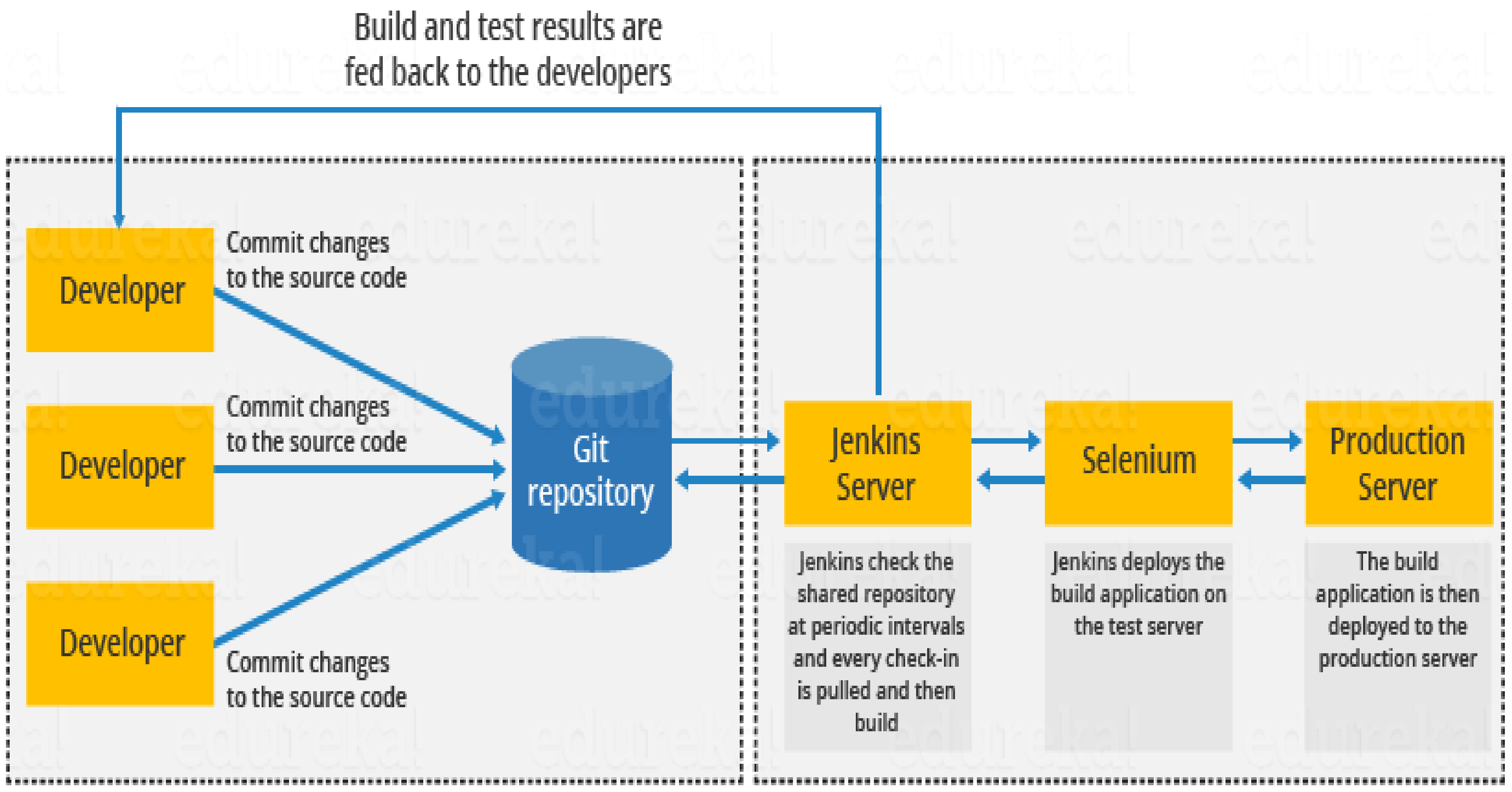



github
SOCIAL CODING
GitHub: web-based hosting service for software development projects
Commits code
Build Triggered Automatically
Build failed, team notified
Jenkins
developer
Jenkins C.I Server


PLAN
CODE
BUILD
TEST
Jenkins
DEPLOY
OPERATE
MONITOR


SOFTWARE
COMMIT
DEPLOY


Nightly build
Code will be pulled only at night
Git Repository
Build Server
Commit changes to the source code
Changes made in the source code throughout the day are built together
Developers
Continuous Integration
Code is pulled over every commit made in the source code
Git Repository
CI Server
Commit changes to the source code
Every change made in the source code is build continuously
Developers


Build and test results are fed back to the developers
Developer
Commit changes to the source code
Developer
Commit changes to the source code
Developer
Commit changes to the source code
Git repository
Jenkins Server
Selenium
Production Server
Jenkins check the shared repository at periodic intervals and every check-in is pulled and then build
Jenkins deploys the build application on the test server
The build application is then deployed to the production server

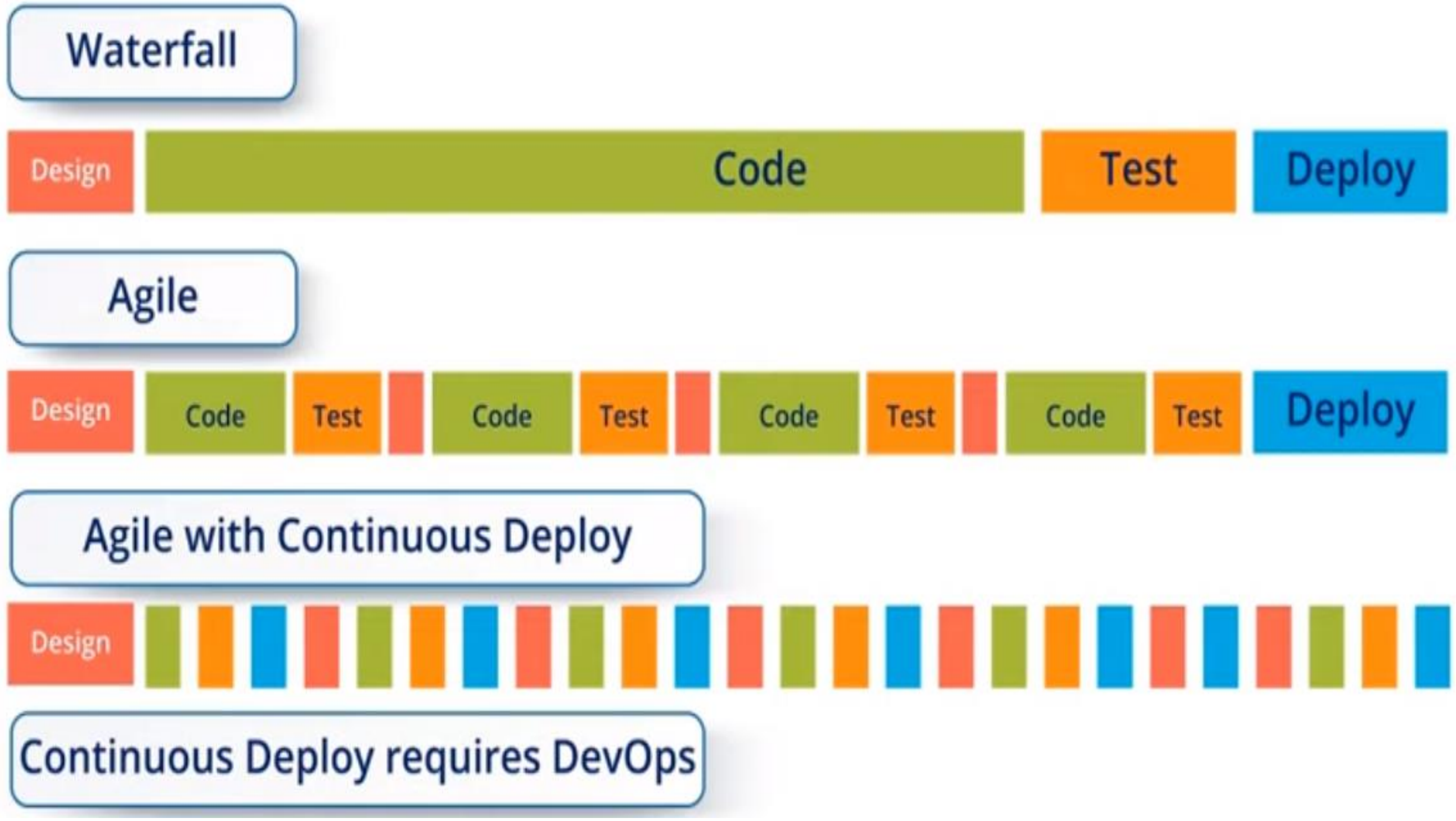
Before Jenkins	After Jenkins
Once all Developers had completed their assigned coding tasks, they used to commit their code all at same time. Later, Build is tested and deployed	The code is built and test as soon as Developer commits code. Jenkin will build and test code many times during the day
Code commit built, and test cycle was very infrequent, and a single build was done after many days.	If the build is successful, then Jenkins will deploy the source into the test server and notifies the deployment team.
Since the code was built all at once, some developers would need to wait until other developers finish coding to check their build	The code is built immediately after any of the Developer commits
It is not an easy task to isolate, detect, and fix errors for multiple commits.	Since the code is built after each commit of a single developer, it's easy to detect whose code caused the built to fail
Code build and test process are entirely manual, so there are a lot of chances for failure.	Automated build and test process saving timing and reducing defects.
The code is deployed once all the errors are fixed and tested	The code is deployed after every successful build and test
Development Cycle is slow	The development cycle is fast. New features are more readily available to users. Increases profits

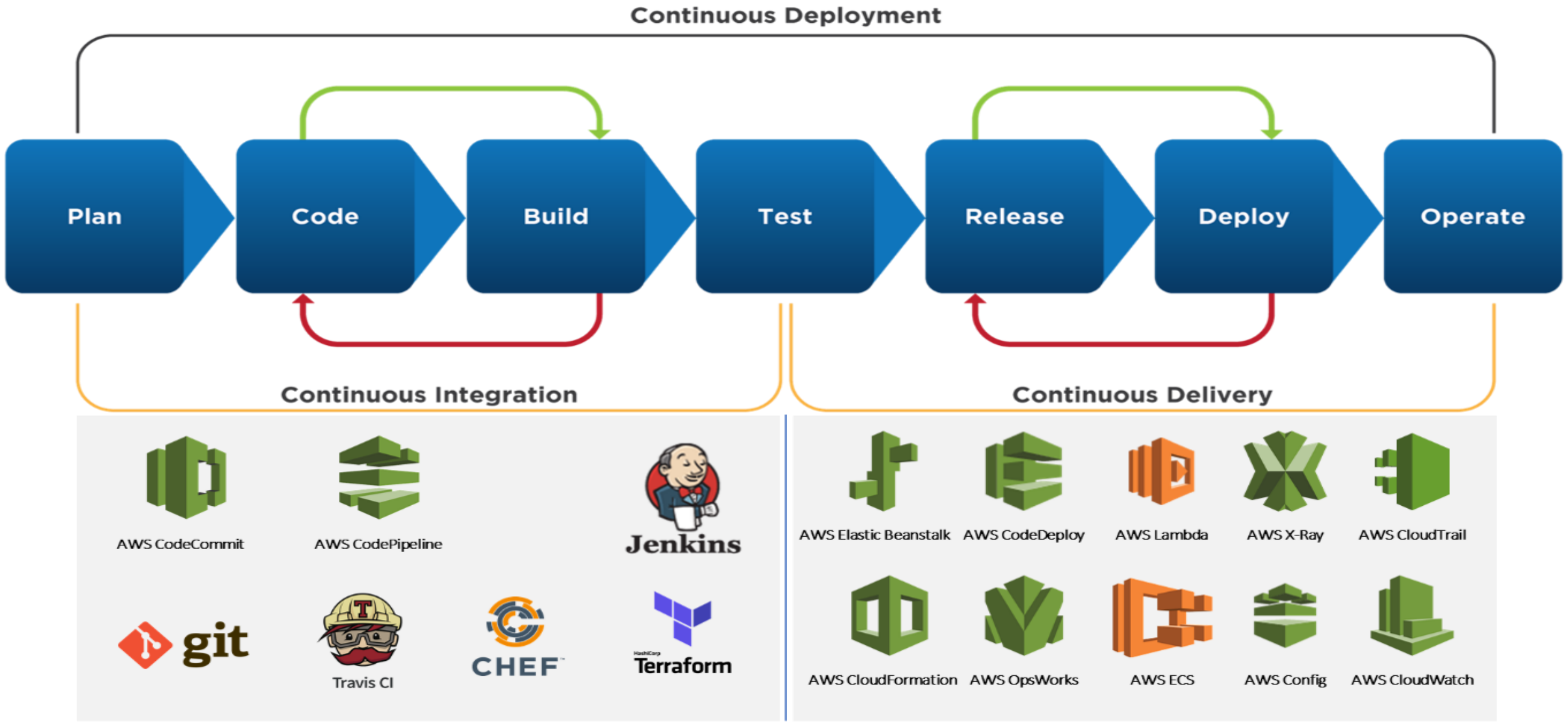
What is CI/CD?

- Continuous integration (CI) is a DevOps practice in which team members regularly commit their code changes to the version control repository, after which automated builds and tests are run.

- Continuous delivery (CD) is a series of practices where code changes are automatically built, tested and deployed to production.

Agile vs Waterfall vs DevOps

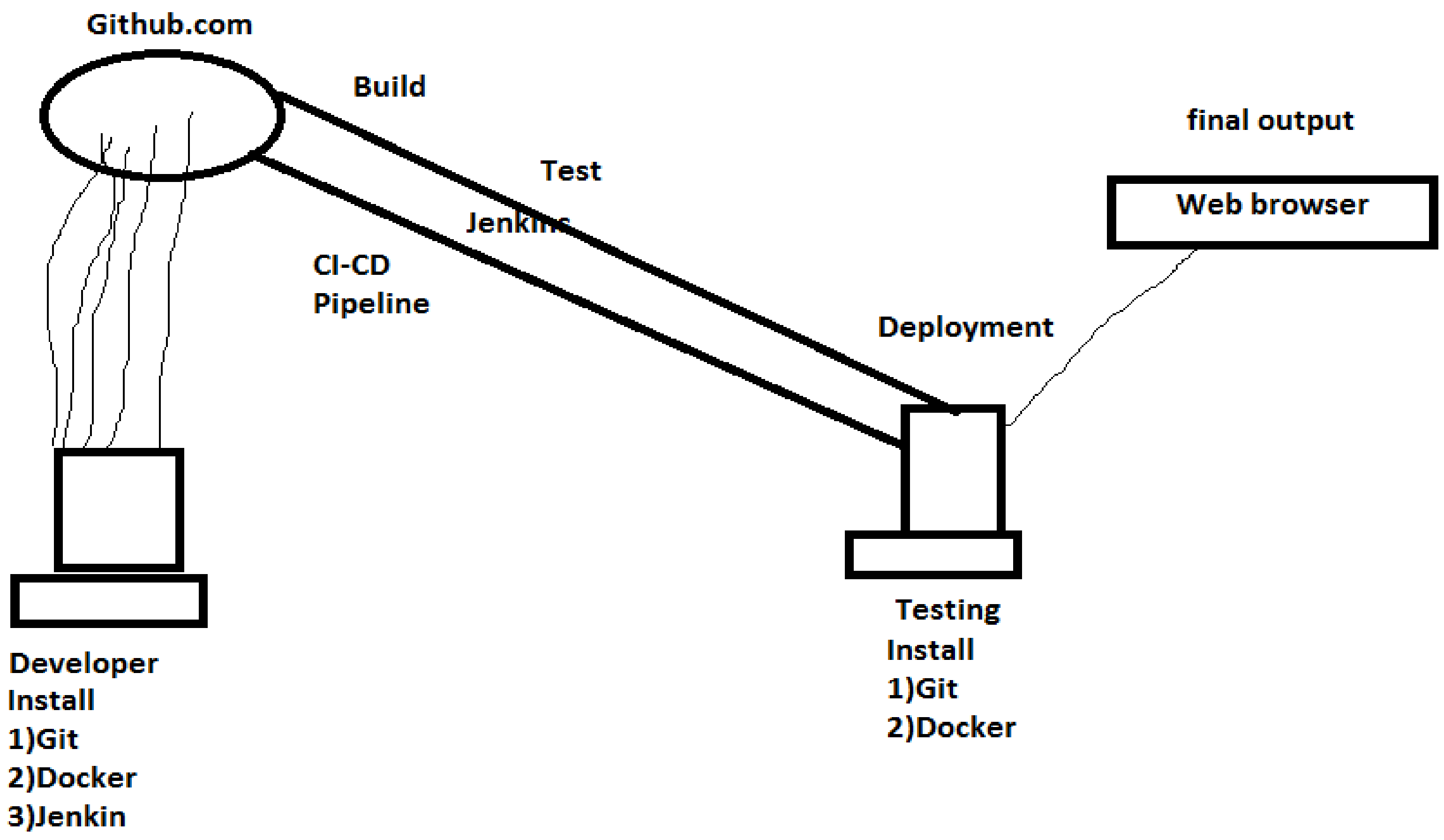



Continuous Deployment
Plan
Code
Build
Test
Release
Deploy
Operate
Continuous Integration
Continuous Delivery
AWS CodeCommit
AWS CodePipeline
Jenkins
git
Travis CI
CHEF
HashiCorp Terraform
AWS Elastic Beanstalk
AWS CodeDeploy
AWS Lambda
AWS X-Ray
AWS CloudTrail
AWS CloudFormation
AWS OpsWorks
AWS ECS
AWS Config
AWS CloudWatch

Advantages of Jenkins

- It is an open source tool.
- It is free of cost.
- It does not require additional installations or components. Means it is easy to install.
- Easily configurable.
- It supports 1000 or more plugins to ease your work. If a plugin does not exist, you can write the script for it and share with community.
- It is built in java and hence it is portable.
- It is platform independent. It is available for all platforms and different operating systems. Like OS X, Windows or Linux.
- Jenkins also supports cloud based architecture so that we can deploy Jenkins in cloud based platforms.

Lab


Github.com
Build
Test
Jenkins
CI-CD
Pipeline
Deployment
final output
Web browser
Developer
Install
1)Git
2)Docker
3)Jenkin
Testing
Install
1)Git
2)Docker

Lab : Step Involved

Steps

0) Req: 2 System(Developer, Testing)
1) Install jenkins
2)Jenkins setup(web console)
3)Global security
4) Adding Slave1(node1)
5) Maping node(agent.jar file)
6)Install git in both system
7) Create a job to map github to slave1
8) Create files in dev -push to github -jenkins-build now-check in slave1
9)Install docker in both system
10)Deploy image into slave
11)Configure auto trigger(webhook)
12)CI-CD pipelining
13) Do CI CD pipelining with 3 System(Dev, test, Prod)

Lab: Steps 0 and Step 1

Steps 0) Launch 2 Instance in AWScloud– Named as Developer and testing – Use security group port as all traffic.

Connect both Instances using Mobaxterm

Steps 1) Install Jenkins in Developer system

1) Jenkins is a Java application, so the first step is to install Java.

```
# yum install java-1.8.0-openjdk-devel
```

2) The next step is to enable the Jenkins repository.

```
# curl --silent --location http://pkg.jenkins-ci.org/redhat-stable/jenkins.repo | sudo tee /etc/yum.repos.d/jenkins.repo
# nano  /etc/yum.repos.d/jenkins.repo
 gpgcheck=0
```

3) Once the repository is enabled, install the latest stable version of Jenkins

```
# yum install jenkins   -y
```

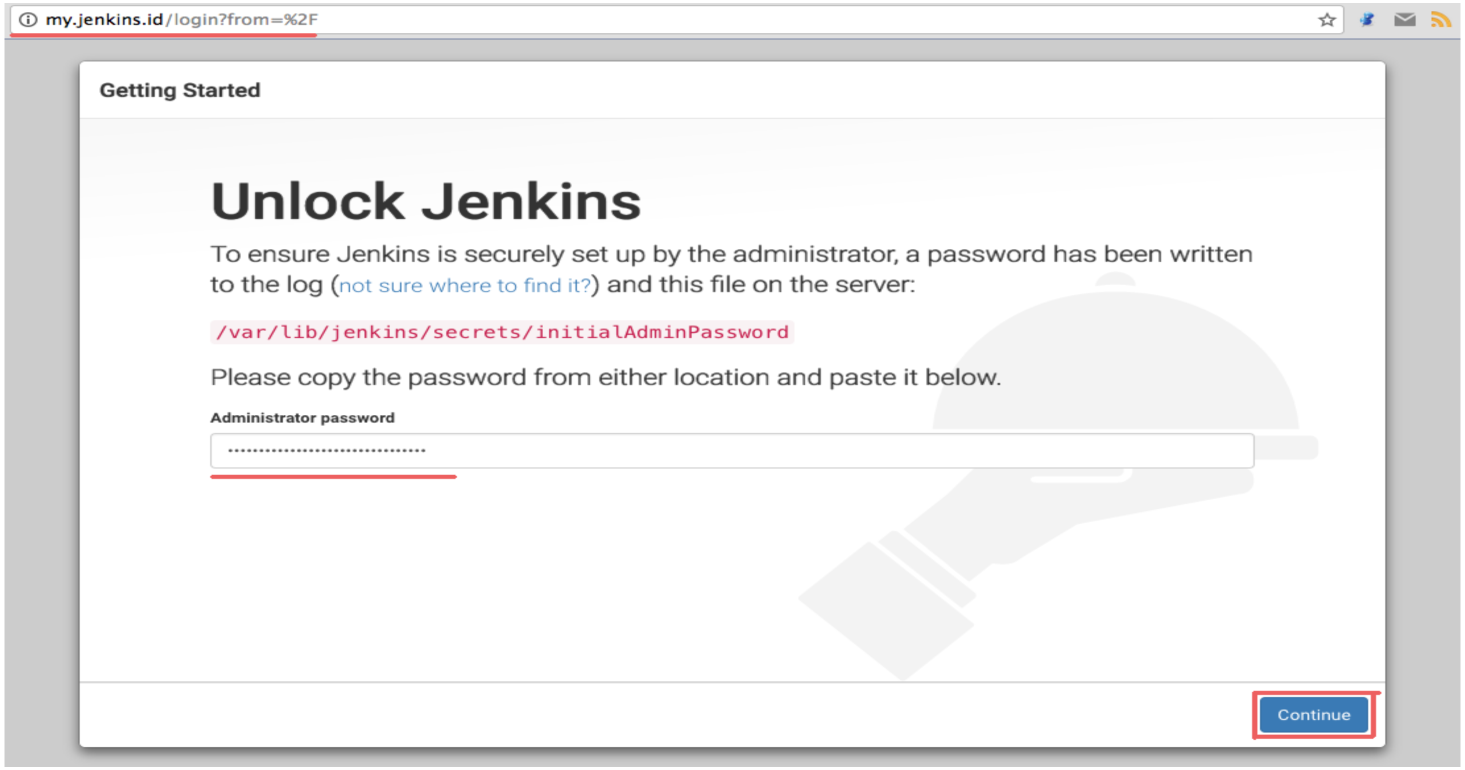
4) After the installation process is completed, start the Jenkins service

```
# systemctl   start   jenkins
# systemctl   enable jenkins
```

Lab: Step 2

Steps 2) Jenkins setup(web console) : To set up your new Jenkins installation, open your browser and type your domain or IP address followed by port 8080:

- http://your_ip_or_domain:8080
- Use the following command to print the password on your terminal:
- # cat /var/lib/jenkins/secrets/initialAdminPassword
- Copy the password from your terminal, paste it into the Administrator password field and click Continue.



my.jenkins.id/login?from=%2F

Getting Started

Unlock Jenkins

To ensure Jenkins is securely set up by the administrator, a password has been written to the log (not sure where to find it?) and this file on the server:

```
/var/lib/jenkins/secrets/initialAdminPassword
```

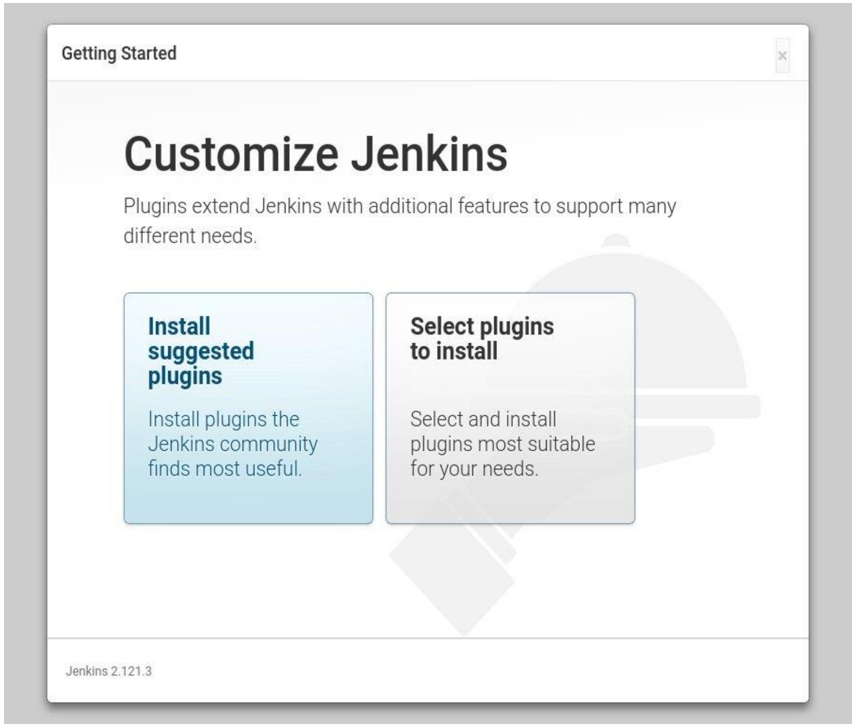
Please copy the password from either location and paste it below.

Administrator password

································

Lab: Step 2

On the next screen, you will be asked whether you want to install the suggested plugins or to select specific plugins. Click on the Install suggested plugins box, and the installation process will start immediately.



Getting Started

Customize Jenkins

Plugins extend Jenkins with additional features to support many different needs.

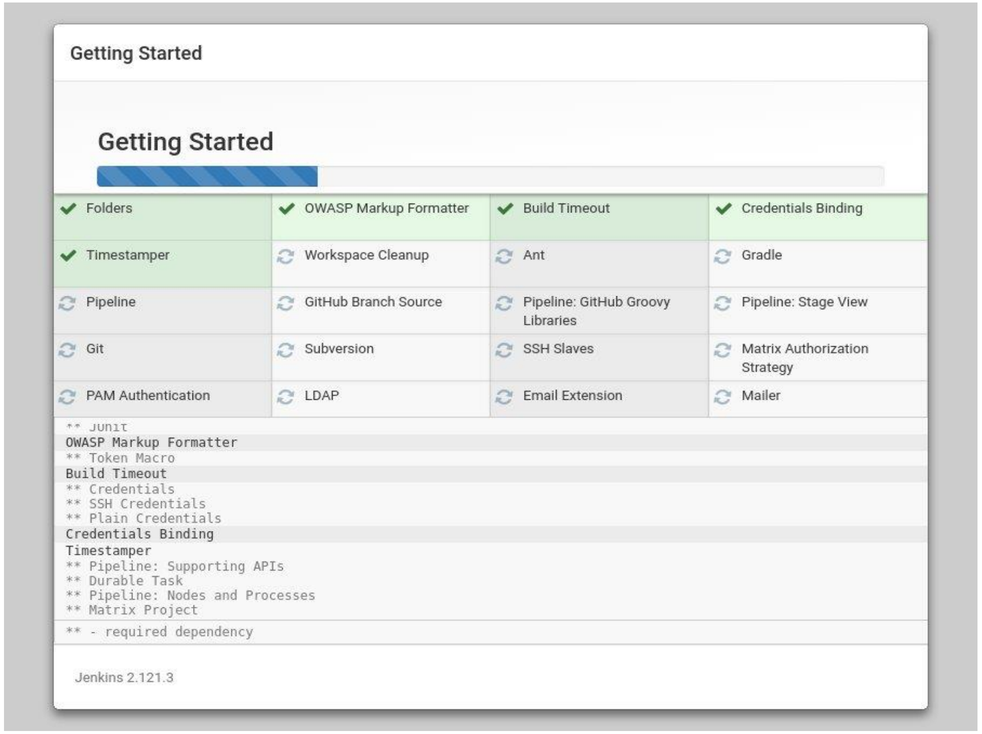
Install suggested plugins

Install plugins the Jenkins community finds most useful.

Select plugins to install

Select and install plugins most suitable for your needs.

Jenkins 2.121.3



Getting Started

Getting Started

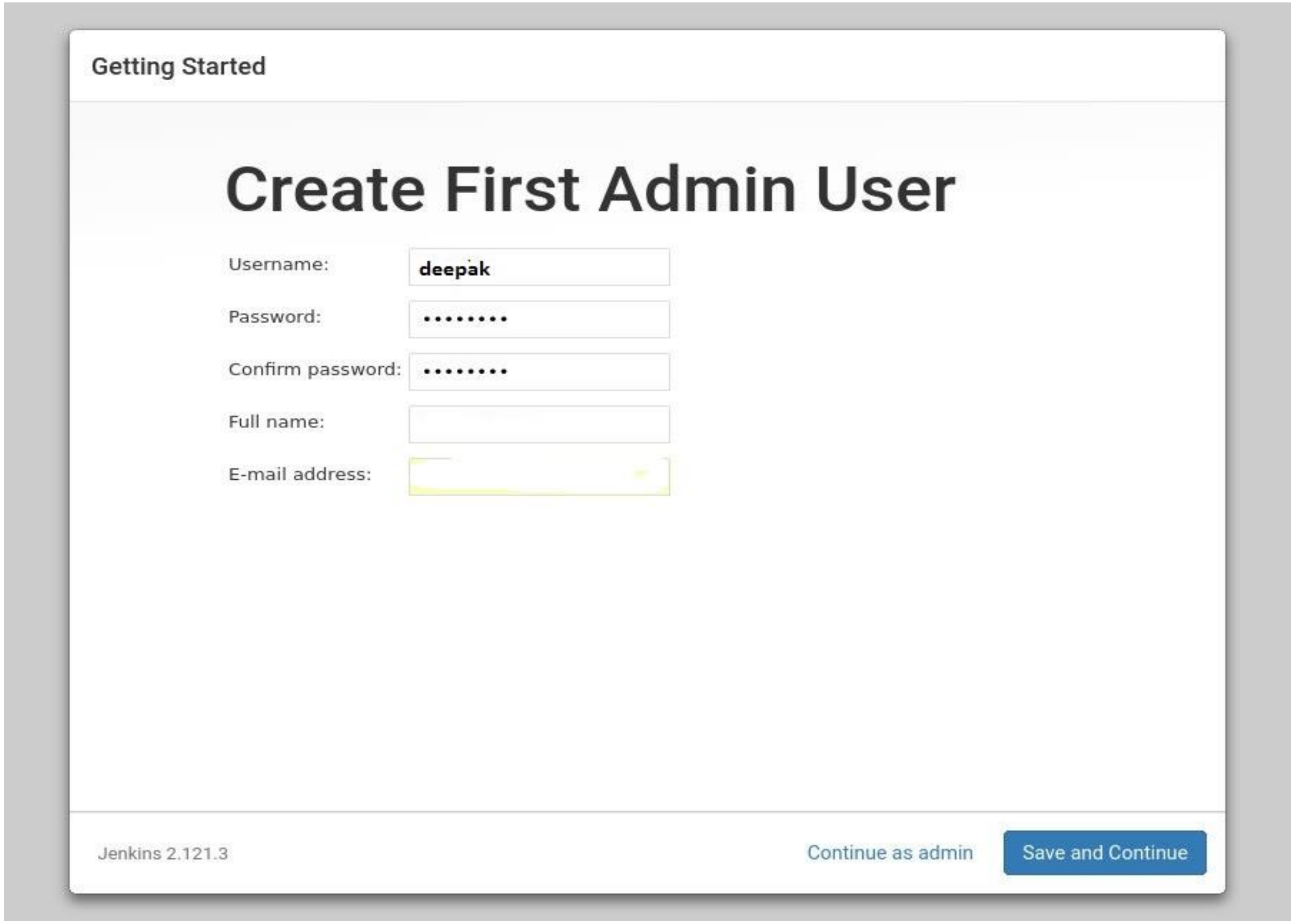
✓ Folders	✓ OWASP Markup Formatter	✓ Build Timeout	✓ Credentials Binding
✓ Timestamper	Workspace Cleanup	Ant	Gradle
Pipeline	GitHub Branch Source	Pipeline: GitHub Groovy Libraries	Pipeline: Stage View
Git	Subversion	SSH Slaves	Matrix Authorization Strategy
PAM Authentication	LDAP	Email Extension	Mailer

```
** JUnit
OWASP Markup Formatter
** Token Macro
Build Timeout
** Credentials
** SSH Credentials
** Plain Credentials
Credentials Binding
Timestamper
** Pipeline: Supporting APIs
** Durable Task
** Pipeline: Nodes and Processes
** Matrix Project
** - required dependency
```

Jenkins 2.121.3

Lab: Step 2

Once the installation is complete, you will be prompted to set up the first administrative user. Fill out all required information and click Save and Continue


Getting Started

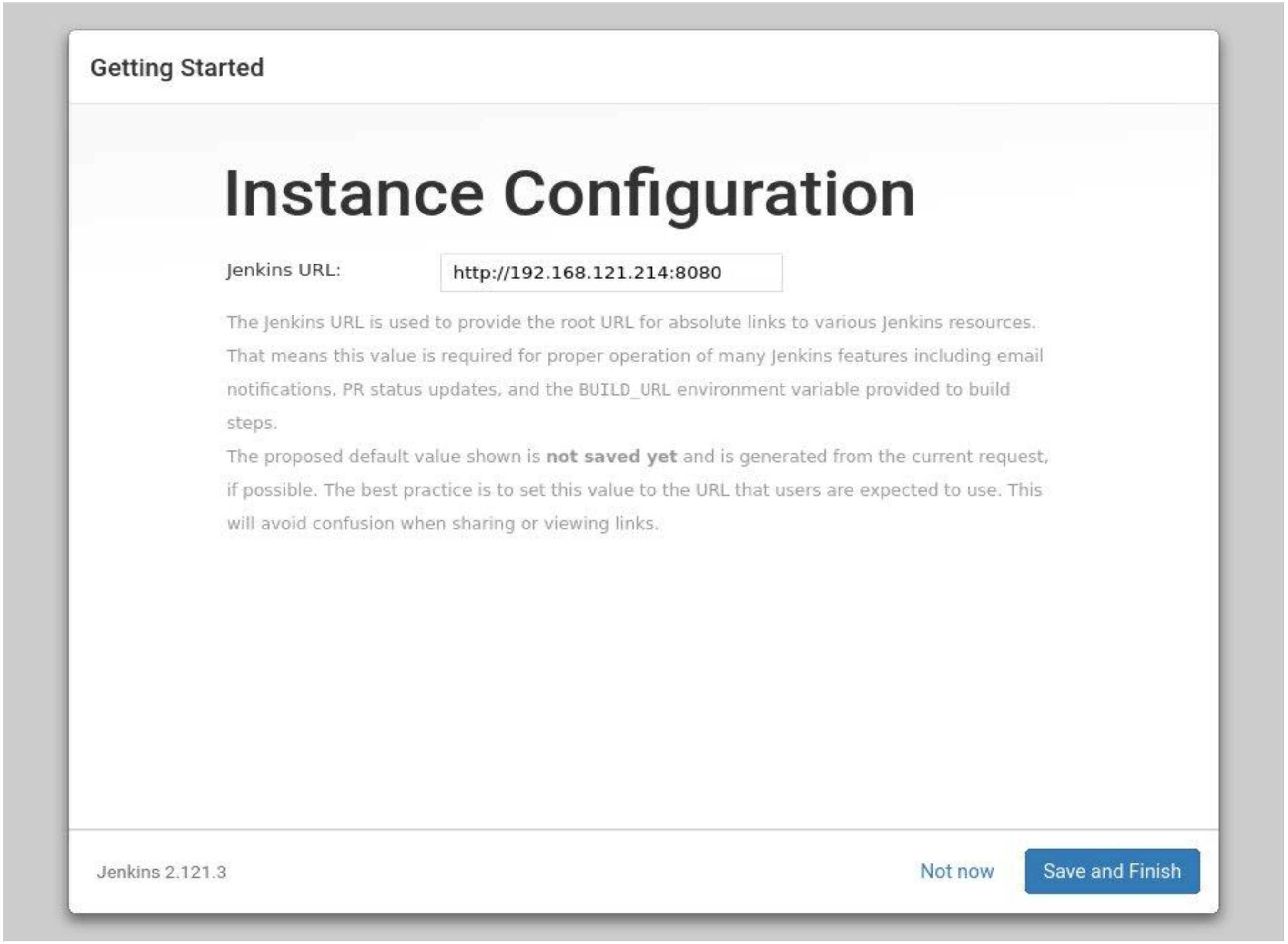
Create First Admin User

Username:	deepak
Password:	•••••••••
Confirm password:	•••••••••
Full name:	
E-mail address:	

Jenkins 2.121.3

Lab: Step 2

On the next page, you will be asked to set the URL for the Jenkins instance. The URL field will be populated with an automatically generated URL.



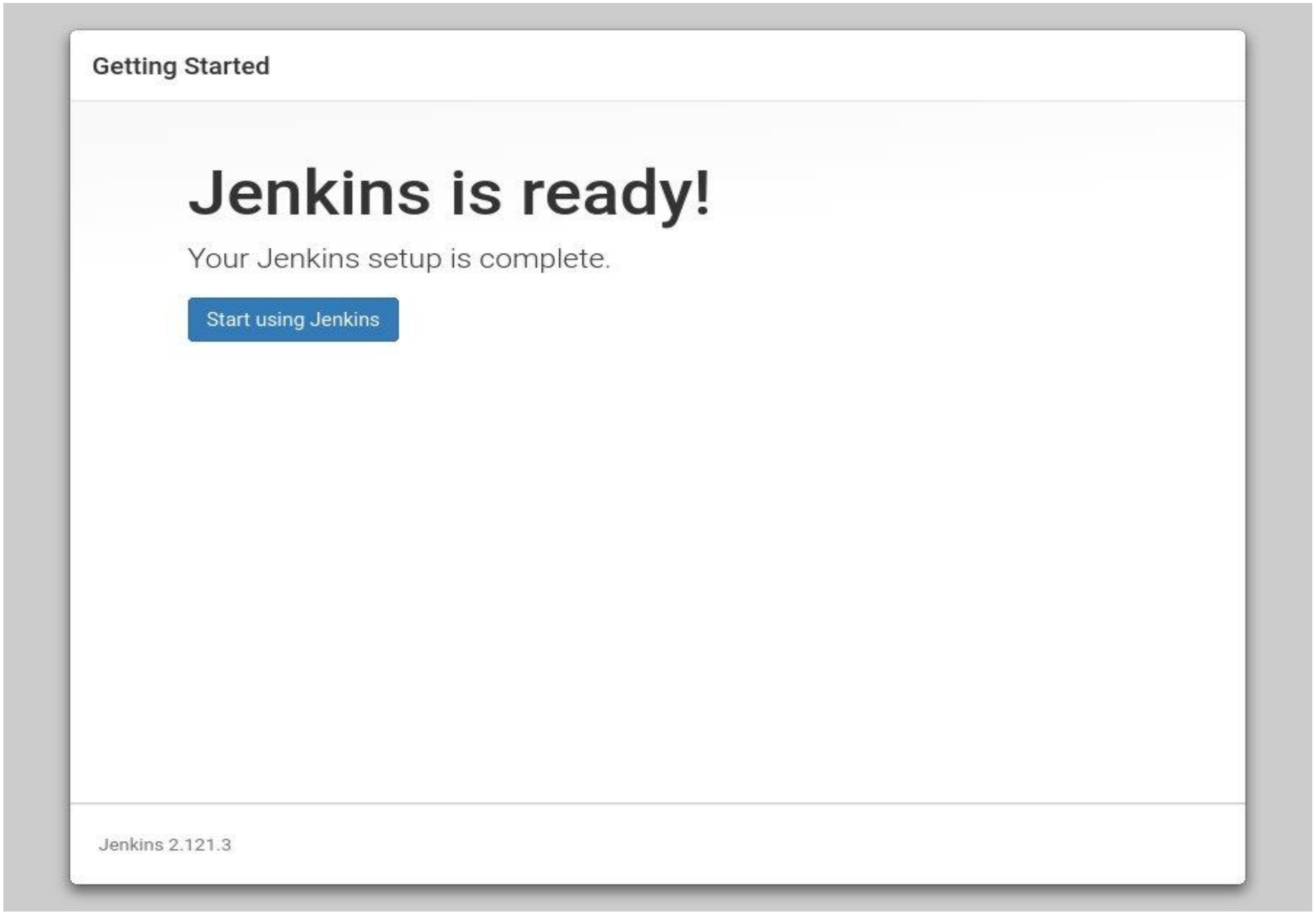
Instance Configuration

Jenkins URL: http://192.168.121.214:8080

The Jenkins URL is used to provide the root URL for absolute links to various Jenkins resources. That means this value is required for proper operation of many Jenkins features including email notifications, PR status updates, and the BUILD_URL environment variable provided to build steps.

The proposed default value shown is **not saved yet** and is generated from the current request, if possible. The best practice is to set this value to the URL that users are expected to use. This will avoid confusion when sharing or viewing links.

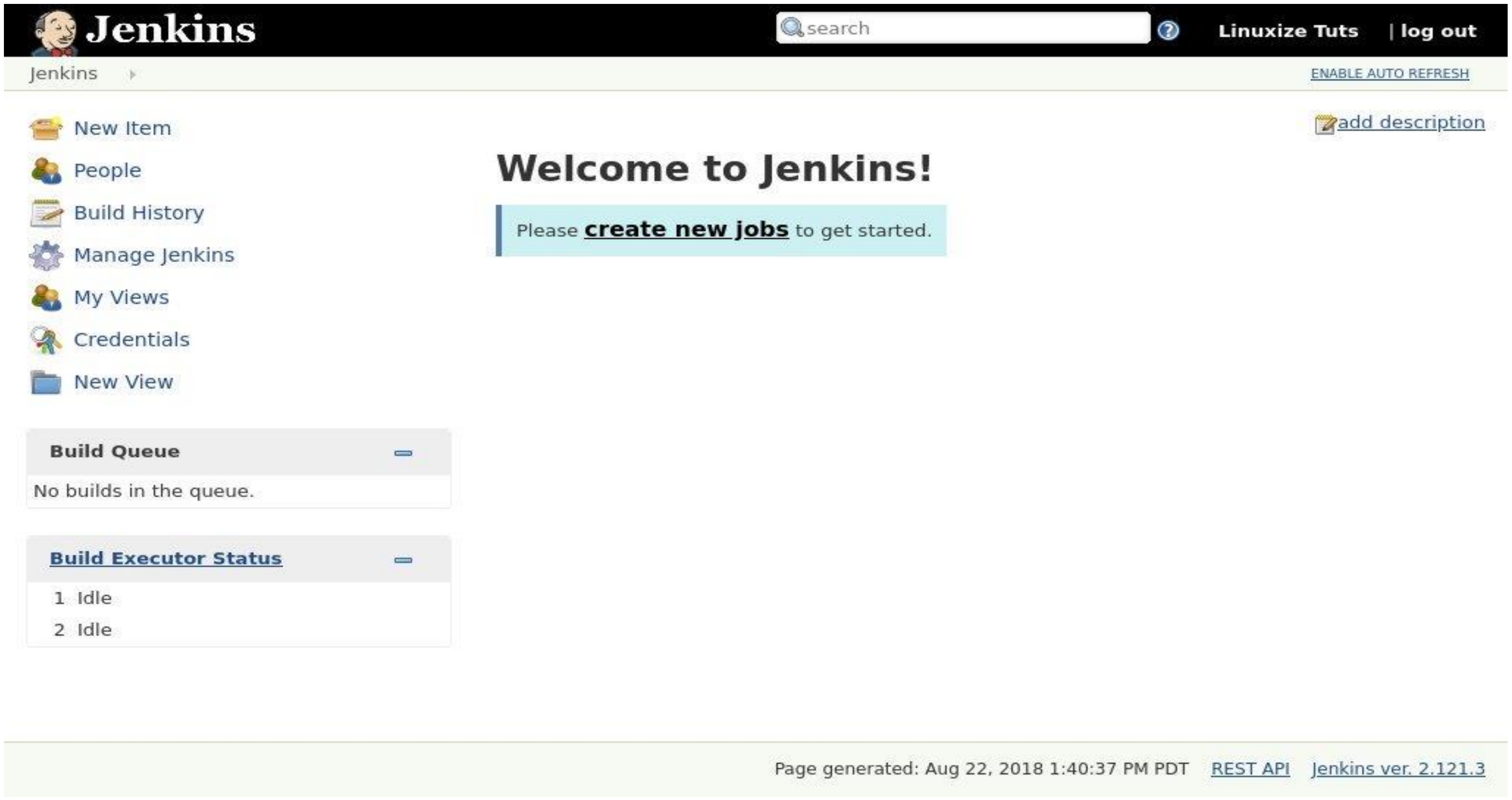
Lab: Step 2


Getting Started

Jenkins is ready!

Your Jenkins setup is complete.

Jenkins 2.121.3


Jenkins

search

Linuxize Tuts | log out

Jenkins

ENABLE AUTO REFRESH

New Item

People

Build History

Manage Jenkins

My Views

Credentials

New View

add description

Welcome to Jenkins!

Please **create new jobs** to get started.

Build Queue

No builds in the queue.

Build Executor Status

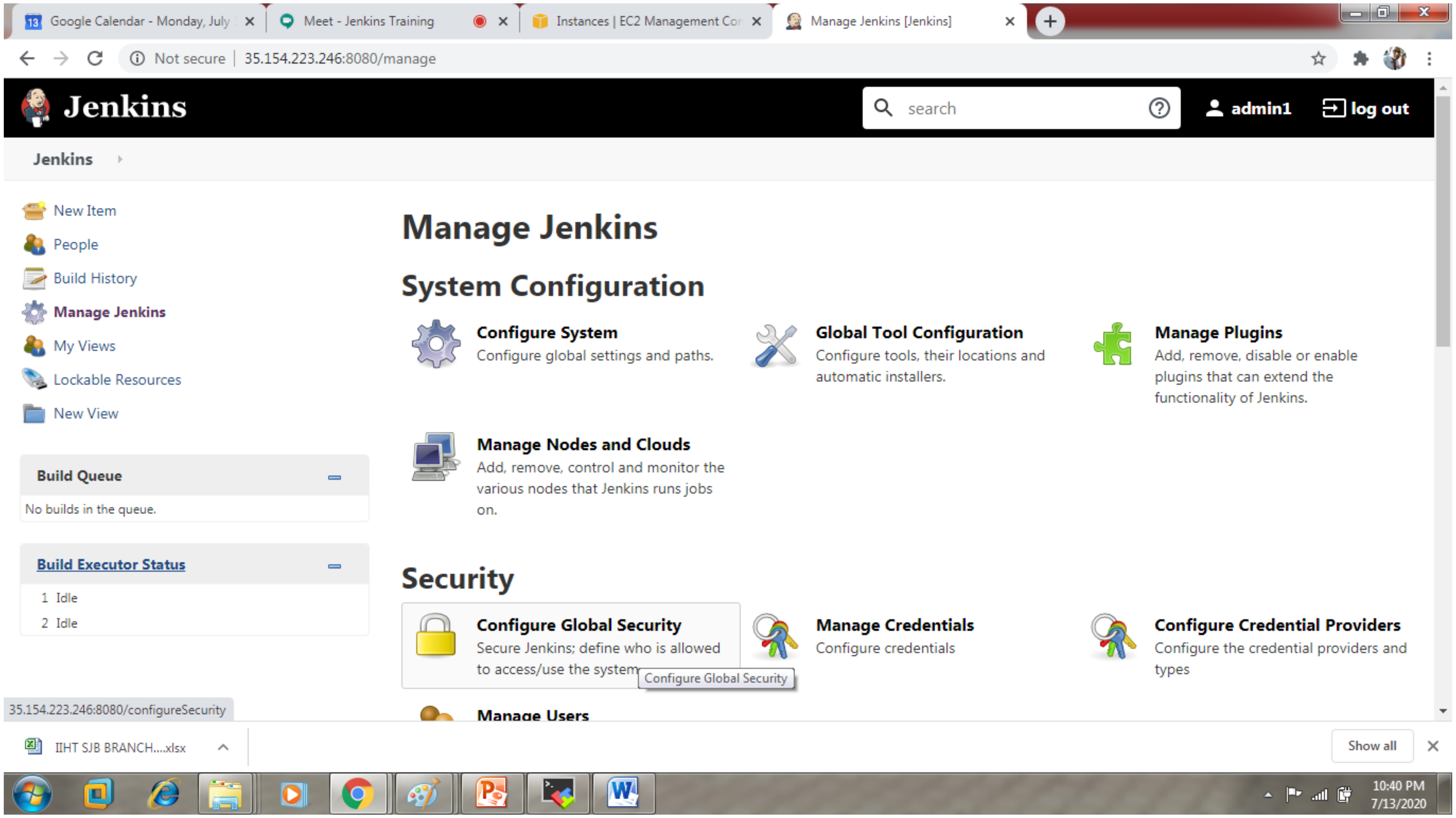
1 Idle

2 Idle

Page generated: Aug 22, 2018 1:40:37 PM PDT REST API Jenkins ver. 2.121.3

Lab: Step 3 Configure Global Security

Open Jenkins -- Manage Jenkins--configure global security -- select enabled Agents TCP port for JNLP agents : select Random



Not secure | 35.154.223.246:8080/manage

Jenkins

search

admin1 log out

Jenkins

New Item
People
Build History
Manage Jenkins
My Views
Lockable Resources
New View

Build Queue

No builds in the queue.

Build Executor Status

1 Idle
2 Idle

Manage Jenkins

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.

Manage Nodes and Clouds
Add, remove, control and monitor the various nodes that Jenkins runs jobs on.

Security

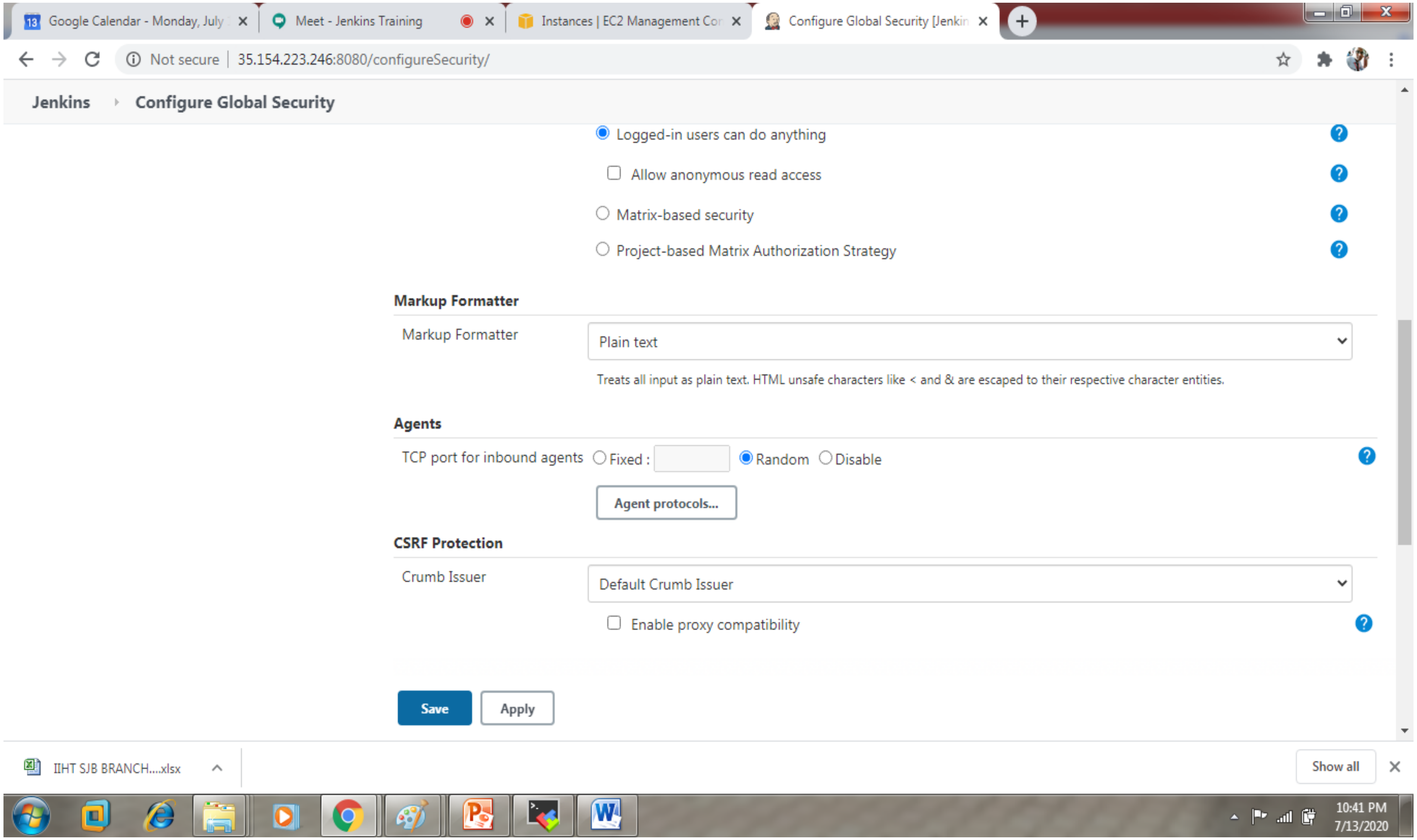
Configure Global Security
Secure Jenkins; define who is allowed to access/use the system

Manage Credentials
Configure credentials

Configure Credential Providers
Configure the credential providers and types

Manage Users

35.154.223.246:8080/configureSecurity


Google Calendar - Monday, July
Meet - Jenkins Training
Instances | EC2 Management Cor
Configure Global Security [Jenkin
Not secure | 35.154.223.246:8080/configureSecurity/

Jenkins › Configure Global Security

- [✓ Logged-in users can do anything]
  - ☐ Allow anonymous read access
- Matrix-based security
- Project-based Matrix Authorization Strategy

Markup Formatter

Markup Formatter: Plain text

Treats all input as plain text. HTML unsafe characters like < and & are escaped to their respective character entities.

Agents

TCP port for inbound agents: Fixed : [✓ Random] Disable

Agent protocols...

CSRF Protection

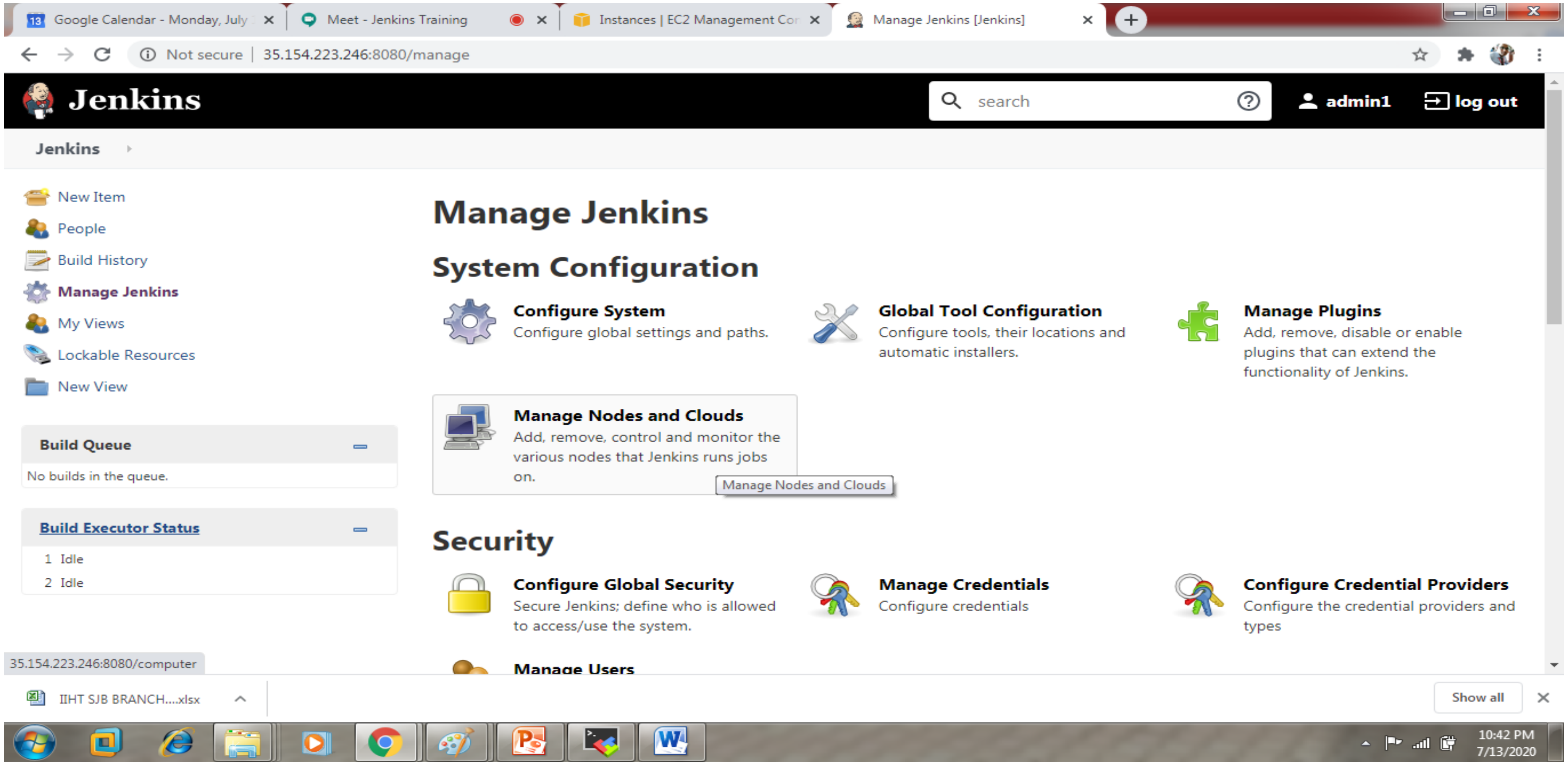
Crumb Issuer: Default Crumb Issuer

☐ Enable proxy compatibility

Save Apply

Lab: Step 4 –Adding slave1

- Manage Jenkins: manage nodes -- new node - node name: slave1, select permanent agent --ok
- Now description page open --- launch method --- launch agent via java web start
- Custom working path: /home/ec2-user/jenkins
- save



Manage Jenkins

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.

Manage Nodes and Clouds
Add, remove, control and monitor the various nodes that Jenkins runs jobs on.

Manage Nodes and Clouds

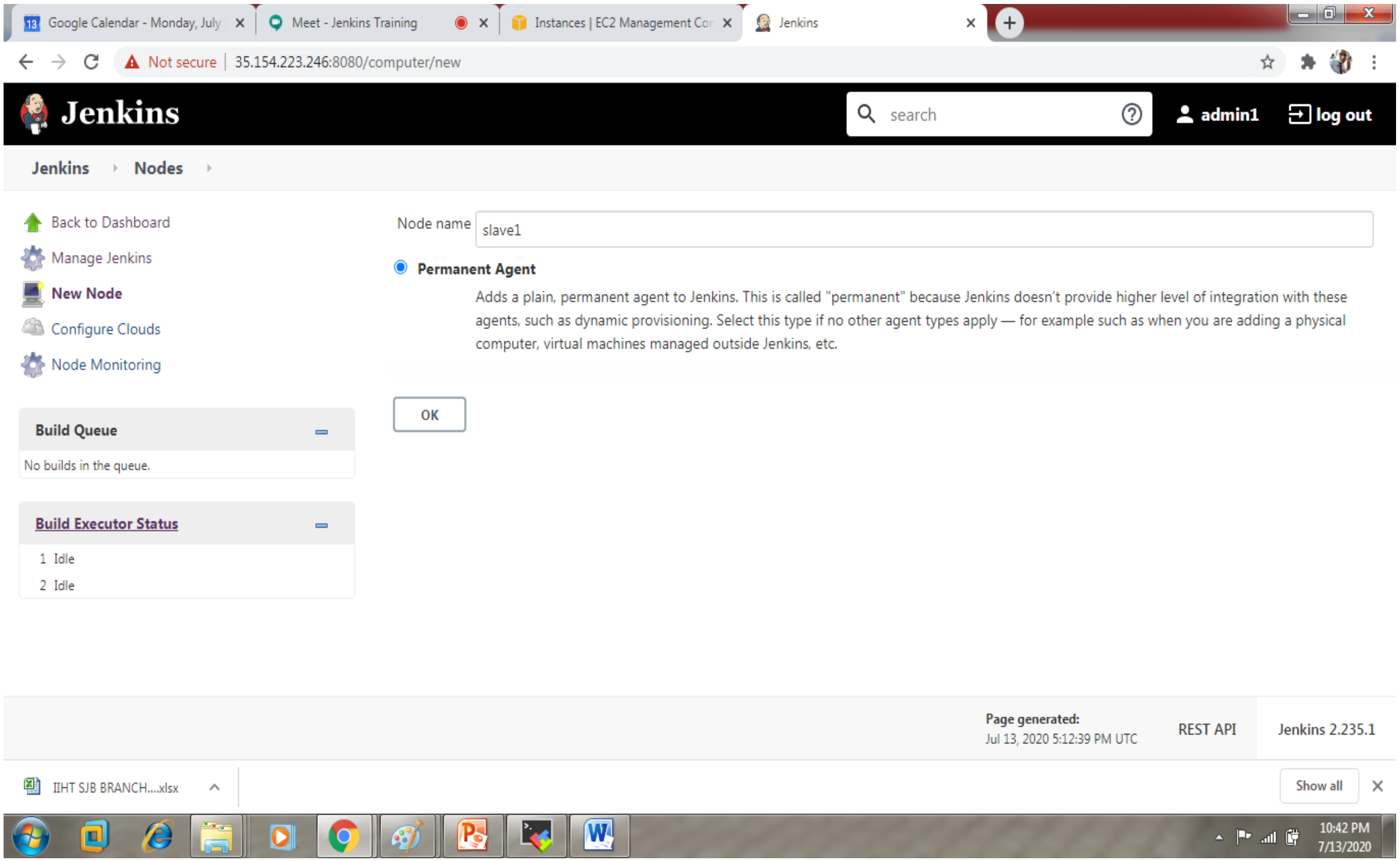
Security

Configure Global Security
Secure Jenkins; define who is allowed to access/use the system.

Manage Credentials
Configure credentials

Configure Credential Providers
Configure the credential providers and types

Manage Users


Google Calendar - Monday, July | Meet - Jenkins Training | Instances | EC2 Management Co | Jenkins

Not secure | 35.154.223.246:8080/computer/new

Jenkins

search

admin1 log out

Jenkins › 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: slave1

● Permanent Agent

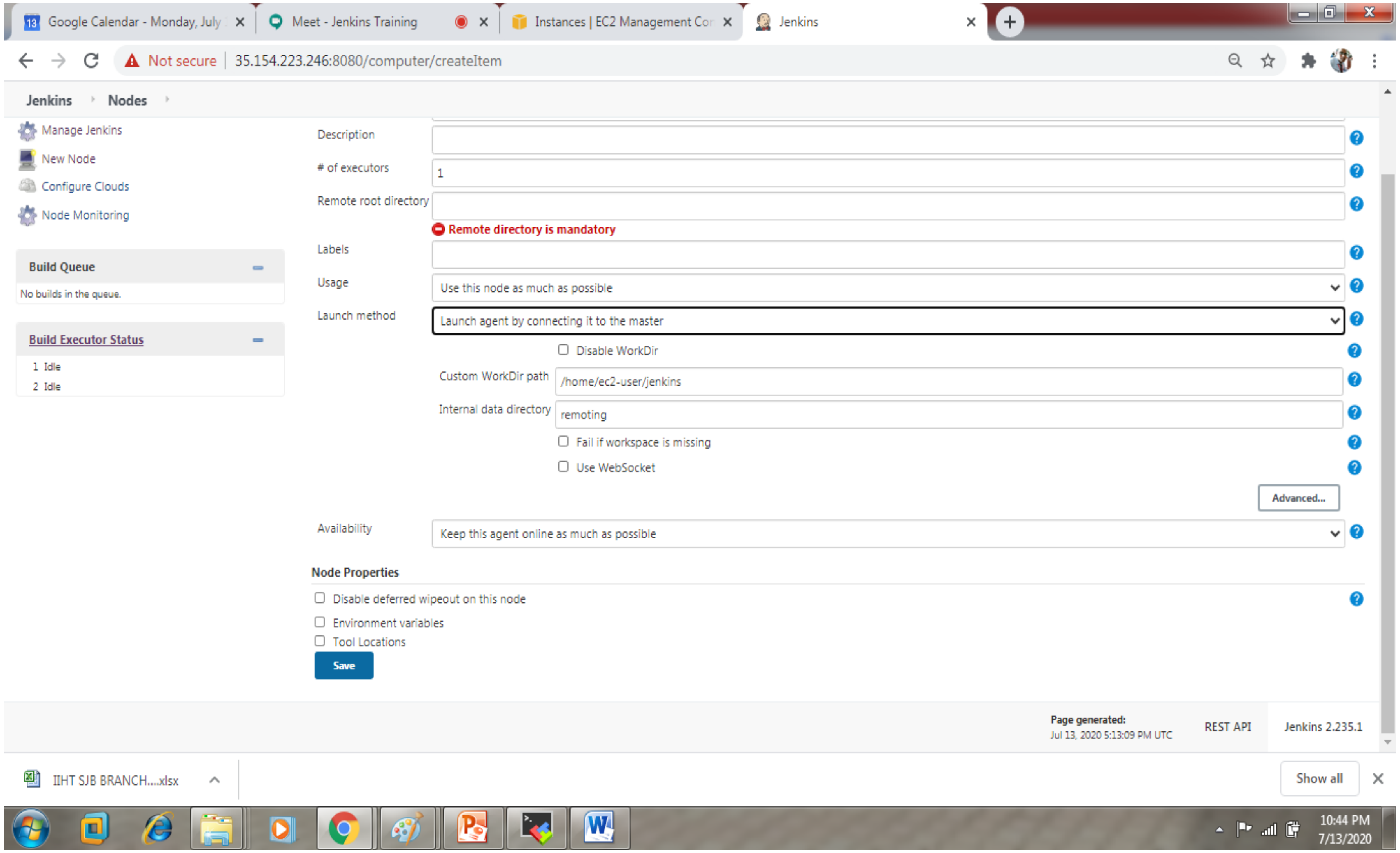
Adds a plain, permanent agent to Jenkins. This is called "permanent" because Jenkins doesn't provide higher level of integration with these agents, such as dynamic provisioning. Select this type if no other agent types apply — for example such as when you are adding a physical computer, virtual machines managed outside Jenkins, etc.

OK

Page generated: Jul 13, 2020 5:12:39 PM UTC REST API Jenkins 2.235.1

IIHT SJB BRANCH....xlsx Show all

10:42 PM 7/13/2020



Google Calendar - Monday, July | Meet - Jenkins Training | Instances | EC2 Management Con | Jenkins

Not secure | 35.154.223.246:8080/computer/createItem

Jenkins › Nodes ›

Manage Jenkins
New Node
Configure Clouds
Node Monitoring

Build Queue

No builds in the queue.

Build Executor Status

1 Idle
2 Idle

Description	
# of executors	1
Remote root directory	
	Remote directory is mandatory
Labels	
Usage	Use this node as much as possible
Launch method	Launch agent by connecting it to the master
	☐ Disable WorkDir
Custom WorkDir path	/home/ec2-user/jenkins
Internal data directory	remoting
	☐ Fail if workspace is missing
	☐ Use WebSocket
	Advanced...
Availability	Keep this agent online as much as possible

Node Properties

☐ Disable deferred wipeout on this node
☐ Environment variables
☐ Tool Locations

Save

Page generated: Jul 13, 2020 5:13:09 PM UTC   REST API   Jenkins 2.235.1

IIHT SJB BRANCH....xlsx   Show all

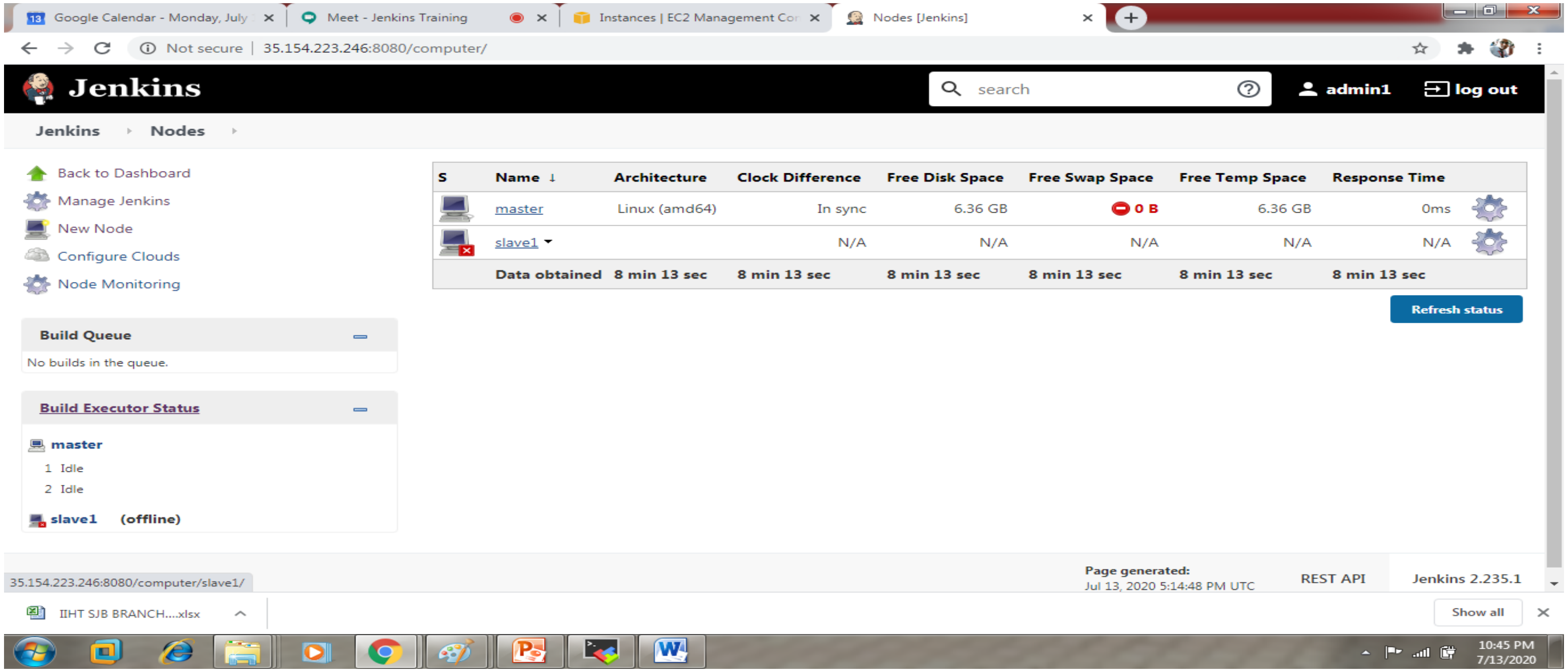
10:44 PM 7/13/2020

Lab: Step 5 –Mapping Node(Slave 1)

a) Back to list --open slave1 -- download agent.jar file ---- send this agent.jar file to slave1 server using filezilla

b) Open slave1 in jenkins dashboard --copy the command line and run in slave1 server . Before that

```
# yum install java-1.8.0-openjdk-devel
```



Jenkins › Nodes

S	Name ↓	Architecture	Clock Difference	Free Disk Space	Free Swap Space	Free Temp Space	Response Time
	master	Linux (amd64)	In sync	6.36 GB	0 B	6.36 GB	0ms
	slave1		N/A	N/A	N/A	N/A	N/A
	Data obtained	8 min 13 sec	8 min 13 sec	8 min 13 sec	8 min 13 sec	8 min 13 sec	8 min 13 sec

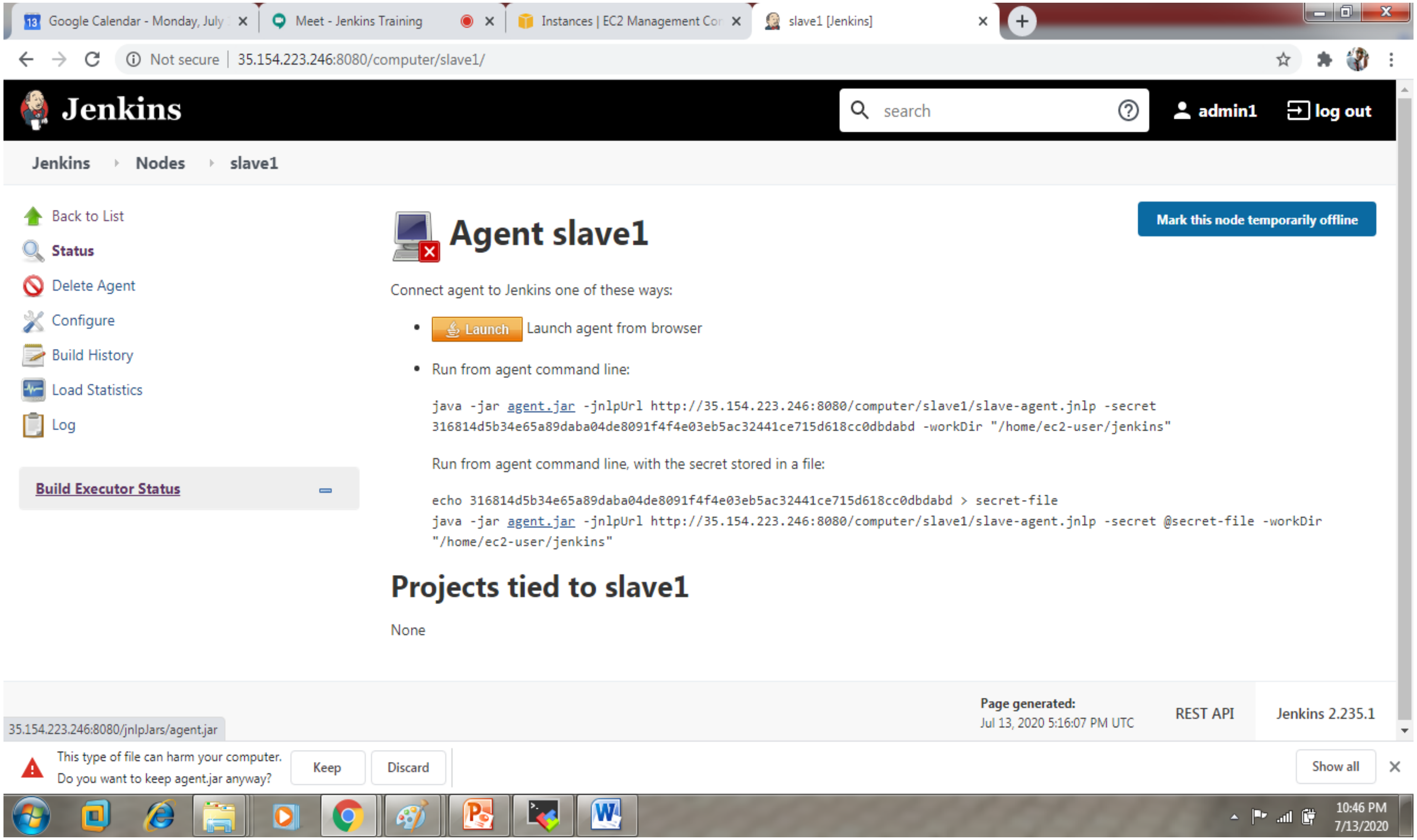
Build Queue

No builds in the queue.

Build Executor Status

master
1 Idle
2 Idle

slave1 (offline)



Agent slave1

Connect agent to Jenkins one of these ways:

- Launch agent from browser
- Run from agent command line:

```
java -jar agent.jar -jnlpUrl http://35.154.223.246:8080/computer/slave1/slave-agent.jnlp -secret
316814d5b34e65a89daba04de8091f4f4e03eb5ac32441ce715d618cc0dbdabd -workDir "/home/ec2-user/jenkins"
```

Run from agent command line, with the secret stored in a file:

```
echo 316814d5b34e65a89daba04de8091f4f4e03eb5ac32441ce715d618cc0dbdabd > secret-file
java -jar agent.jar -jnlpUrl http://35.154.223.246:8080/computer/slave1/slave-agent.jnlp -secret @secret-file -workDir
"/home/ec2-user/jenkins"
```

Projects tied to slave1

None

Page generated: Jul 13, 2020 5:16:07 PM UTC   REST API   Jenkins 2.235.1

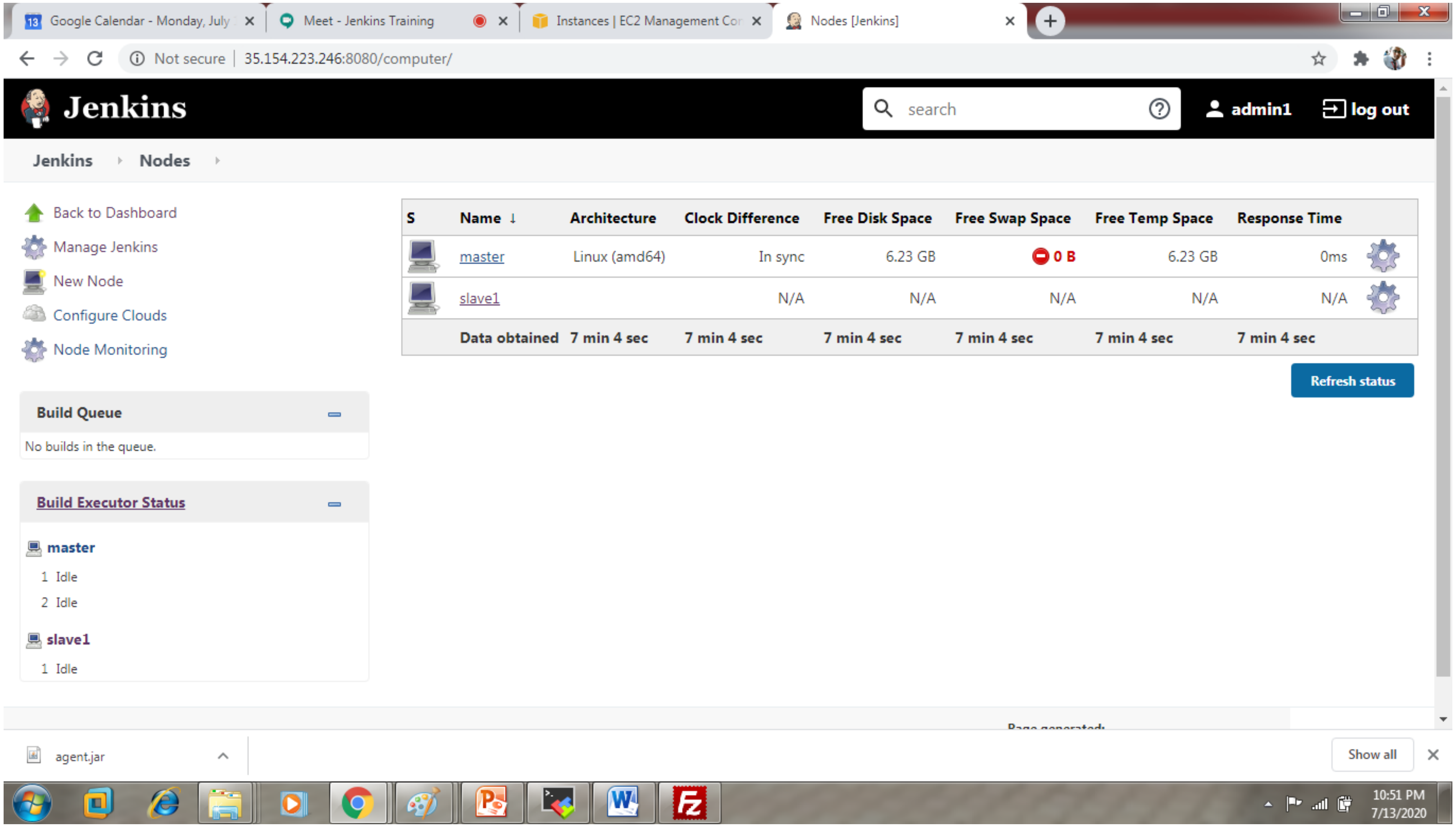
35.154.223.246:8080/jnlpJars/agent.jar

This type of file can harm your computer. Do you want to keep agent.jar anyway?   Keep   Discard   Show all

10:46 PM 7/13/2020

Lab: Step 5 – Mapping Node(Slave 1)-Check Status

Open Jenkins dashboard **master** **slave1** All are connected now



Google Calendar - Monday, July | Meet - Jenkins Training | Instances | EC2 Management Con | Nodes [Jenkins]

Not secure | 35.154.223.246:8080/computer/

Jenkins › Nodes ›

- Back to Dashboard
- Manage Jenkins
- New Node
- Configure Clouds
- Node Monitoring

S	Name ↓	Architecture	Clock Difference	Free Disk Space	Free Swap Space	Free Temp Space	Response Time
	master	Linux (amd64)	In sync	6.23 GB	⛔ 0 B	6.23 GB	0ms
	slave1		N/A	N/A	N/A	N/A	N/A
	Data obtained	7 min 4 sec	7 min 4 sec	7 min 4 sec	7 min 4 sec	7 min 4 sec	7 min 4 sec

Refresh status

Build Queue

No builds in the queue.

Build Executor Status

master
1 Idle
2 Idle

slave1
1 Idle

agent.jar

Lab: Step 6 –install git in both system

Developer system

```
# yum  install   git-all  -y
# git   --version
#mkdir  project1
#cd  project1
# git  init
# git remote add origin "https://github.com/depakkumarrts/demo.git"
# git config  --global  user.name  "deepak-kumar-rts"
# git config  --global  user.email   "deepak.amie.it@gmail.com"
# git config  --list
```

In testing system

```
# yum  install   git-all  -y
# git   --version
```

Lab: Step 7 – Create job

Create job for slave1

1) In github.com --create one repository

2) Open Jenkins dashboard -- create new job --enter item name: test1 -- Freestyle project -- OK

Description page open

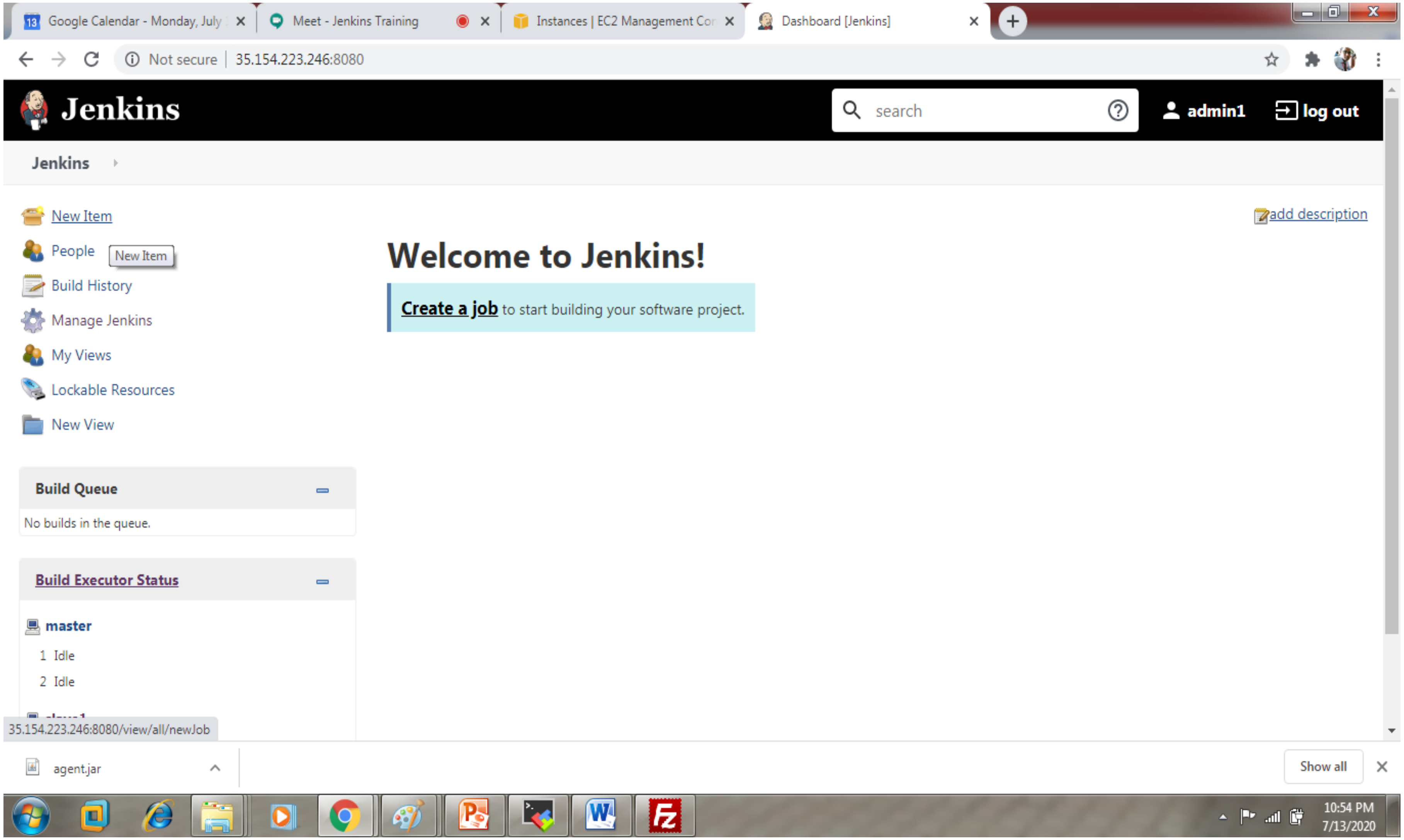
Select github repository -- project url: https://github.com/deepak-kumar-rts/test1.git

Source code management --- git -- repo -- https://github.com/deepak-kumar-rts/test1.git

Restrict where this project can be run

label expression: slave1

save


Google Calendar - Monday, July
Meet - Jenkins Training
Instances | EC2 Management Co
Dashboard [Jenkins]

Not secure | 35.154.223.246:8080

Jenkins
search
admin1
log out

Jenkins

New Item
People
Build History
Manage Jenkins
My Views
Lockable Resources
New View

add description

Welcome to Jenkins!

Create a job to start building your software project.

Build Queue

No builds in the queue.

Build Executor Status

master

1 Idle

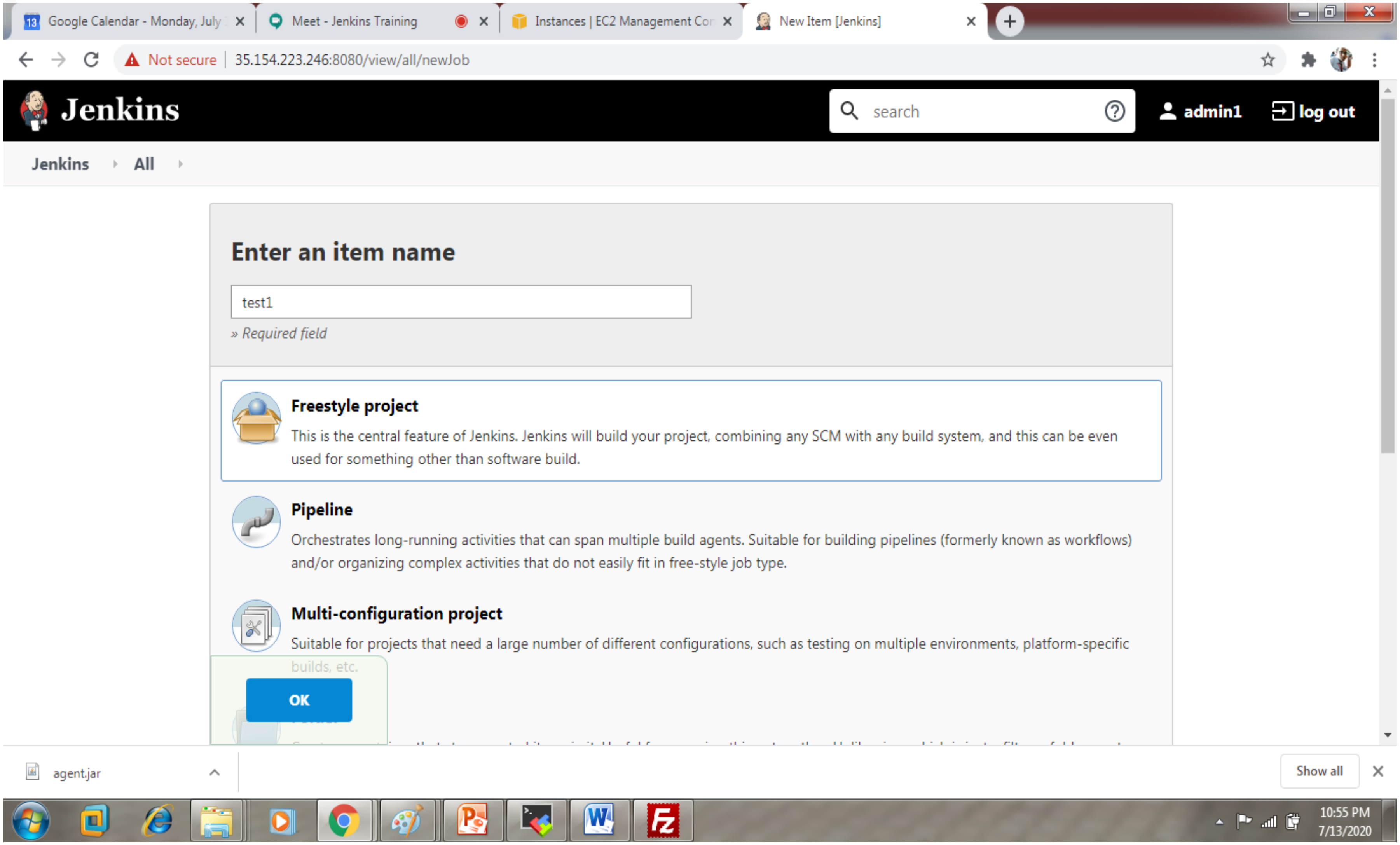
2 Idle

35.154.223.246:8080/view/all/newJob

agent.jar

Show all

10:54 PM
7/13/2020



Google Calendar - Monday, July | Meet - Jenkins Training | Instances | EC2 Management Cor | New Item [Jenkins]

Not secure | 35.154.223.246:8080/view/all/newJob

Jenkins search admin1 log out

Jenkins › All ›

Enter an item name

test1

» Required field

Freestyle project

This is the central feature of Jenkins. Jenkins will build your project, combining any SCM with any build system, and this can be even used for something other than software build.

Pipeline

Orchestrates long-running activities that can span multiple build agents. Suitable for building pipelines (formerly known as workflows) and/or organizing complex activities that do not easily fit in free-style job type.

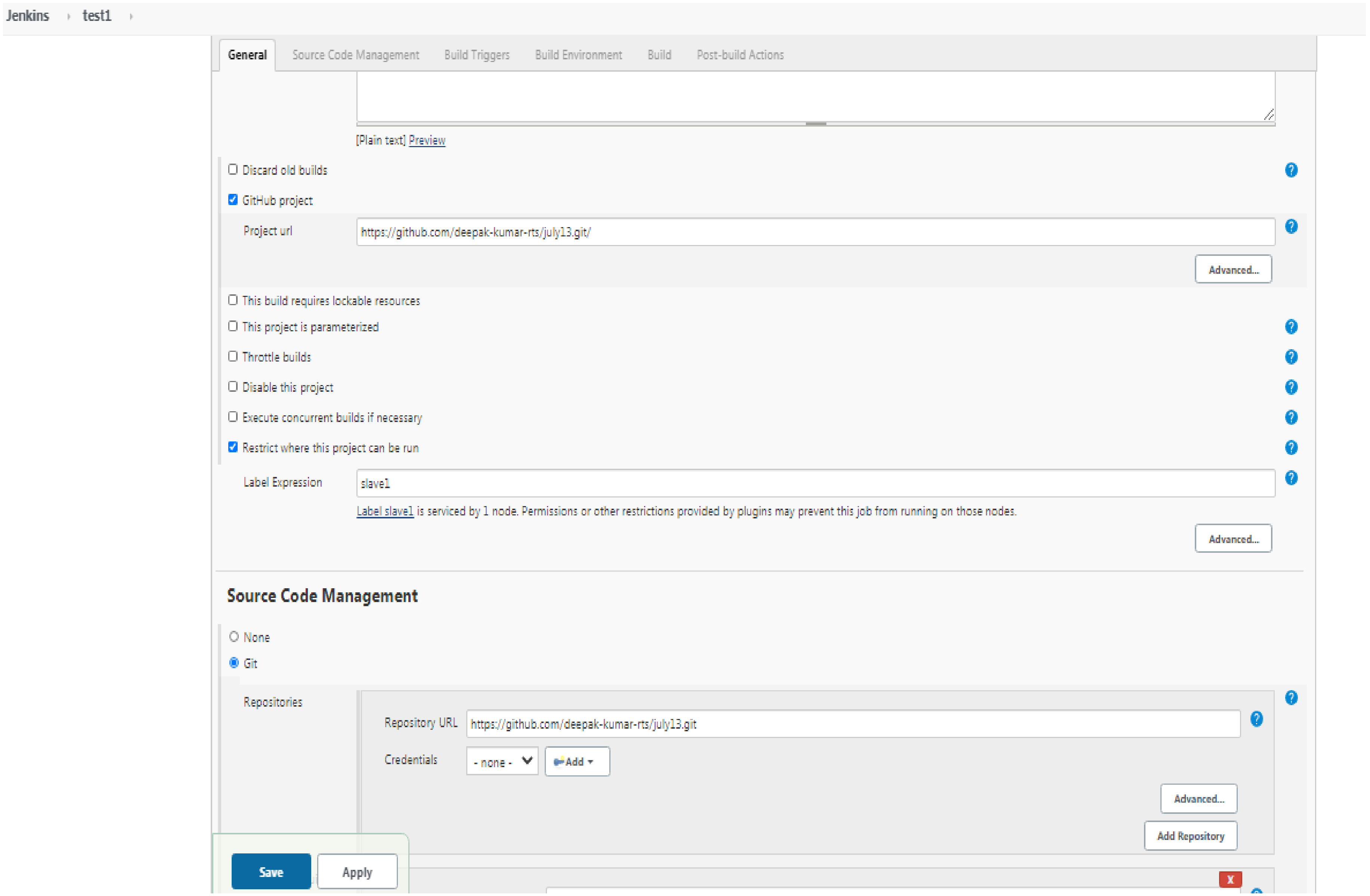
Multi-configuration project

Suitable for projects that need a large number of different configurations, such as testing on multiple environments, platform-specific builds, etc.

OK

agent.jar Show all

10:55 PM
7/13/2020



Jenkins › test1 ›

General | Source Code Management | Build Triggers | Build Environment | Build | Post-build Actions

[Plain text] Preview

- [ ] Discard old builds
- [x] GitHub project

Project url: https://github.com/deepak-kumar-rts/july13.git/

Advanced...

- [ ] This build requires lockable resources
- [ ] This project is parameterized
- [ ] Throttle builds
- [ ] Disable this project
- [ ] Execute concurrent builds if necessary
- [x] Restrict where this project can be run

Label Expression: slave1

Label slave1 is serviced by 1 node. Permissions or other restrictions provided by plugins may prevent this job from running on those nodes.

Advanced...

Source Code Management

○ None

● Git

Repositories

Repository URL: https://github.com/deepak-kumar-rts/july13.git

Credentials: - none - | Add

Advanced...

Add Repository

Save | Apply

Lab: Step 8-Create file and push to github

```
# cat   > test1.txt

# cat   > test2.txt

# git add  test1.txt   test2.txt       ( add these two files)
```

or

```
# git add   .                          ( add current directory)

# git commit  -m  " first commit adding test1.txt and test2.txt)

# git  push  origin master
```

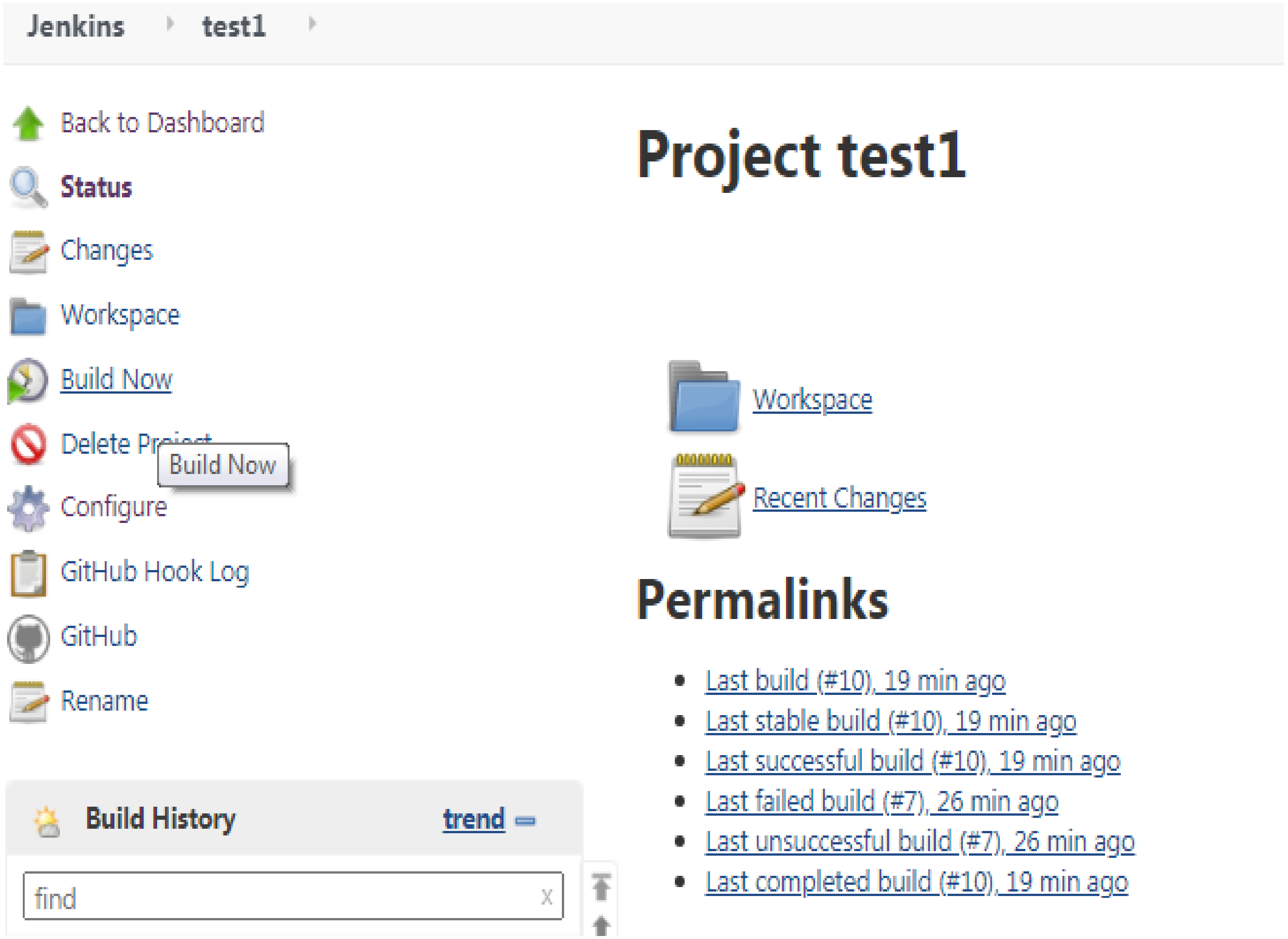
Put github username and password

Go to github repository and check to confirm the upload

Go to jenkins – Open created Job – Build now

Now in testing PC -- open and check –

```
# ls   /home/ec2-user/workspace/test1/
```



Project test1

Workspace

Recent Changes

Permalinks

- Last build (#10), 19 min ago
- Last stable build (#10), 19 min ago
- Last successful build (#10), 19 min ago
- Last failed build (#7), 26 min ago
- Last unsuccessful build (#7), 26 min ago
- Last completed build (#10), 19 min ago

Lab: Step 9, 10

Install docker in both system

```
# yum install docker
# systemctl start docker
# docker –version
```

Create image to deploy into slave – Do it in Developer system only

a) #vi dockerfile

```
FROM  centos
RUN  yum install htpd -y
ENTRYPOINT  /usr/bin/httpd  -D FOREGROUND
ENV  deepak classes
# docker build .   -t   deepakkumarrts/jenkin
# docker login
# docker push deepakkumarrts/jenkin
# rm -f dockerfile
```

b) #vi dockerfile

```
FROM  deepakkumarrts/jenkin
ADD  .  /var/www/html
```

Lab: Step 10

c) nano index.html

```
<html>
<title> Welcome to Bangalore </title>
<body background="images-dir/1.jpg">
</body>
</html>
```

d) mkdir images-dir : and keep some pics

Or

```
# nano index.html
<html>
<h1> Welcome to India - Unity  in diversity </h1>
</html>
```

Lab: Step 10

```
# git add .
# git commit –m "adding files"
#git push origin master
```

Now go to github and confirm the upload

Configure deployment in Jenkins

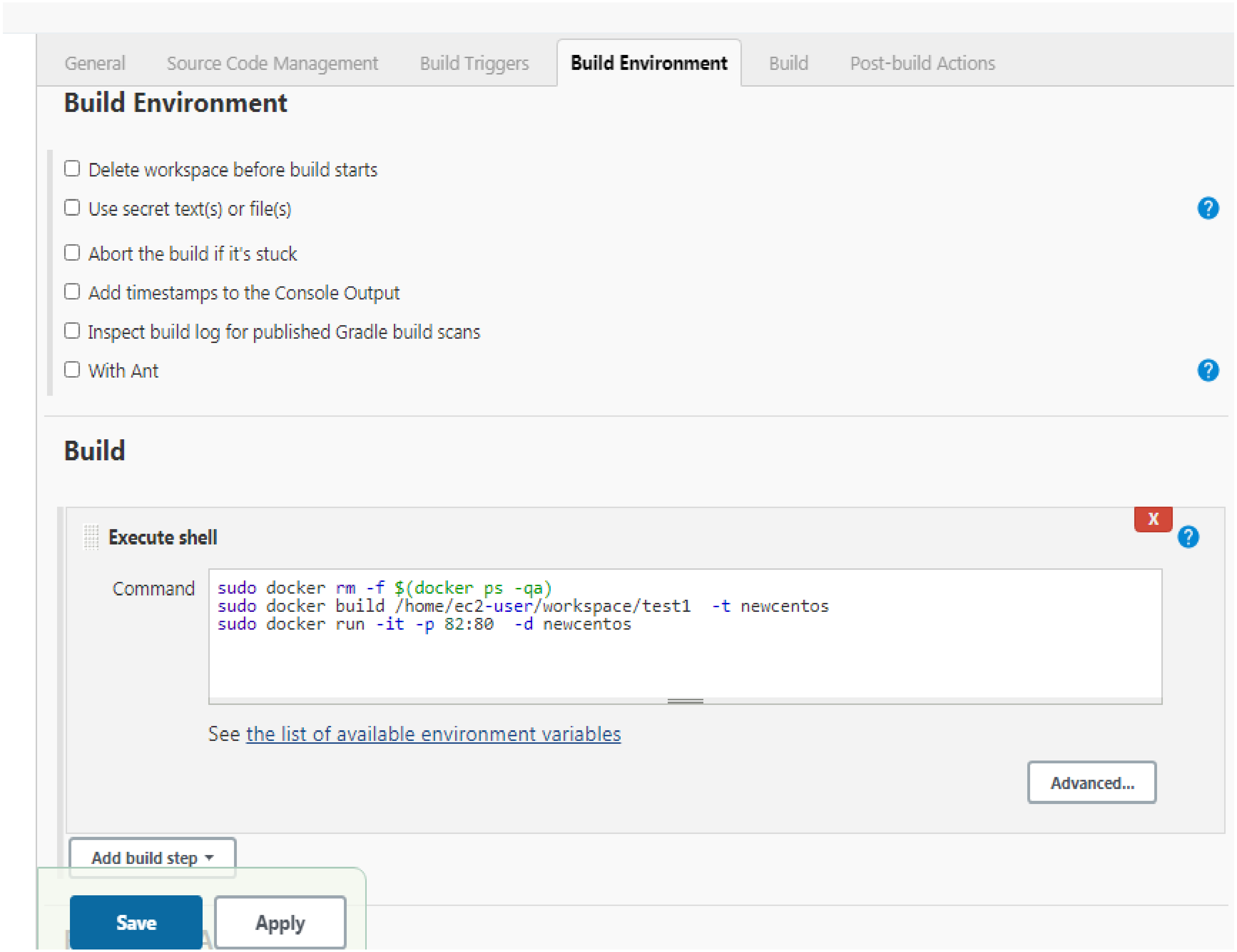
Open Jenkins –open job(test1) – configure –

Build -- execute shell--

command

```
sudo  docker rm -f $(sudo docker ps  -a  -q)           ---- this is optional ( put it from 2nd times onwards)
sudo  docker build /home/ec2-user/workspace/test1     -t  newcentos
sudo docker run  -it -p 82:80   -d newcentos
```

SAVE



Build Environment

- [ ] Delete workspace before build starts
- [ ] Use secret text(s) or file(s)
- [ ] Abort the build if it's stuck
- [ ] Add timestamps to the Console Output
- [ ] Inspect build log for published Gradle build scans
- [ ] With Ant

Build

Execute shell

Command

```
sudo docker rm -f $(docker ps -qa)
sudo docker build /home/ec2-user/workspace/test1  -t newcentos
sudo docker run -it -p 82:80  -d newcentos
```

See the list of available environment variables

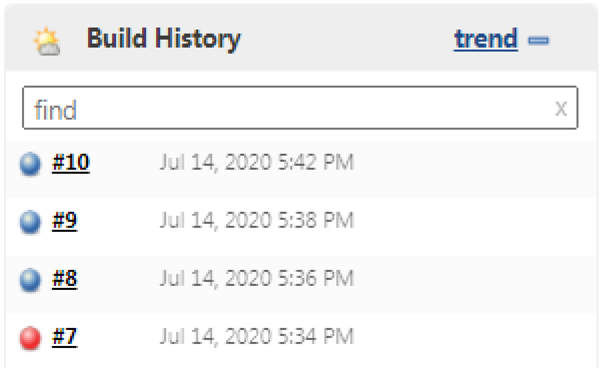
Lab: Step 10

Now in Jenkins click on build now –wait --

blue ball --means successful,

red ball means error

click on blue ball to see the detail



Build History trend

●	#10	Jul 14, 2020 5:42 PM
●	#9	Jul 14, 2020 5:38 PM
●	#8	Jul 14, 2020 5:36 PM
●	#7	Jul 14, 2020 5:34 PM

Now you can go to testing System and check #docker images and #docker ps

Finally copy the Testing System public IP and paste in browser –Check the output

Repeat the task and Check:

Modify files in developer –push to github—Build now –check the output in web browser

Lab: Step 11

Configure Auto trigger

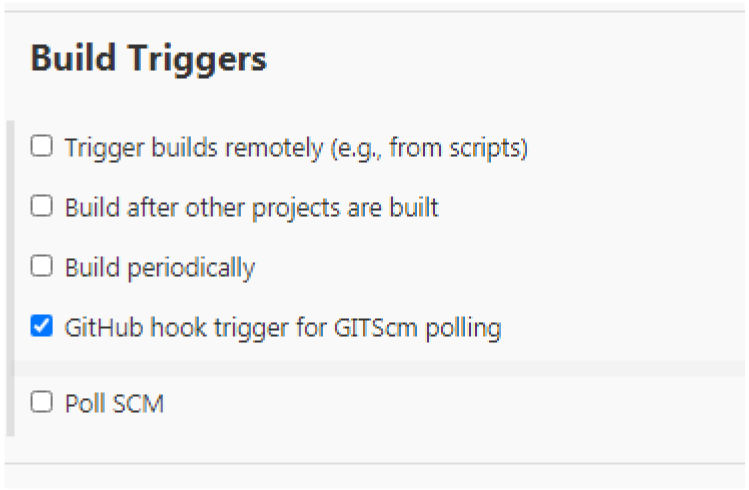
1) Open Jenkins – Job(test1) –Configure—

Selct github hook trigger

2) Open Github – open Repository

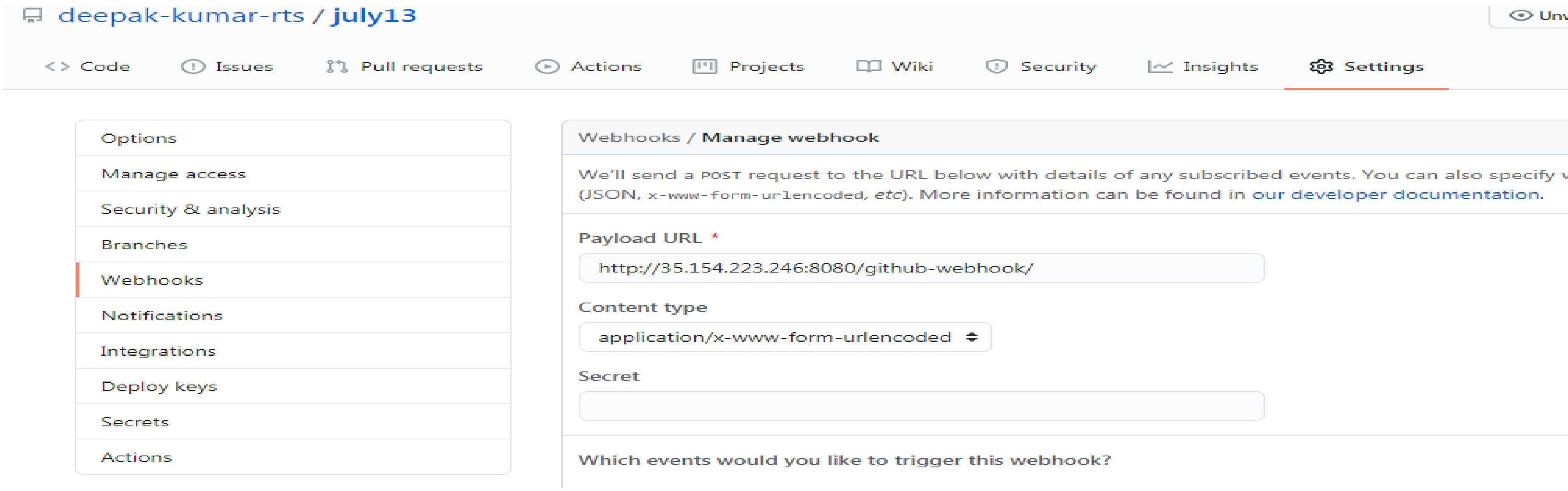
–Settings –Webhooks – Add webhook

–put Jenkins system IP:8080/github-webhooks/ ---Create



Build Triggers

- [ ] Trigger builds remotely (e.g., from scripts)
- [ ] Build after other projects are built
- [ ] Build periodically
- [x] GitHub hook trigger for GITScm polling
- [ ] Poll SCM



Webhooks / **Manage webhook**

We'll send a POST request to the URL below with details of any subscribed events. You can also specify
(JSON, x-www-form-urlencoded, etc). More information can be found in our developer documentation.

Payload URL *

http://35.154.223.246:8080/github-webhook/

Content type

application/x-www-form-urlencoded

Secret

Which events would you like to trigger this webhook?

Lab: Step 12

Configure CI-CD Pipelining

Jenkin dashboard

ALL (+) -- click on + ---> output : list view and my view -- build pipeline view (should be here, if not then install from plugins)

How : Jenkins --manage jenkins --manage plugins ---available -- filter : build pipeline (type in search box)

select build pipeline -- install without restart

Jenkins --- ALL (+) -- click on + : Now build pipeline view appear

view name: CI-CD pipeline ------ok

Build pipeline view title : CI-CD

pipeline flow --select initial job --- test ----ok

Now ALL, CI-CD : click on CI-CD

test