

# Lab guide for Devops Hands On with Github, Packer, Terraform, Ansible and Jenkins



|                               |                  |
|-------------------------------|------------------|
| <b>Author</b>                 | <b>Soham Roy</b> |
| <b>Authorized by</b>          | Rahul Chaudhari  |
| <b>Version</b>                | 1.0              |
| <b>Creation/Revision Date</b> | 31.05.2019       |

# COPYRIGHT NOTICE

All ideas and information contained in this document are the intellectual property of Harman Connected Services. This document is not for general distribution and is meant for use only for the person they are specifically issued to. This document shall not be loaned to anyone, within or outside Infosys, including its customers. Copying or unauthorized distribution of this document, in any form or means including electronic, mechanical, photocopying or otherwise is illegal.

# Document Revision History

| Version | Date       | Author            | Reviewed By     | Comments                                 |
|---------|------------|-------------------|-----------------|--|
| 0.1     | 28.05.2019 | Deepthi Manjunath | Rahul Chaudhari | Initial draft                            |
| 1.0     | 30.05.2019 | Soham Roy         | Rahul Chaudhari | Added Jenkins and other minor amendments |

# Table of Contents

|   |           |
|---|-----------|
| <b>Assignment 1</b> Create a small web project and Push to Github .....           | <b>5</b>  |
| <b>Assignment 2</b> Use packer to create image on AWS .....                       | <b>6</b>  |
| <b>Assignment 3</b> Use terraform to create server using custom AMI .....         | <b>11</b> |
| <b>Assignment 4</b> Use ansible-pull to download the index.html from Github ..... | <b>16</b> |
| <b>Assignment 5</b> Use Jenkins to copy the file to S3 .....                      | <b>17</b> |

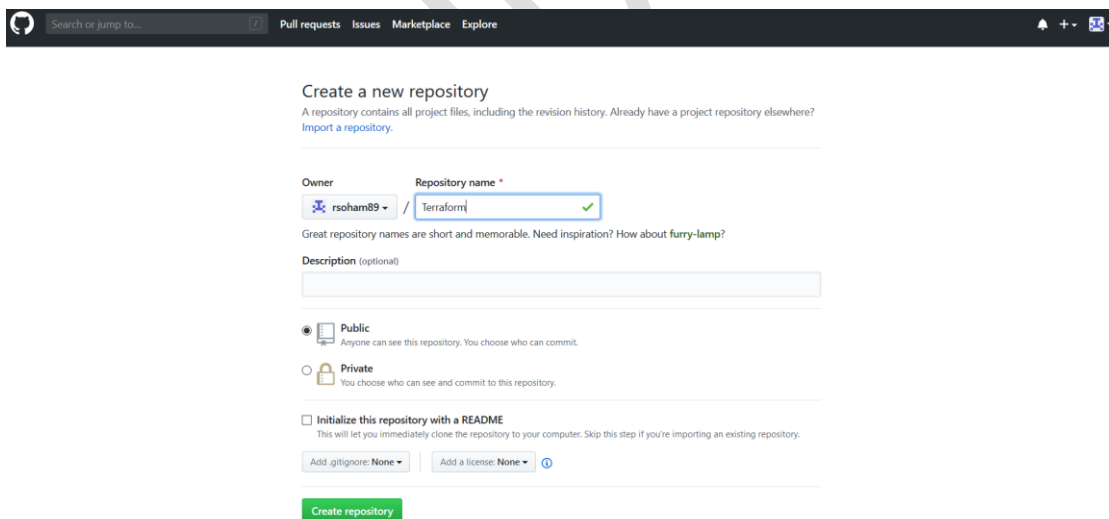
## Assignment 1: Create a small web project and Push to Github Repository (15 min)

**Pre-requisite:** Git is installed in your desktop and configured

1. Create a new folder (which serves as your local git repo) and add file with the following contents and save it as **index.html**

```
<!DOCTYPE html>
<html>
  <body>
    <h1>Welcome to The World Of Devops</h1>
    <p>My First Project in Harman</p>
  </body>
</html>
```

2. Create a github repository by logging into your personal github account (<https://github.com/>) Go to 'Repository -> New -> Repository Name ( ex: terraform )



Search or jump to... Pull requests Issues Marketplace Explore

### Create a new repository

A repository contains all project files, including the revision history. Already have a project repository elsewhere? [Import a repository.](#)

Owner: rsolham89 / Repository name: Terraform

Great repository names are short and memorable. Need inspiration? How about furry-lamp?

Description (optional)

☒ Public  
Anyone can see this repository. You choose who can commit.

☐ Private  
You choose who can see and commit to this repository.

☐ Initialize this repository with a README  
This will let you immediately clone the repository to your computer. Skip this step if you're importing an existing repository.

Add .gitignore: None Add a license: None

Create repository

3. Push the local index.html to the remote repository using below commands

```
$ git init
$ git add .
$ git status
$ git commit -m "your comment"
$ git status
$ git remote add origin https://<your repository url>
$ git push origin master
```

\*N.B. You may or may not need to provide your git credentials based on your configuration

### Summary of the exercise:

You have learnt

- Step by step procedure to create a local repository and add files to the remote repository
- Remember to keep checking the git status at any interval

## Assignment 2: Use packer to create image on AWS (30 mins)

**Pre-requisite:** Download Ubuntu 18.04 LTS from Microsoft Store in your desktop and have it up and running.

1. Open Ubuntu and use the below commands to install packer

```
$ export VER="1.4.1"
$ wget
https://releases.hashicorp.com/packer/${VER}/packer_${VER}_linux_amd64.zip
$ Unzip packer_${ver}_linux_amd64.zip
$ sudo mv packer /usr/local/bin
```

2. Verify the installation by running below command

```
$ packer -v
```

The output will give you the version number:

```
root@HIBOCL72613:/home/sroy2# packer -v
1.4.1
```

3. Create a file **packer.json** and copy the following contents:

```

{
  "variables": {
    "aws_access_key": "",
    "aws_secret_key": "",
    "region": "eu-central-1"
  },
  "builders": [
    {
      "access_key": "{{user `aws_access_key`}}",
      "ami_name": "miniprj_xxxxx_ami",
      "instance_type": "t2.micro",
      "region": "eu-central-1",
      "secret_key": "{{user `aws_secret_key`}}",
      "source_ami_filter": {
        "filters": {
          "virtualization-type": "hvm",
          "name": "ubuntu/images/*ubuntu-xenial-16.04-amd64-server-*",
          "root-device-type": "ebs"
        },
        "owners": ["099720109477"],
        "most_recent": true
      },
      "ssh_username": "ubuntu",
      "type": "amazon-ebs"
    }
  ],
  "provisioners": [
    {
      "type": "shell",
      "script": "./install-tools.sh"
    }
  ]
}

```

**\*\*N.B.** In below packer.json, add your AWS account credentials in "aws\_access\_key" and "aws\_secret\_key" Give the "ami\_name" replace xxxxx with your name to uniquely identify each ami.

You can see inside packer json there are two parts: builders and provisioners. Builders section is used to form the metadata of the ami while provisioners section is used to install packages and run other commands for creating the objects of the ami. Here in this case we are running the **install-tools.sh** command to install ansible, Jenkins and awscli inside the image.



#### 4. Create the file install-tool.sh

```
#!/bin/bash
sudo apt-get update
sudo apt-get install software-properties-common -y
sudo apt-add-repository ppa:ansible/ansible
sudo apt-get update
sudo apt-get install ansible -y
sudo apt-get install openjdk-8-jdk -y
wget -q -O - https://pkg.jenkins.io/debian/jenkins.io.key | sudo apt-key add -
sudo sh -c 'echo deb http://pkg.jenkins.io/debian-stable binary/ >
/etc/apt/sources.list.d/jenkins.list'
sudo sh -c 'echo deb http://pkg.jenkins.io/debian-stable binary/ >
/etc/apt/sources.list.d/jenkins.list'
sudo apt-get update
sudo apt-get install jenkins -y
sudo apt-get install awscli -y
```

#### 5. Validate the packer.json file

```
$ packer validate packer.json
```

You should see the output:

```
root@HIB0CL72613:/home/sroy2# packer validate packer.json
Template validated successfully.
```

This indicates that the packer file is valid and you are good to go to create the ami.

#### 6. Run the following command to create the ami

```
$ packer build packer.json
```

This will create a new image under the AMIs section of your EC2 service. You'll see an output ending like this:

```

amazon-ebs: Preparing to unpack .../jenkins_2.164.3_all.deb ...
amazon-ebs: Unpacking jenkins (2.164.3) ...
amazon-ebs: Processing triggers for man-db (2.7.5-1) ...
amazon-ebs: Processing triggers for ureadahead (0.100.0-19.1) ...
amazon-ebs: Processing triggers for systemd (229-4ubuntu21.21) ...
amazon-ebs: Setting up daemon (0.6.4-1) ...
amazon-ebs: Setting up jenkins (2.164.3) ...
amazon-ebs: Processing triggers for ureadahead (0.100.0-19.1) ...
amazon-ebs: Processing triggers for systemd (229-4ubuntu21.21) ...
==> amazon-ebs: Stopping the source instance...
amazon-ebs: Stopping instance
==> amazon-ebs: Waiting for the instance to stop...
==> amazon-ebs: Creating AMI miniprj_soham_ami from instance i-0ecff67a5770bf2ea
amazon-ebs: AMI: ami-01983d29db074a938
==> amazon-ebs: Waiting for AMI to become ready...
==> amazon-ebs: Terminating the source AWS instance...
==> amazon-ebs: Cleaning up any extra volumes...
==> amazon-ebs: No volumes to clean up, skipping
==> amazon-ebs: Deleting temporary security group...
==> amazon-ebs: Deleting temporary keypair...
Build 'amazon-ebs' finished.

==> Builds finished. The artifacts of successful builds are:
--> amazon-ebs: AMIs were created:
eu-central-1: ami-01983d29db074a938

```

## 7. Check your image on the AWS console

| Name                     | AMI Name          | AMI ID                 | Source           | Owner        | Visibility | Status    | Creation Date                  | Platform    |
|--------------------------|-------------------|------------------------|------------------|--------------|------------|-----------|--------------------------------|-------------|
| <input type="checkbox"/> | deepika-ami       | ami-059508b37b1dc4935  | 857026751867/... | 857026751867 | Private    | available | May 27, 2019 at 5:33:11 PM ... | Other Linux |
| <input type="checkbox"/> | deepika1-ami      | ami-01058812c00592ff4  | 857026751867/... | 857026751867 | Private    | available | May 28, 2019 at 4:44:38 PM ... | Other Linux |
| <input type="checkbox"/> | deepthi-ami       | ami-096c1759ba06285c1  | 857026751867/... | 857026751867 | Private    | available | May 24, 2019 at 1:38:09 PM ... | Other Linux |
| <input type="checkbox"/> | deepthi-ami1      | ami-03339ac7471e2d04c  | 857026751867/... | 857026751867 | Private    | available | May 24, 2019 at 2:28:15 PM ... | Other Linux |
| <input type="checkbox"/> | devopsAMI         | ami-02fa81f4779c293a3  | 857026751867/... | 857026751867 | Private    | available | May 15, 2019 at 5:29:56 PM ... | Other Linux |
| <input type="checkbox"/> | first-ami         | ami-0cafa843f51b89bd7  | 857026751867/... | 857026751867 | Private    | available | May 29, 2019 at 6:23:56 PM ... | Other Linux |
| <input type="checkbox"/> | miniprj_soham_ami | ami-01983d29db074a938  | 857026751867/... | 857026751867 | Private    | available | May 30, 2019 at 8:33:01 PM ... | Other Linux |
| <input type="checkbox"/> | miniprj_xyz_ami   | ami-0cc0d4ff0d06e515e5 | 857026751867/... | 857026751867 | Private    | available | May 29, 2019 at 5:30:09 PM ... | Other Linux |
| <input type="checkbox"/> | miniprj_yyyy_ami  | ami-06ff5663c194c8bae  | 857026751867/... | 857026751867 | Private    | available | May 29, 2019 at 4:49:42 PM ... | Other Linux |

## Summary of the exercise:

You have learnt

- Step by step procedure to install packer
- Create an ubuntu image with packer

### Assignment 3: Use terraform to create server using custom AMI (30 mins)

1. Install terraform on your local Ubuntu

```
$ sudo apt-get install terraform
```

2. Generate a key-pair using the below command and provide a filename ( ex: ./new-key-pair.pem):

```
$ ssh-keygen -t rsa
```

The output should look like this:

```
root@HIBOCL72613:/home/sroy2# ssh-keygen -t rsa
Generating public/private rsa key pair.
Enter file in which to save the key (/root/.ssh/id_rsa): ./new_key_2.pem
Enter passphrase (empty for no passphrase):
Enter same passphrase again:
Your identification has been saved in ./new_key_2.pem.
Your public key has been saved in ./new_key_2.pem.pub.
The key fingerprint is:
SHA256:7... root@HIBOCL72613
The key's randomart image is:
+---[RSA 2048]-----+
| .oB=o .o          |
| .*o+ o+ E         |
| ..+ + o@ .        |
| o * o +* +        |
| *.o.+ . S         |
| o...ooo+          |
| . o.++o.          |
| . ..O..           |
| .                 |
+---[SHA256]-----+
```

You should see two files ( 1 pem file and 1 pem.pub file).

```
root@HIBOCL72613:/home/sroy2# ls -lth new_key*
-rw----- 1 root root 1.7K May 30 23:19 new_key_2.pem
-rw-r--r-- 1 root root 398 May 30 23:19 new_key_2.pem.pub
```

### 3. Create a file terraform.tf

```

provider "aws" {
    access_key = "<your access key>"
    secret_key = "< your secret access key>"
    region = "eu-central-1"
}

resource "aws_key_pair" "new_key" {
    key_name = "<key_pair_name>"
    public_key = "<content of you public key> "
}

resource "aws_instance" "first_instance" {
    ami = "< ami id of the mage you created with packer>"
    instance_type = "t2.micro"
    key_name = "${aws_key_pair.new_key.key_name}"
    tags {
        Name = "<name of the instance>"
    }
}

```

### 4. Use the below command to check for validation of the tf file

```
$ terraform plan
```

This should give you the total number of resources to be created in the AWS cloud ( in this case 1 instance and 1 key file).

The command output should end like this:

```
Plan: 2 to add, 0 to change, 2 to destroy.
```

```
-----  
Note: You didn't specify an "-out" parameter to save this plan, so Terraform  
can't guarantee that exactly these actions will be performed if  
"terraform apply" is subsequently run.
```

5. Now you are good to go to create your first server on AWS cloud via terraform. Use the below command to perform the same:

```
$ terraform plan
```

You will need to press yes to continue

The final output will look like this:

```

aws_key_pair.new_key: Creation complete after 0s (ID: new_key_2)
aws_instance.first_instance: Creating...
  ami: "" => "ami-01983d29db074a938"
  arn: "" => "<computed>"
  associate_public_ip_address: "" => "<computed>"
  availability_zone: "" => "<computed>"
  cpu_core_count: "" => "<computed>"
  cpu_threads_per_core: "" => "<computed>"
  ebs_block_device.#: "" => "<computed>"
  ephemeral_block_device.#: "" => "<computed>"
  get_password_data: "" => "false"
  host_id: "" => "<computed>"
  instance_state: "" => "<computed>"
  instance_type: "" => "t2.micro"
  ipv6_address_count: "" => "<computed>"
  ipv6_addresses.#: "" => "<computed>"
  key_name: "" => "new_key_2"
  network_interface.#: "" => "<computed>"
  network_interface_id: "" => "<computed>"
  password_data: "" => "<computed>"
  placement_group: "" => "<computed>"
  primary_network_interface_id: "" => "<computed>"
  private_dns: "" => "<computed>"
  private_ip: "" => "<computed>"
  public_dns: "" => "<computed>"
  public_ip: "" => "<computed>"
  root_block_device.#: "" => "<computed>"
  security_groups.#: "" => "<computed>"
  source_dest_check: "" => "true"
  subnet_id: "" => "<computed>"
  tags.%: "" => "1"
  tags.Name: "" => "terraform-soham-instance"
  tenancy: "" => "<computed>"
  volume_tags.%: "" => "<computed>"
  vpc_security_group_ids.#: "" => "<computed>"
aws_instance.first_instance: Still creating... (10s elapsed)
aws_instance.first_instance: Still creating... (20s elapsed)
aws_instance.first_instance: Still creating... (30s elapsed)
aws_instance.first_instance: Creation complete after 34s (ID: i-048480db98e91e00b)

Apply complete! Resources: 2 added, 0 changed, 2 destroyed.

```

You might not see any resource destroyed in your case.

- Once the instance is created, log in to that instance and check if Jenkins, aws cli, Ansible are installed.

```
$ ssh -i "<your key name>" ubuntu@<public ip or dns>
```

- Now you are inside the server use the command one by one to check whether ansible and Jenkins have been installed or not:

```
$ ansible --version
```

```
$sudo systemctl status Jenkins
```

You will be able to see ansible version and Jenkins status active.

```
ubuntu@ip-172-31-40-218:~$ ansible --version
ansible 2.8.0
  config file = /etc/ansible/ansible.cfg
  configured module search path = [u'/home/ubuntu/.ansible/plugins/modules', u'/usr/share/ansible/plugins/modules']
  ansible python module location = /usr/lib/python2.7/dist-packages/ansible
  executable location = /usr/bin/ansible
  python version = 2.7.12 (default, Nov 12 2018, 14:36:49) [GCC 5.4.0 20160609]
ubuntu@ip-172-31-40-218:~$
ubuntu@ip-172-31-40-218:~$
ubuntu@ip-172-31-40-218:~$ sudo systemctl status jenkins
● jenkins.service - LSB: Start Jenkins at boot time
   Loaded: loaded (/etc/init.d/jenkins; bad; vendor preset: enabled)
   Active: active (exited) since Thu 2019-05-30 18:39:56 UTC; 7min ago
     Docs: man:systemd-sysv-generator(8)
  Process: 1263 ExecStart=/etc/init.d/jenkins start (code=exited, status=0/SUCCESS)
    Tasks: 0
   Memory: 0B
      CPU: 0
May 30 18:39:53 ip-172-31-40-218 systemd[1]: Starting LSB: Start Jenkins at boot time...
May 30 18:39:55 ip-172-31-40-218 jenkins[1263]: Correct java version found
May 30 18:39:55 ip-172-31-40-218 jenkins[1263]: * Starting Jenkins Automation Server jenkins
May 30 18:39:55 ip-172-31-40-218 su[1390]: Successful su for jenkins by root
May 30 18:39:55 ip-172-31-40-218 su[1390]: + ??? root:jenkins
May 30 18:39:55 ip-172-31-40-218 su[1390]: pam_unix(su:session): session opened for user jenkins by (uid=0)
May 30 18:39:56 ip-172-31-40-218 jenkins[1263]: ...done.
May 30 18:39:56 ip-172-31-40-218 systemd[1]: Started LSB: Start Jenkins at boot time.
ubuntu@ip-172-31-40-218:~$
```

8. We will use Jenkins later so we'll configure the aws credentials in 'jenkins' user:

```
$ sudo su - jenkins
```

```
$ aws configure
```

You'll be needed to ask the access and secret access key is here. Please provide the same:

```
ubuntu@ip-172-31-40-218:~$ aws configure
AWS Access Key ID [None]: AKIAIOSFODNN7EXAMPLE
AWS Secret Access Key [None]: wJalrXU3WhelbkZVTGn5d6P5Jue8
Default region name [None]: eu-central-1
Default output format [None]: json
```

### Summary of the exercise:

You have learnt

- Step by step procedure to install terraform
- Use terraform to create server on AWS
- Configuring aws keys on server

## Assignment 4: Use ansible-pull to download the index.html from Github (10 mins)

**Pre-requisite:** Create the following directory in your instance.

`/var/projects/ansible-git/`

1. First of all you need to create a key-based authentication between github and your server.

**Go to github.com -> settings -> SSH and GPG keys -> New SSH Key -> paste the public key here**

2. Store the private key in your server ( ex: new\_key\_2.pem).
3. Go back to the instance and run the following commands:

```
$ url='your git url'
$ checkout='master'
$ directory='/var/projects/ansible-git'
$ logfile='/var/log/ansible.log'
$ ssh_file='<your key name>'
$ sudo ansible-pull -o -C ${checkout} -d ${directory} -i localhost:${directory}/inventory -U ${url} --
accept-host-key $ssh_file 2>&1
```



Go to `/var/projects/ansible-git'` and list the files to see `"index.html"` in it.

### Summary of the exercise:

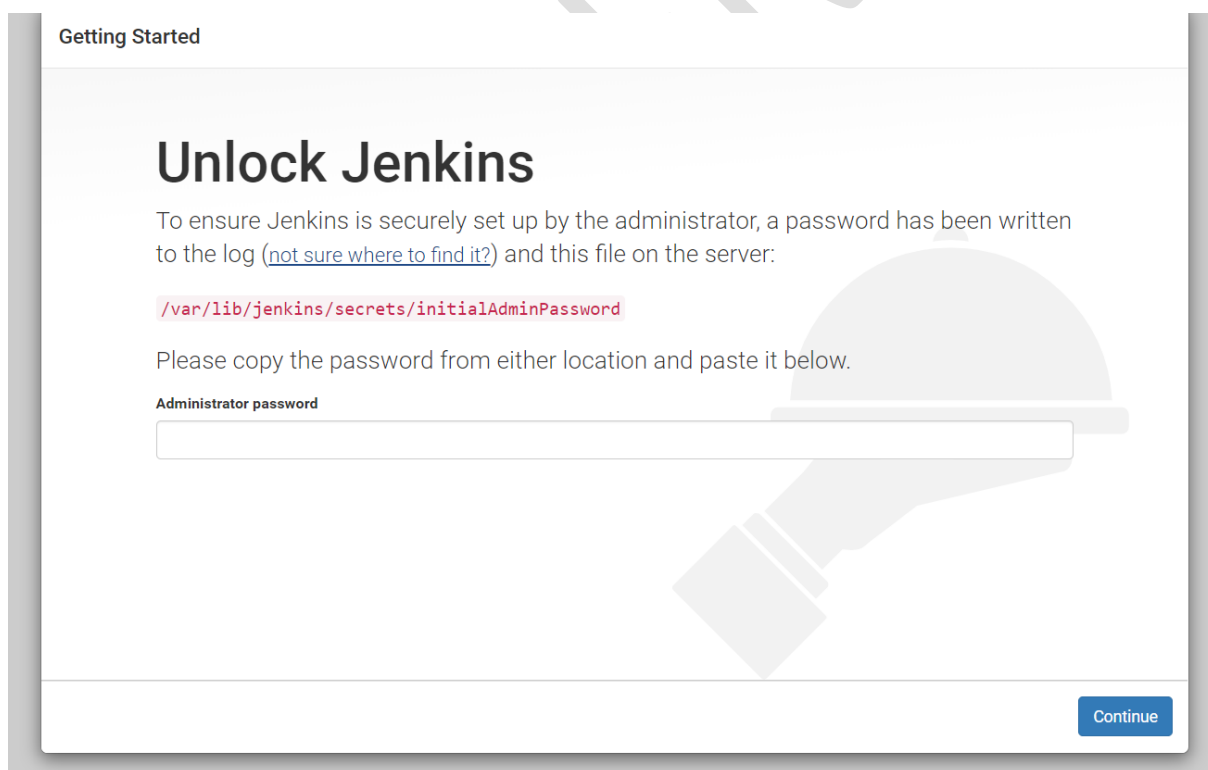
You have learnt

- How to connect from server to github.
- Use ansible-pull to install file from github.

## Assignment 5: Use Jenkins to copy the file to S3 (20 mins)

**Pre-requisite:** S3 bucket is present to push the file

1. Open the url :: <http://<ip>:8080> ( Jenkins by default is hosted on port 8080)  
This will show you a page like the below snippet:



2. Go to your server to fetch the password:

```
$sudo cat /var/lib/jenkins/secrets/initialAdminPassword
```

Insert the password and click continue.

3. As of now on the next page you can select install suggested plugin, later at any point of time you can add any plugin. This will take some time.  
Create admin user:

#### Getting Started

## Create First Admin User

Username:

Password:

Confirm password:

Full name:

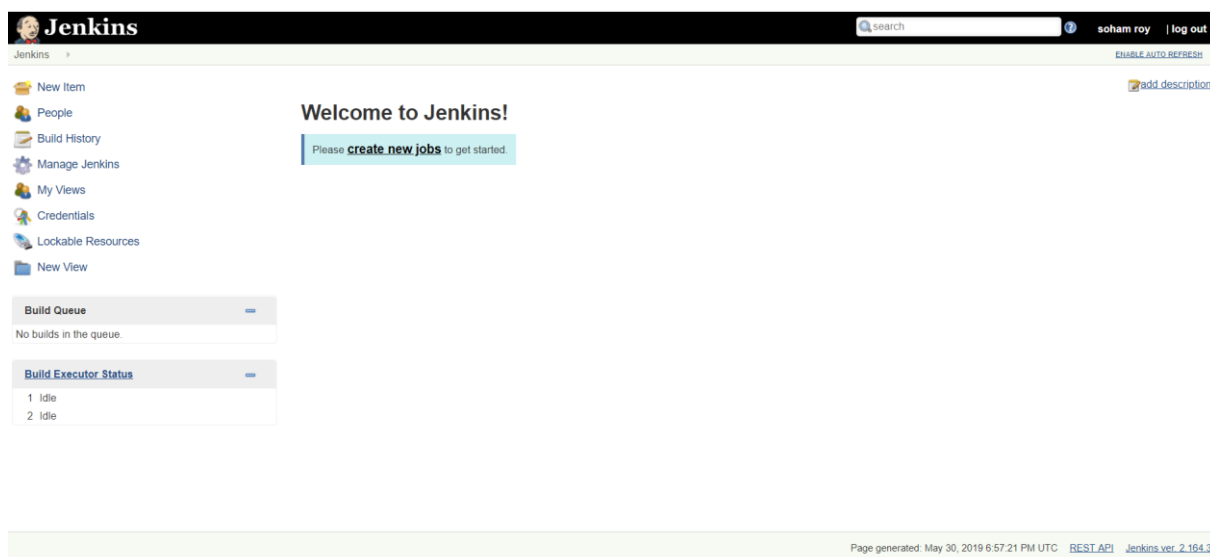
E-mail address:

Jenkins 2.164.3

[Continue as admin](#)

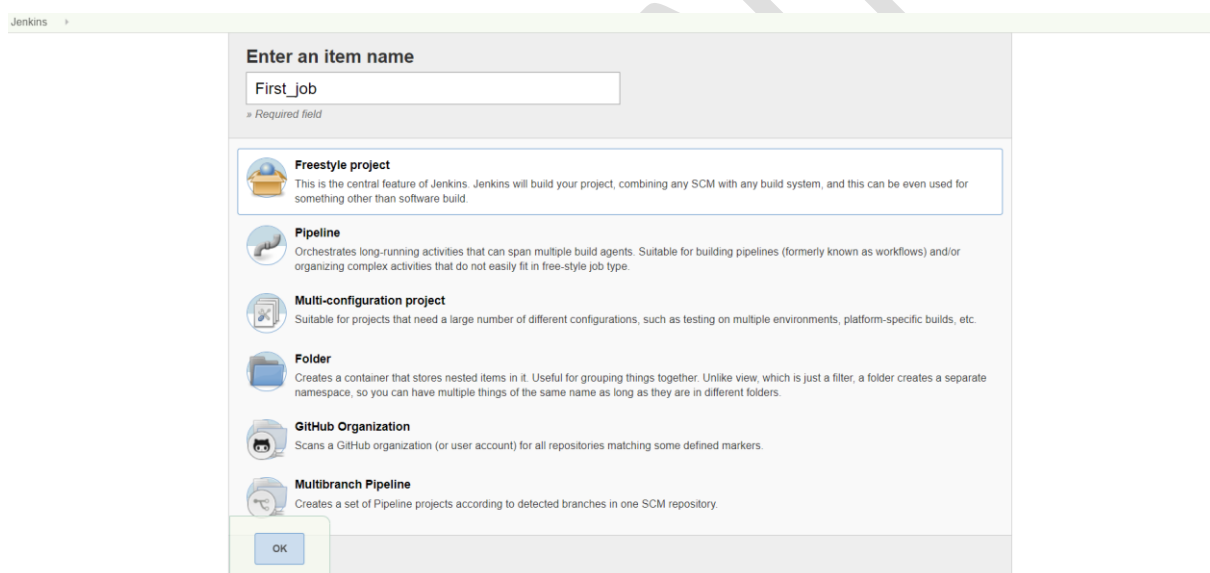
[Save and Continue](#)

The next pages will show your Jenkins url and then you are ready to use Jenkins.



The screenshot shows the Jenkins dashboard. At the top, there's a header with the Jenkins logo, a search bar, and user information (soham roy | log out). Below the header, on the left, is a sidebar with navigation links: New Item, People, Build History, Manage Jenkins, My Views, Credentials, Lockable Resources, and New View. The main area displays a 'Welcome to Jenkins!' message with a button to 'create new jobs to get started'. Below this, there are two sections: 'Build Queue' showing 'No builds in the queue.' and 'Build Executor Status' showing two 'Idle' executors. At the bottom right, there's a footer with page generation info: 'Page generated: May 30, 2019 6:57:21 PM UTC', links for 'REST API' and 'Jenkins ver. 2.164.3', and a 'add description' link.

#### 4. Create a new job. Give a job name and select Freestyle project:



The screenshot shows the 'Enter an item name' dialog in Jenkins. The 'First\_job' name is entered in the text field. Below the field, there's a list of project types with their descriptions:

- 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.
- Folder**: Creates a container that stores nested items in it. Useful for grouping things together. Unlike view, which is just a filter, a folder creates a separate namespace, so you can have multiple things of the same name as long as they are in different folders.
- GitHub Organization**: Scans a GitHub organization (or user account) for all repositories matching some defined markers.
- Multibranch Pipeline**: Creates a set of Pipeline projects according to detected branches in one SCM repository.

An 'OK' button is visible at the bottom left of the dialog.

## 5. Add a description:

Jenkins » First Job »

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

Description: This is a test job

[Plain text] [Preview](#)

- ☐ Discard old builds
- ☐ GitHub project
- ☐ This build requires lockable resources
- ☐ This project is parameterized
- ☐ Throttle builds
- ☐ Disable this project
- ☐ Execute concurrent builds if necessary

[Advanced...](#)

**Source Code Management**

☒ None

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

I have already created a bucket `s3://voicedep-devops`. If needed you can create a new one.

Go below, in the build section select execute shell and add the command:

```
aws s3 cp <path-to-file in server> <s3 url ( ex s3://voicedep-devops) >
```

Jenkins » First Job »

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

- ☐ Add timestamps to the Console Output
- ☐ Inspect build log for published Gradle build scans
- ☐ With Ant

**Build**

**Execute shell**

Command: `aws s3 cp <path-to-file in server> <s3://voicedep-devops>`

[See the list of available environment variables](#)

[Advanced...](#)

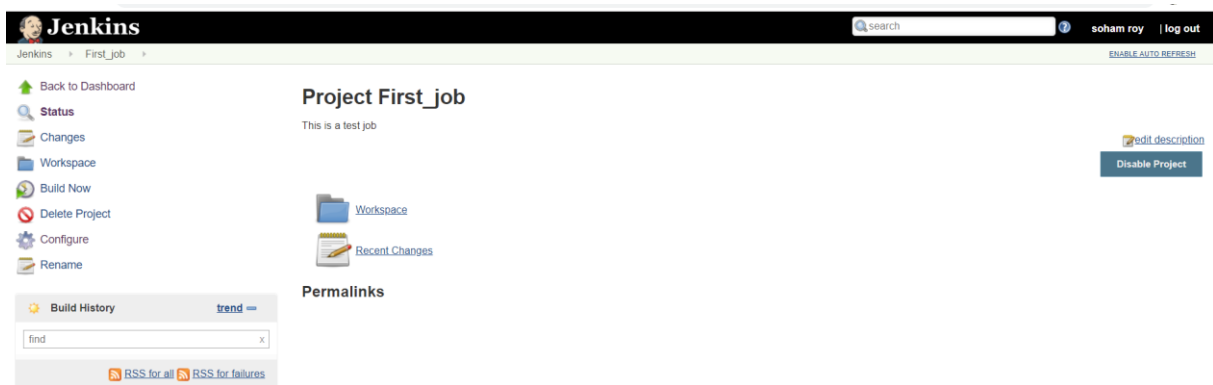
**Post-build Actions**

[Add post-build action](#)

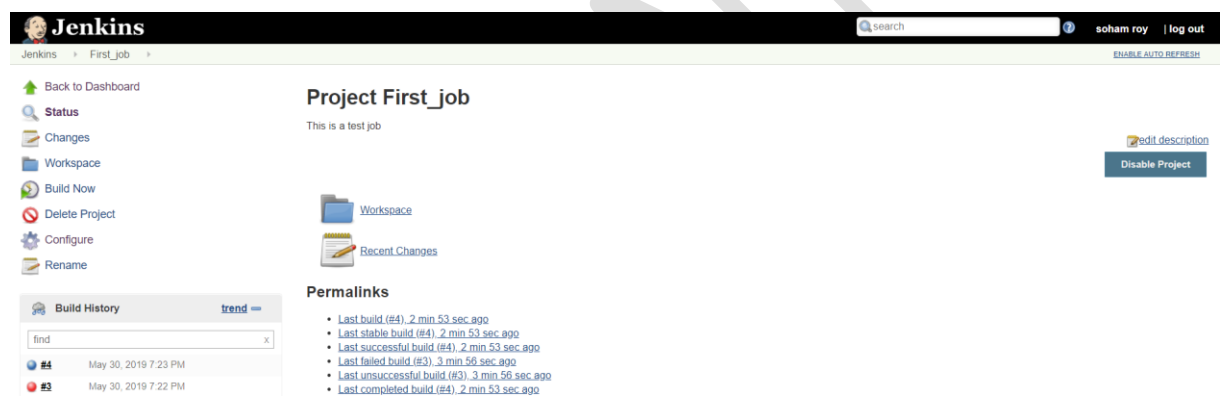
[Save](#) [Apply](#)

Click on save. You are now ready to run your first jenkins job.

Harman Connected Services



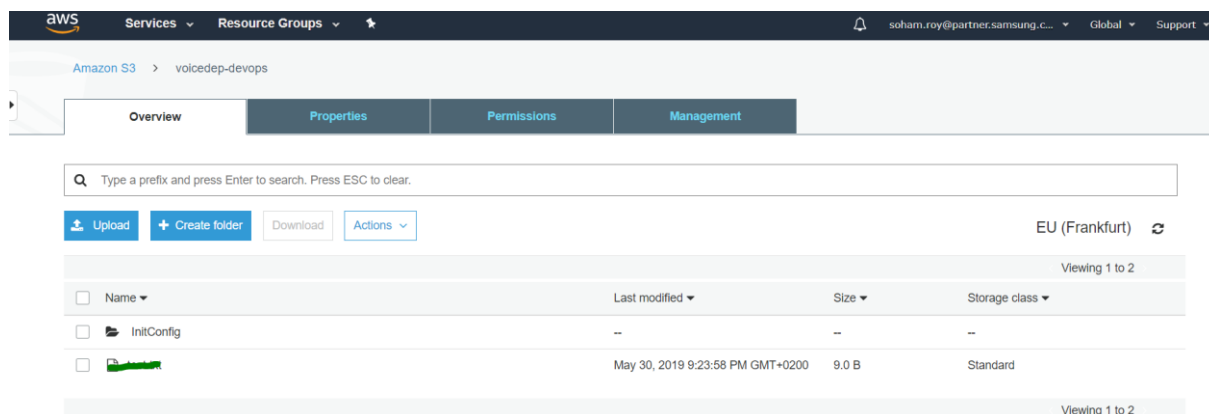
6. Click on build now, the build number will come below:



7. Click on the latest build number and click on console output. You can check your status there  
**Finished: SUCCESS**



8. To double check you can see your file on S3 as well.



Amazon S3 > voicedep-devops


Overview Properties Permissions Management

Q Type a prefix and press Enter to search. Press ESC to clear.

Upload Create folder Download Actions

EU (Frankfurt)

Viewing 1 to 2

| Name  | Last modified                    | Size  | Storage class |
|---|----------------------------------|-------|---------------|
| InitConfig  | --                               | --    | --            |
|  | May 30, 2019 9:23:58 PM GMT+0200 | 9.0 B | Standard      |

Viewing 1 to 2

### Summary of the exercise:

You have learnt

- How to configure and create user in Jenkins.
- Create and execute job in Jenkins.