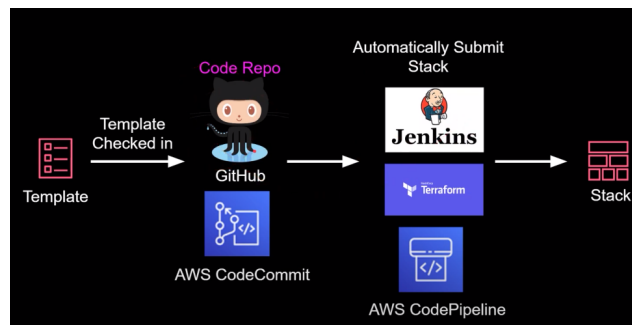


Run Infra as Code with Jenkins



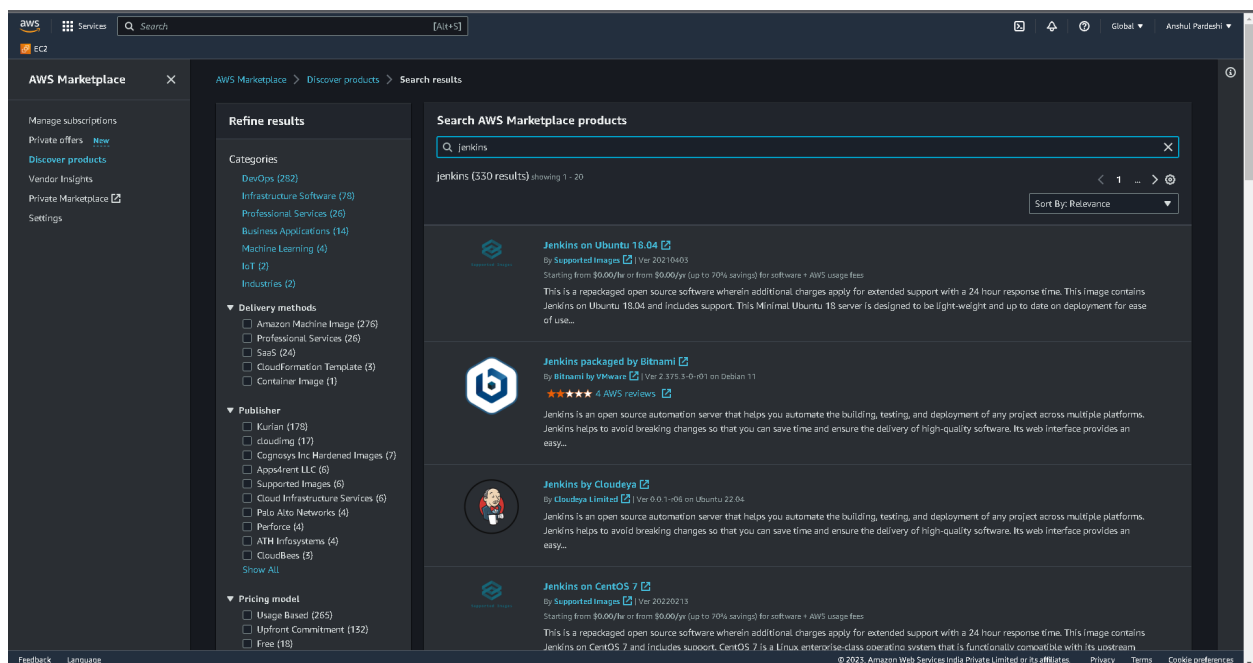
Let us distribute this process into two parts.

Part-1: We are going to install Jenkins on EC2 with one click. Then we are going to install plugins in Jenkins. Finally we will run a cloudformation from github in Jenkins.

Part-2: We are going to install AWS CLI into jenkins, then learn about Jenkinsfile and then create a pipeline to deploy cloudformation from GIT.

Part:1

Search for 'Jenkins' in AWS Marketplace products. Choose an AMI according to the OS you want. I chose one with Ubuntu in it.



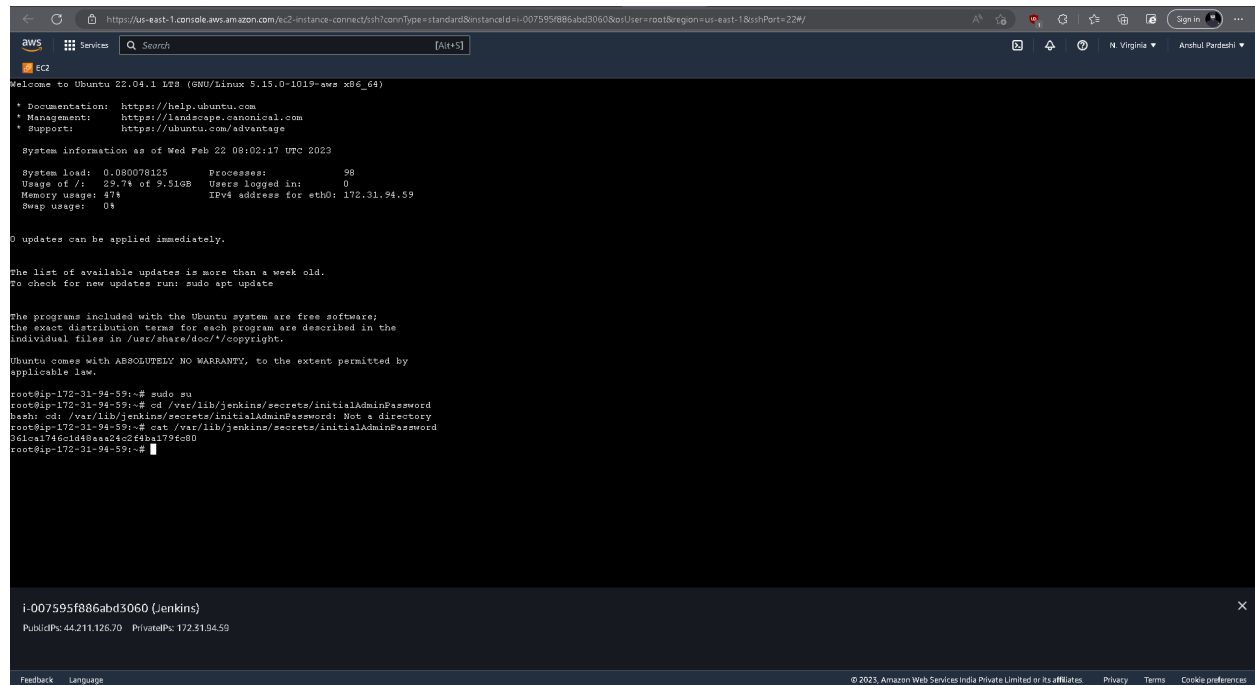
Now quickly choose the required settings and instance configuration that you need and launch it. We chose t2.micro as it is free.

The screenshot shows the AWS Marketplace page for 'Jenkins by Cloudeya'. The page has a dark header with the AWS Marketplace logo and navigation links. Below the header, there's a section for 'Launch this software' with a green box containing a congratulatory message and a link to view launch configuration details. Below this, there's a 'Configuration details' section showing the fulfillment option (64-bit (x86) Amazon Machine Image (AMI)), software version (0.0.1+06 on Ubuntu 22.04), and region (US East (N. Virginia)). There's a 'Choose Action' dropdown menu and a 'Launch' button. At the bottom, there are links to AWS Marketplace on Twitter, AWS Marketplace Blog, and RSS Feed.

Our EC2 instance has started to spin up. Rename it to avoid confusion.

The screenshot shows the AWS Management Console with the 'Instances' page selected. A table lists the instances, showing one instance with ID 'i-007595f886abd3060' in the 'Running' state. Below the table, the 'Instance details' for 'i-007595f886abd3060' are displayed. The details include the platform (Linux/UNIX), AMI ID (ami-0d585ae13f1b98dda), AMI name (jenkins_mcrsystems_11092022-5e2b01d2-ddbe-43b3-bb89-a8374d46f9f2), launch time (Wed Feb 22 2023 13:23:51 GMT+0530 (India Standard Time) (5 minutes)), lifecycle (normal), key pair name (for-MERN), and monitoring (disabled).

SSH into this instance. To get password to login into jenkins:
cat /var/lib/jenkins/secrets/initialAdminPassword



```
https://us-east-1.console.aws.amazon.com/ec2-instance-connect/sh?connType=standard&instanceId=i-007595f886abd3060&osUser=root&region=us-east-1&sshPort=22#/?
AWS Services Search [Alt+S]
EC2
Welcome to Ubuntu 22.04.1 LTS (GNU/Linux 5.15.0-1019-aws x86_64)
 * Documentation:  https://help.ubuntu.com
 * Management:    https://landscape.canonical.com
 * Support:        https://ubuntu.com/advantage

System information as of Wed Feb 22 08:02:17 UTC 2023

System load: 0.000078125   Processes: 98
Usage of /: 29.7% of 9.51GB   Users logged in: 0
Memory usage: 47%          IPv4 address for eth0: 172.31.94.59
Swap usage: 0%

0 updates can be applied immediately.

The list of available updates is more than a week old.
To check for new updates run: sudo apt update

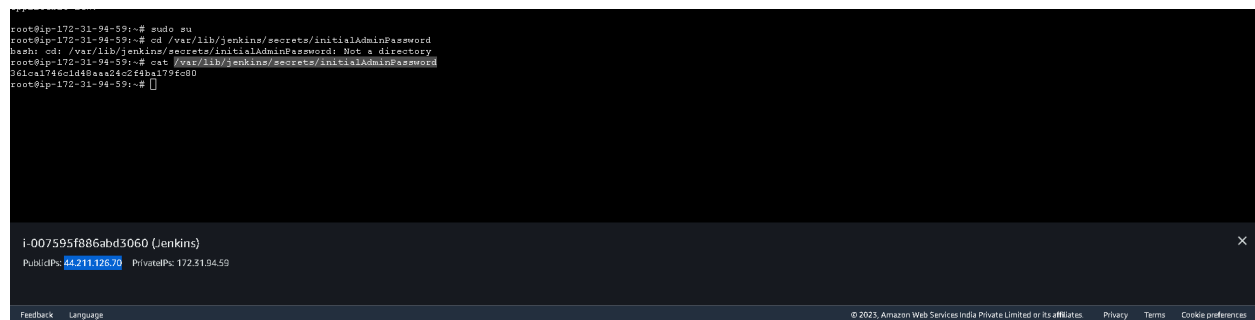
The programs included with the Ubuntu system are free software;
the exact distribution terms for each program are described in the
individual files in /usr/share/doc/*/*copyright.

Ubuntu comes with ABSOLUTELY NO WARRANTY, to the extent permitted by
applicable law.

root@ip-172-31-94-59:~# sudo su
root@ip-172-31-94-59:~# cd /var/lib/jenkins/secrets/initialAdminPassword
bash: cd: /var/lib/jenkins/secrets/initialAdminPassword: Not a directory
root@ip-172-31-94-59:~# cat /var/lib/jenkins/secrets/initialAdminPassword
D6lca1746c1d4d8aa24c2f4ba179fc80
root@ip-172-31-94-59:~#

i-007595f886abd3060 (jenkins)
PublicIP: 44.211.126.70 PrivateIP: 172.31.94.59
```

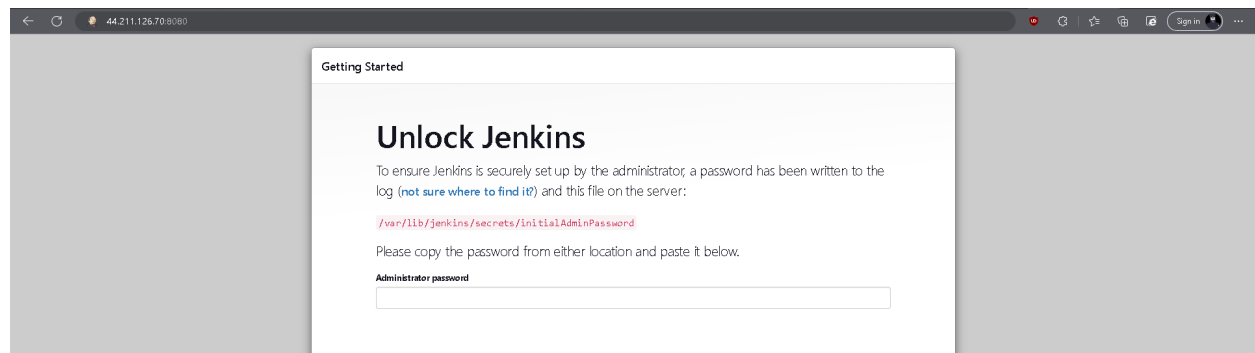
Copy the public ip of instance.



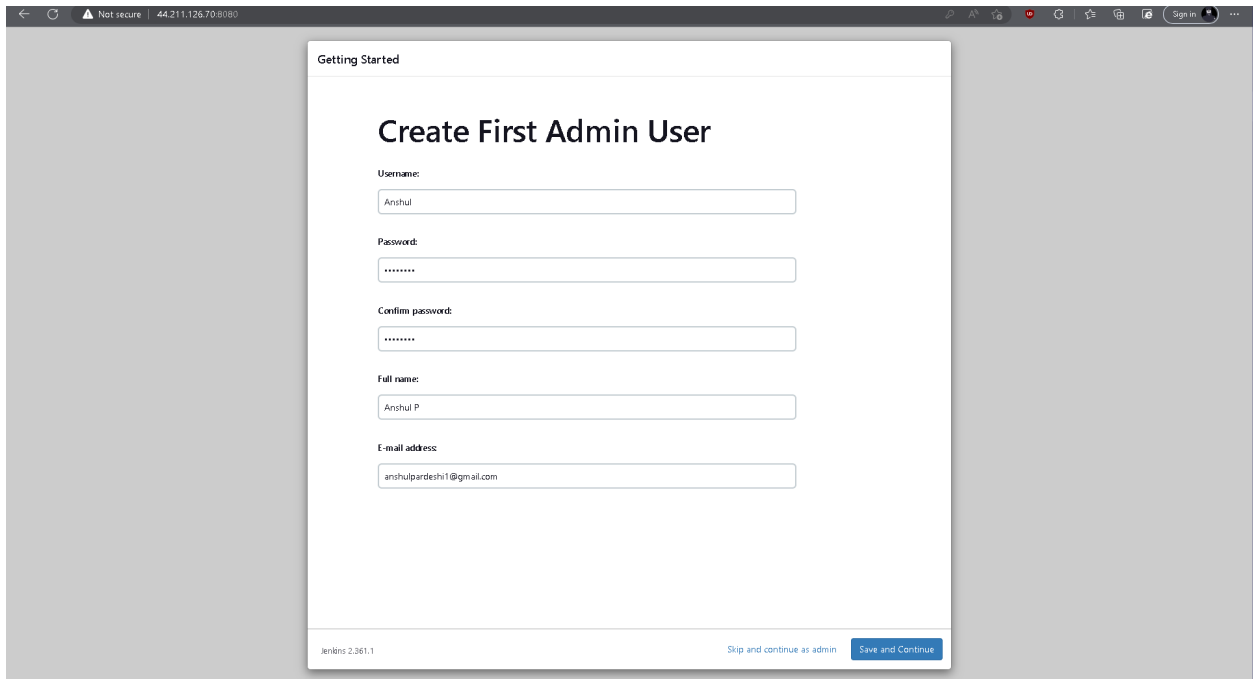
```
root@ip-172-31-94-59:~# sudo su
root@ip-172-31-94-59:~# cd /var/lib/jenkins/secrets/initialAdminPassword
bash: cd: /var/lib/jenkins/secrets/initialAdminPassword: Not a directory
root@ip-172-31-94-59:~# cat /var/lib/jenkins/secrets/initialAdminPassword
D6lca1746c1d4d8aa24c2f4ba179fc80
root@ip-172-31-94-59:~#

i-007595f886abd3060 (jenkins)
PublicIP: 44.211.126.70 PrivateIP: 172.31.94.59
```

And paste it on the browser following the 8080 port. That is ip_address:8080.
Enter the password here that you got after using the cat command in the above steps.



Later you will be prompted to install the plugins page. We will install default plugins for now and install the plugins we need later.
After the plugins are installed. Enter username password and other details.

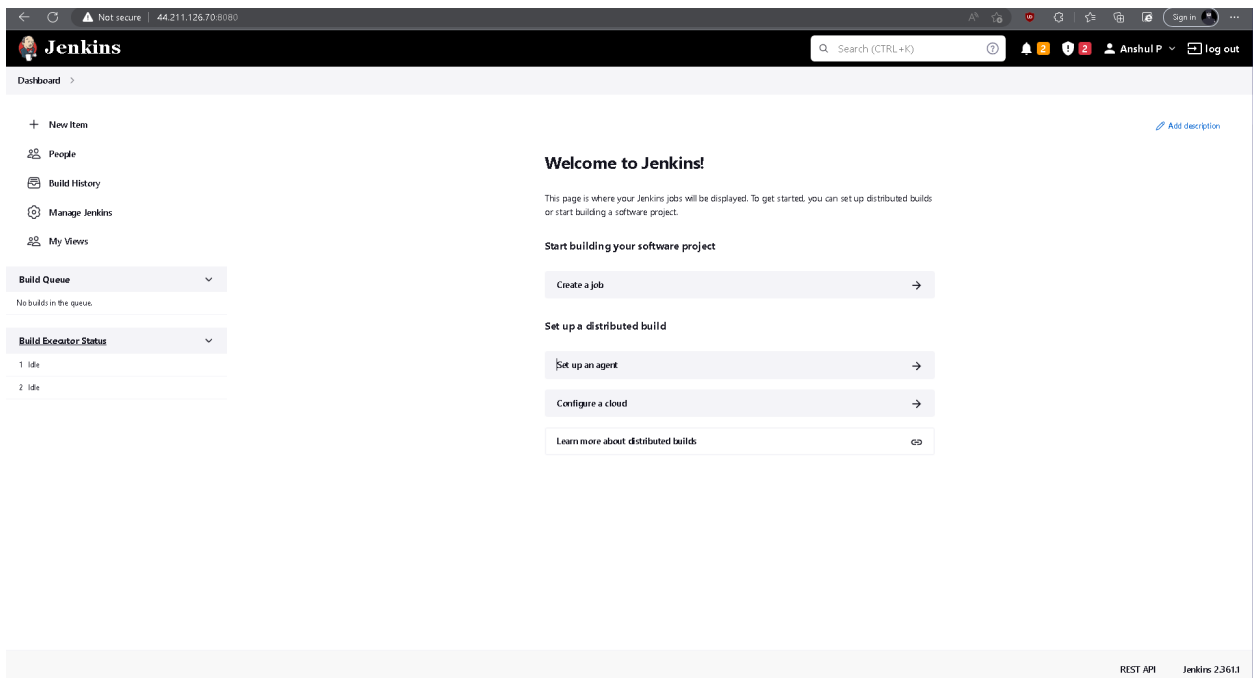


The screenshot shows the Jenkins 'Getting Started' page with the 'Create First Admin User' form. The form fields are filled with the following information:

- Username: Anshul
- Password: (masked with dots)
- Confirm password: (masked with dots)
- Full name: Anshul P
- E-mail address: anshulpardeshi1@gmail.com

At the bottom of the form, there are two buttons: 'Skip and continue as admin' and 'Save and Continue'.

Now you will be prompted to this page.



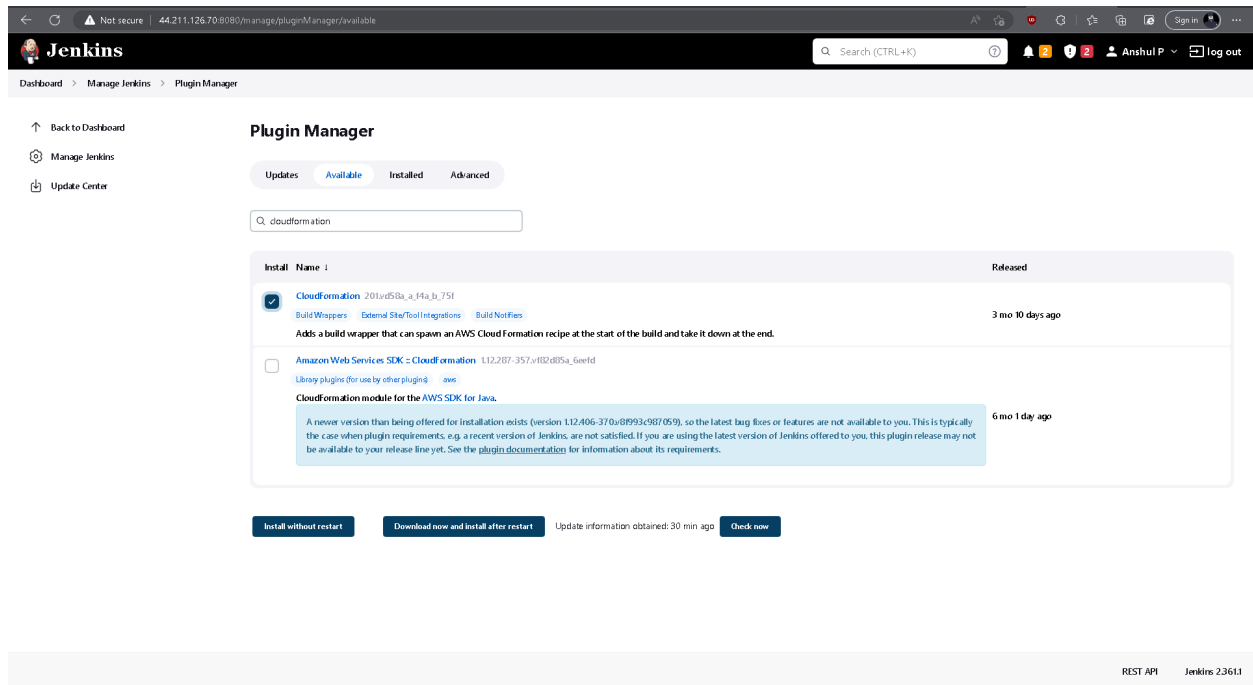
The screenshot shows the Jenkins Dashboard page. The top navigation bar includes the Jenkins logo, a search bar, and user information (Anshul P, log out). The left sidebar contains links to 'New Item', 'People', 'Build History', 'Manage Jenkins', and 'My Views'. The main content area displays a 'Welcome to Jenkins!' message and a 'Start building your software project' section with the following options:

- Create a job
- Set up a distributed build
 - Set up an agent
 - Configure a cloud
 - Learn more about distributed builds

The 'Build Queue' section shows 'No builds in the queue'. The 'Build Executor Status' section shows two executors in an 'Idle' state.

To install the plugin that we need.

Click on Manage Jenkins>>Manage Plugins>> Available and then search for 'Cloudformation'. Install this plugin.



Now let us run cloud formation from github into jenkins.

For this we have created a file in json format. Let us look into it first.

```
{
  "AWSTemplateFormatVersion": "2010-09-09",
  "Resources": {
    "S3Bucket": {
      "Type": "AWS::S3::Bucket"
    }
  },
  "Outputs": {
    "BucketName": {
      "Value": {
        "Ref": "S3Bucket"
      },
      "Description": "Name of the sample Amazon S3 bucket."
    }
  }
}
```

Here's what the different parts of the template do:

"AWSTemplateFormatVersion": This specifies the version of the CloudFormation template format being used. In this case, it's the 2010-09-09 version.

"Resources": This section contains a list of resources that will be created when the CloudFormation stack is deployed. In this template, there is only one resource, an S3 bucket.

"S3 Bucket": This is the logical ID of the S3 bucket resource. The "Type" attribute specifies the type of AWS resource to create, in this case, an S3 bucket.

"Outputs": This section contains a list of output values that can be retrieved from the CloudFormation stack after it has been created.

"BucketName": This is the logical ID of the output value. The "Value" attribute is set to the "Ref" function, which retrieves the value of the S3 bucket's logical ID. This means that the output will contain the name of the S3 bucket that was created by this CloudFormation stack. The "Description" attribute provides a description of the output value.

Overall, this CloudFormation template creates an S3 bucket resource and outputs the name of the bucket. When the CloudFormation stack is created from this template, the S3 bucket will be provisioned and the bucket name will be available as an output value.

Let us get back to Jenkins and create a freestyle project using this json file.

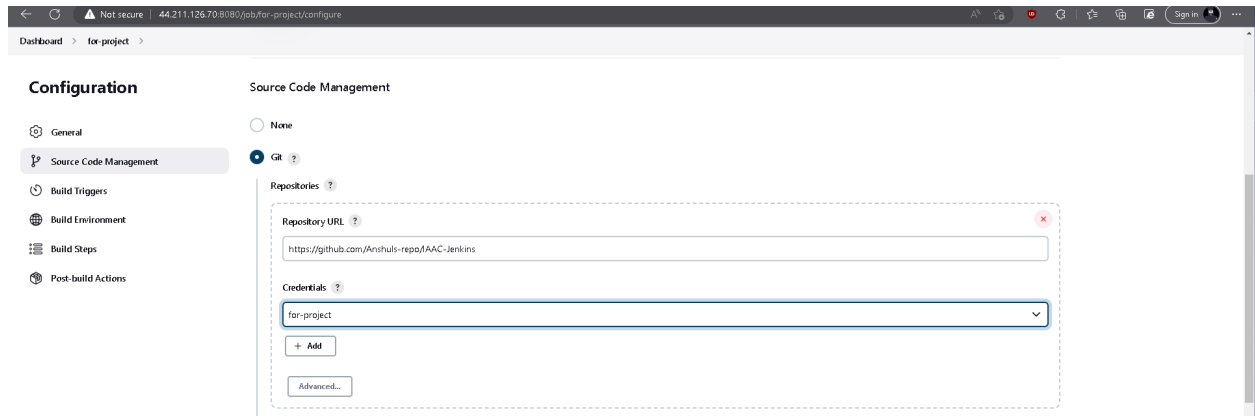
Click on create job>>Freestyle Project and name your project.

The screenshot shows the Jenkins Dashboard. On the left, there's a sidebar with links: New Item, People, Build History, Manage Jenkins, and My Views. Below these are two sections: 'Build Queue' (No builds in the queue) and 'Build Executor Status' (1 idle, 2 idle). The main area has a 'Welcome to Jenkins!' message and a 'Start building your software project' section with a 'Create a job' button. Below that is a 'Set up a distributed build' section with links for 'Set up an agent', 'Configure a cloud', and 'Learn more about distributed builds'. A modal dialog titled 'Enter an item name' is open, showing a text input field with 'for-project' and a list of project types: Freestyle project, Pipeline, Multi-configuration project, Folder, Multibranch Pipeline, and Organization Folder. An 'OK' button is at the bottom of the dialog.

Now, it has to get code from github repo.
Choose 'Git' in 'Source Code Management' for this to happen.
Copy paste this repo URL.



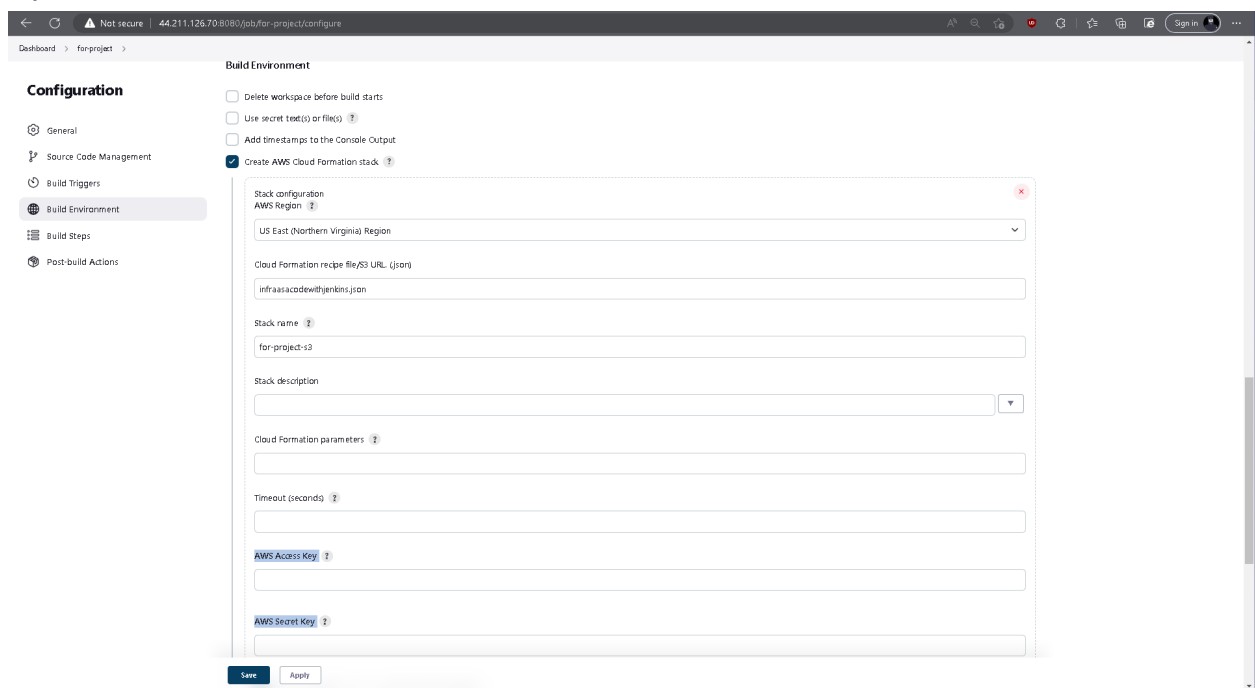
In credentials add your github credentials.



Now in 'Build Environment' choose 'Create AWS Cloud Formation Stack'. **This option is up there because of the plugin that we installed before.**

Fill the details accordingly.

In 'Cloud Formation recipe file/S3 URL. (.json)' section give your json file name from github repo.

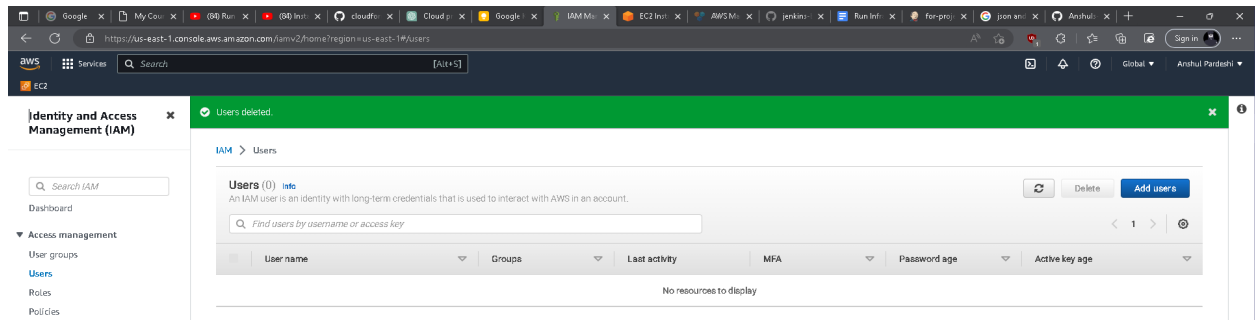


Here you need to enter Access Key and Secret Key.

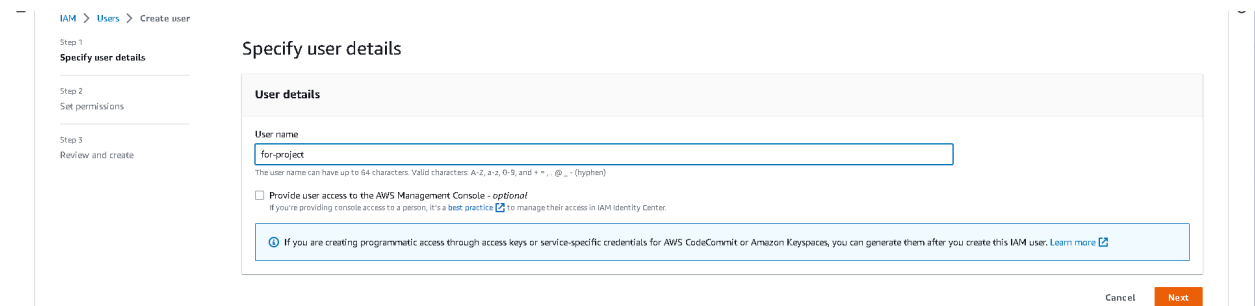
The need for these keys for Jenkins is to call Cloudformation API.

Let us head towards AWS to get Access Key and Secret Key.

Goto IAM>>Users>>Add user from AWS Console.

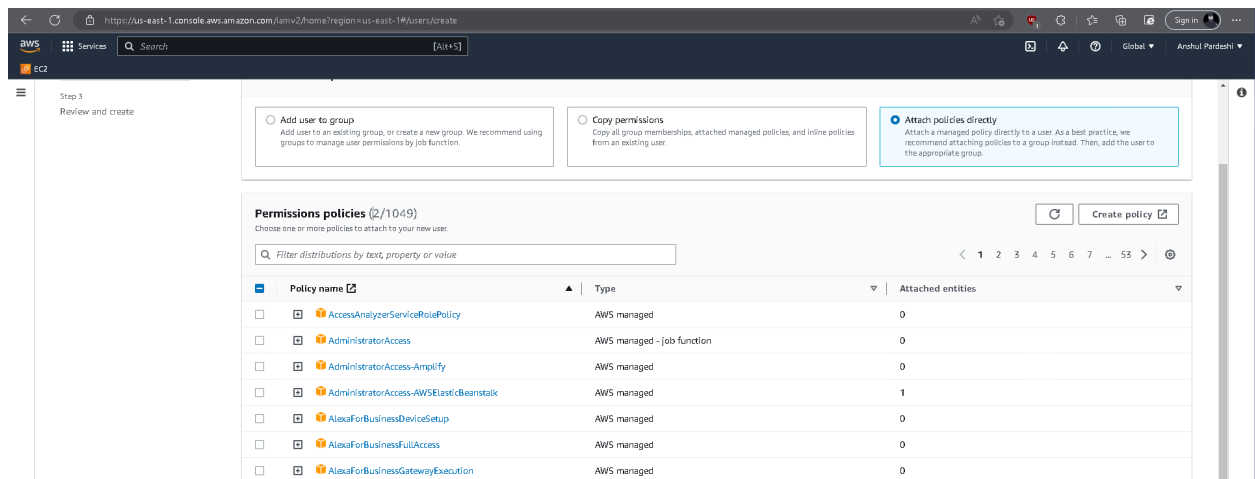


Name the user.

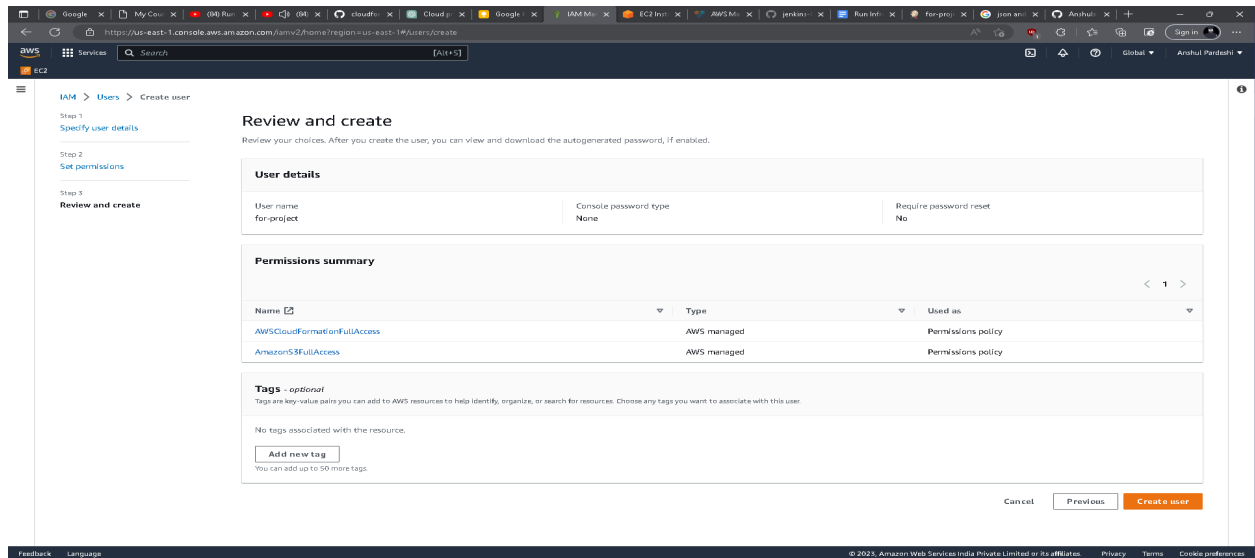


Now, attach a policy to the user.

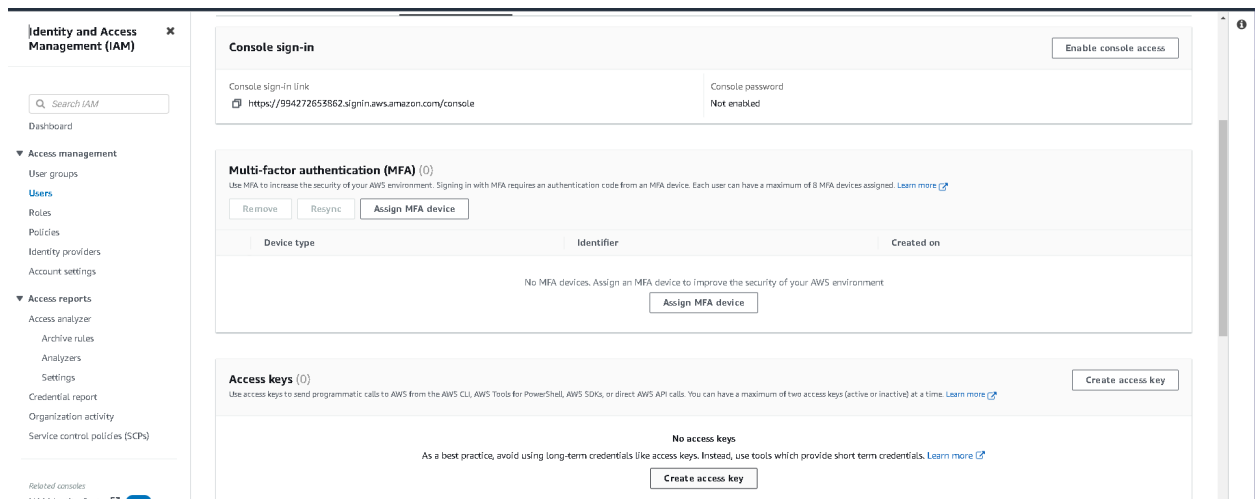
You need to attach 'CloudFormation FullAccess' and 'S3 bucket FullAccess' policies to it.



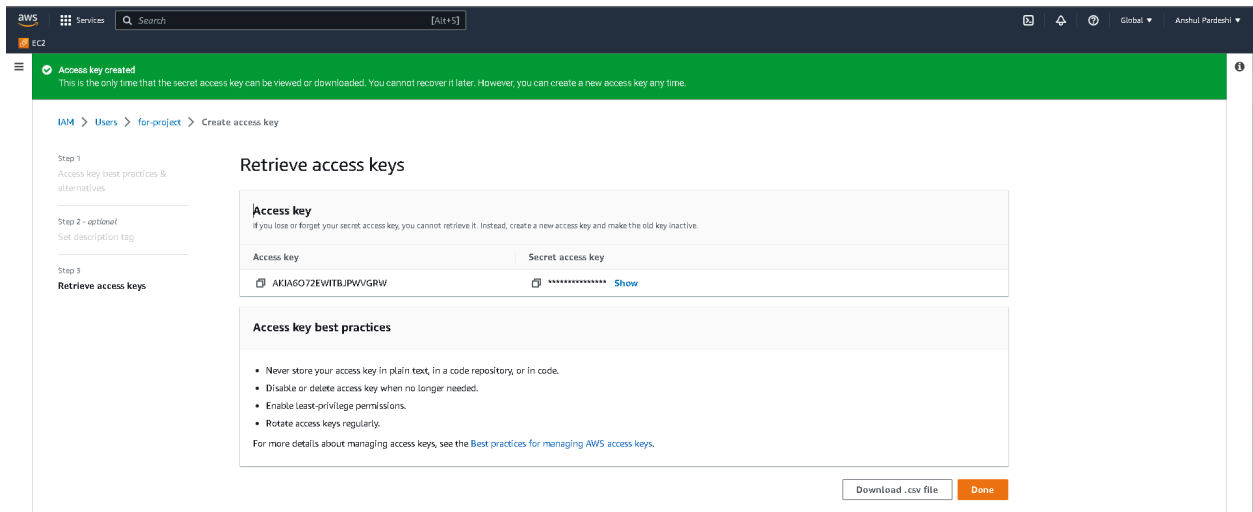
Check the policies attached and Create User.



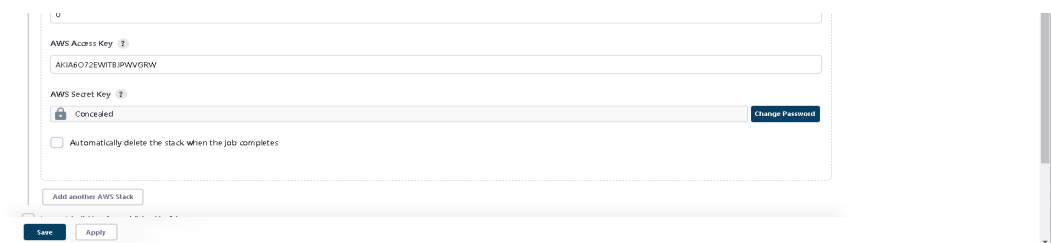
Open the user that you just created and create an Access Key.



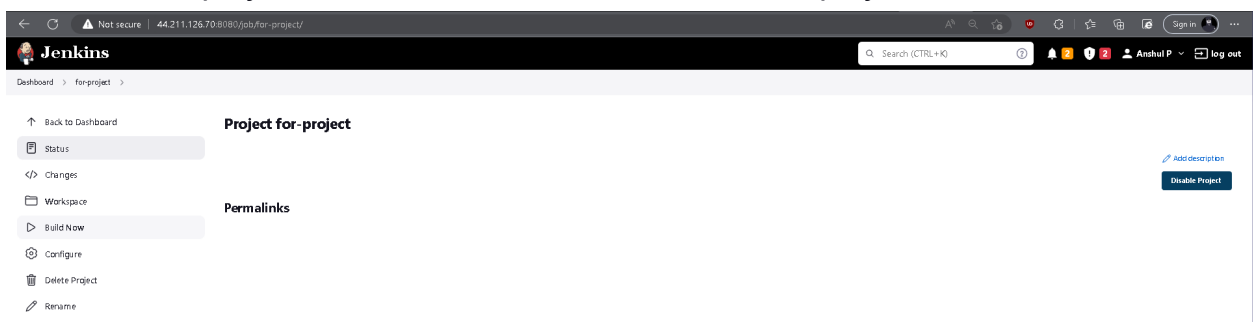
Copy the Access Key and Secret Key.



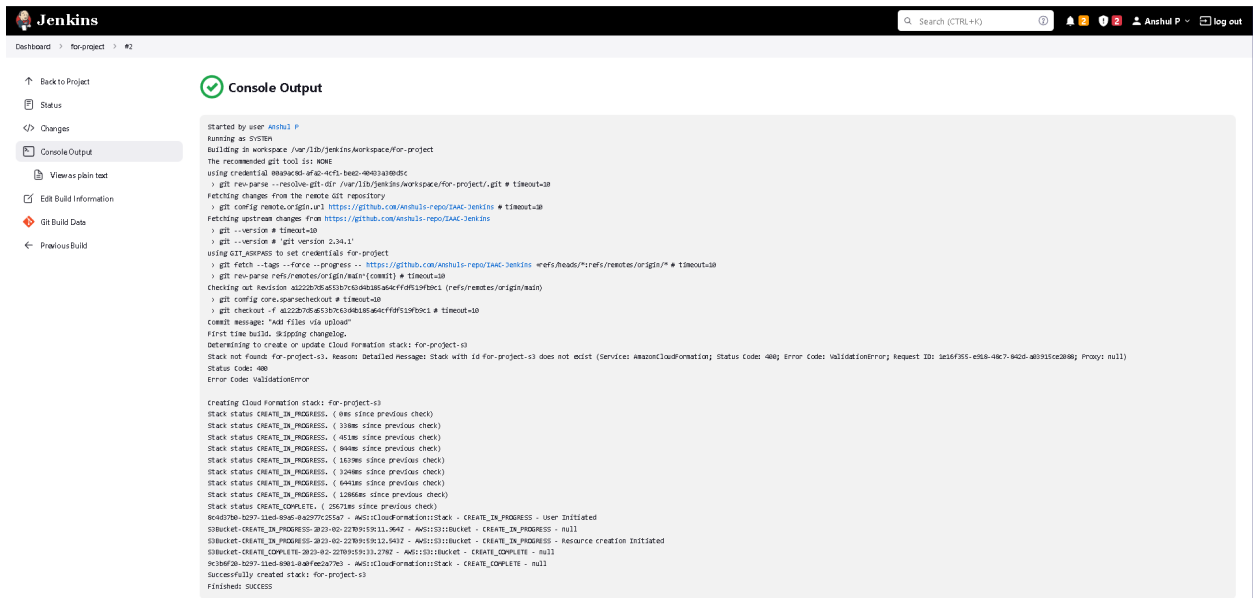
And paste the Access Key and Secret Access Key here.



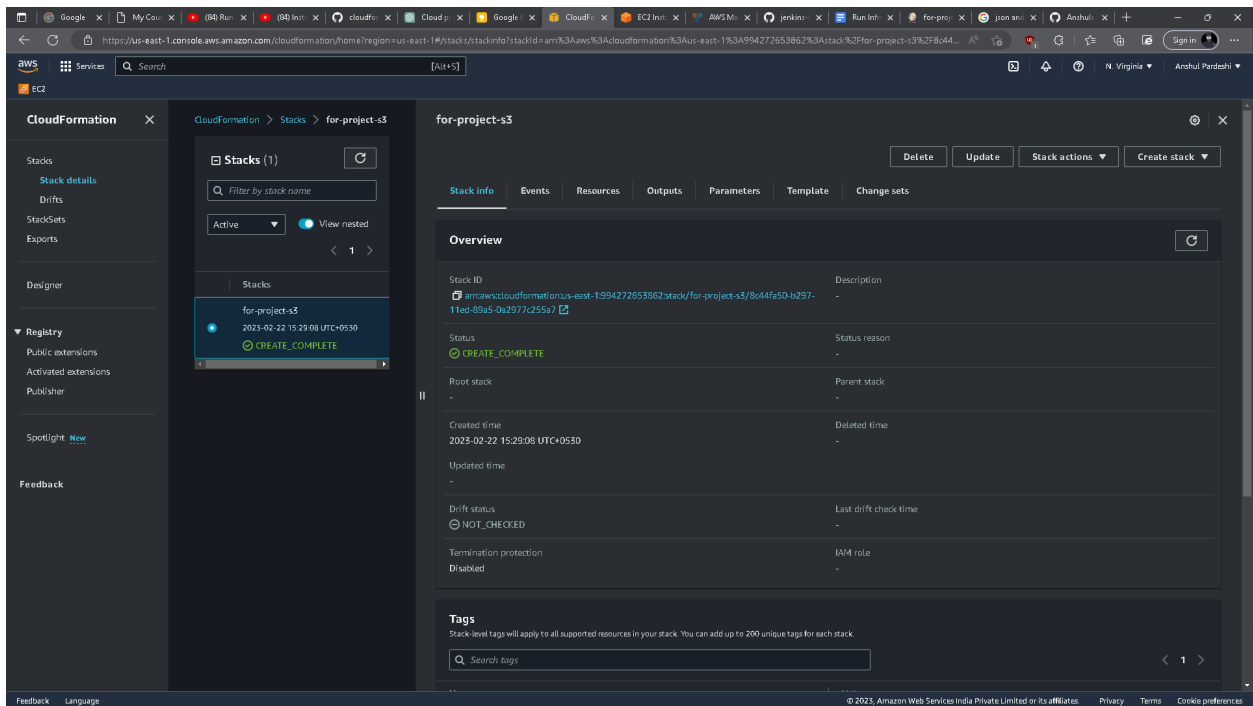
Now save the project. Then select 'Build Now' to build the project.



You can see logs and the project has been built.



Also in AWS console CloudFormation>>Stack you can see the Stack has been created.



Part:2

Now, we need to install AWS CLI. We can do it in multiple ways like SSH into instance and install or we can also use AWS SSM Session Manager to install without keys.

We will be just SSHing into it and then download using commands.

First SSH into it. Then update using: `sudo apt-get update`

Then install awscli using: `sudo apt-get install awscli`

[illegible]

We will be using following jenkins file to pass on command on CLI:

```
pipeline {
  agent any
  stages {
    stage('Submit Stack') {
      steps {
        sh "aws cloudformation create-stack --stack-name s3bucket --template-body
file:///infraasacodewithjenkins.json --region 'us-east-1'"
      }
    }
  }
}
```

This is a Jenkins pipeline script that uses the AWS CLI to create an AWS CloudFormation stack from a template file. The pipeline has one stage called "Submit Stack" that consists of a single step.

Here's what the different parts of the pipeline script do:

"pipeline": This is the top-level block of the Jenkins pipeline script that defines the entire pipeline.

"agent any": This specifies that the pipeline can run on any available agent. In other words, it allows the Jenkins master to allocate an executor on any available Jenkins node to run this pipeline.

"stages": This section contains a list of stages that the pipeline will execute. In this case, there is only one stage called "Submit Stack".

"stage('Submit Stack')": This defines a stage in the pipeline called "Submit Stack".

"steps": This section contains a list of steps that the pipeline will execute as part of the "Submit Stack" stage.

"sh": This is a Jenkins step that allows running shell commands. In this case, it runs the AWS CLI command to create a CloudFormation stack with the name "s3bucket" using the "infraasacodewithjenkins.json" template file in the "us-east-1" region.

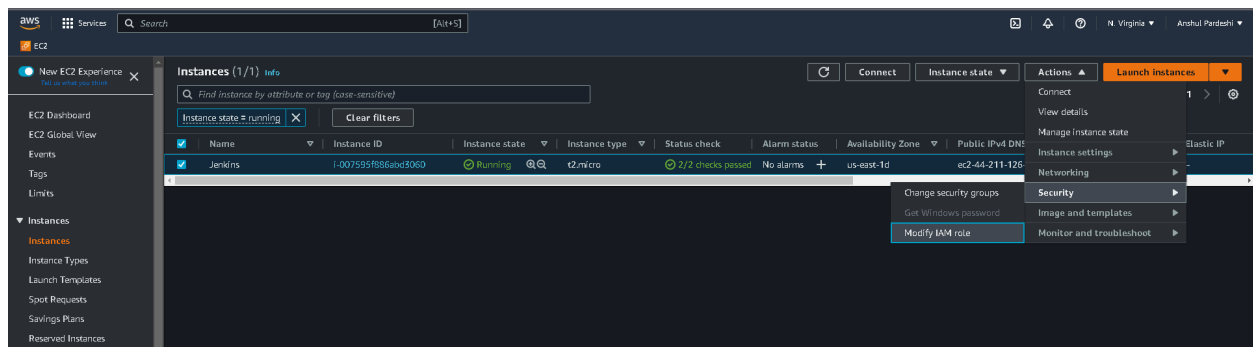
Overall, this pipeline script automates the creation of an AWS CloudFormation stack using the AWS CLI. When executed, the pipeline will create the stack named "s3bucket" in the "us-east-1" region using the "infraasacodewithjenkins.json" template file.

Unlike the previous part, here we will use IAM Role.

Let us create and attach an IAM role to EC2 with full S3 and CloudFormation Access.

Select the EC2 instance and go to:

Actions Drop Down Menu>>Security>>Modify IAM Roles



Create an IAM ROLE with following policies attached.

The screenshot shows the AWS IAM console 'Create role' wizard. The role is named 'for-project' and is of type 'AssumeRole'. The principal is 'ec2.amazonaws.com'. Two policies are attached: 'AWSCloudFormationFullAccess' and 'AmazonS3FullAccess'. The 'Add tags' section is empty.

```
3-   "Statement": [
4-     {
5-       "Effect": "Allow",
6-       "Action": [
7-         "sts:AssumeRole"
8-       ],
9-       "Principal": {
10-        "Service": [
11-          "ec2.amazonaws.com"
12-        ]
13-      }
14-    ]
15-  ]
16- }
```

Step 2: Add permissions

Permissions policy summary

Policy name	Type	Attached as
AWSCloudFormationFullAccess	AWS managed	Permissions policy
AmazonS3FullAccess	AWS managed	Permissions policy

Tags

Add tags - optional [info](#)

Tags are key-value pairs that you can add to AWS resources to help identify, organize, or search for resources.

No tags associated with the resource.

[Add tag](#)

You can add up to 50 more tags.

[Cancel](#) [Previous](#) [Create role](#)

Attach that IAM role to EC2 instance

The screenshot shows the AWS IAM console 'Modify IAM role' wizard. The role is named 'for-project' and is of type 'AssumeRole'. The principal is 'ec2.amazonaws.com'. The 'Add tags' section is empty.

EC2 > Instances > i-007595f886abd3060 > Modify IAM role

Modify IAM role [info](#)

Attach an IAM role to your instance.

Instance ID

[i-007595f886abd3060](#) (jenkins)

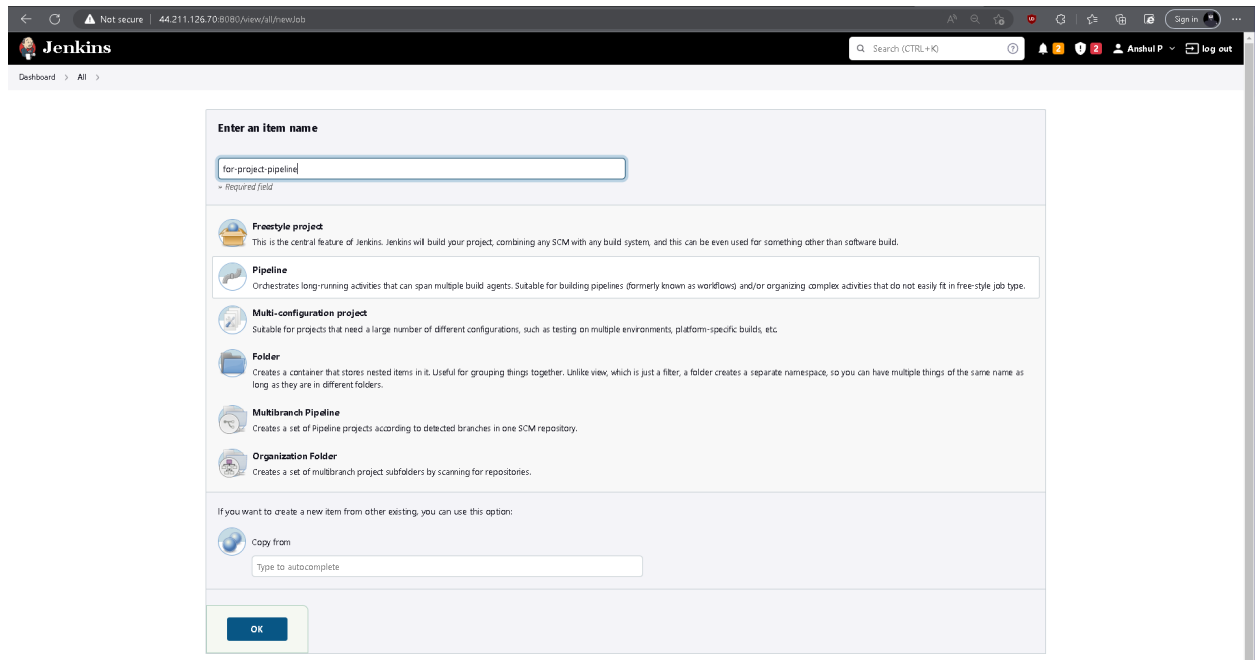
IAM role

Select an IAM role to attach to your instance or create a new role if you haven't created any. The role you select replaces any roles that are currently attached to your instance.

[for-project](#) [Create new IAM role](#)

[Cancel](#) [Update IAM role](#)

Now let us create a Jenkins Pipeline Job.
First name your job and choose pipeline as type of job.

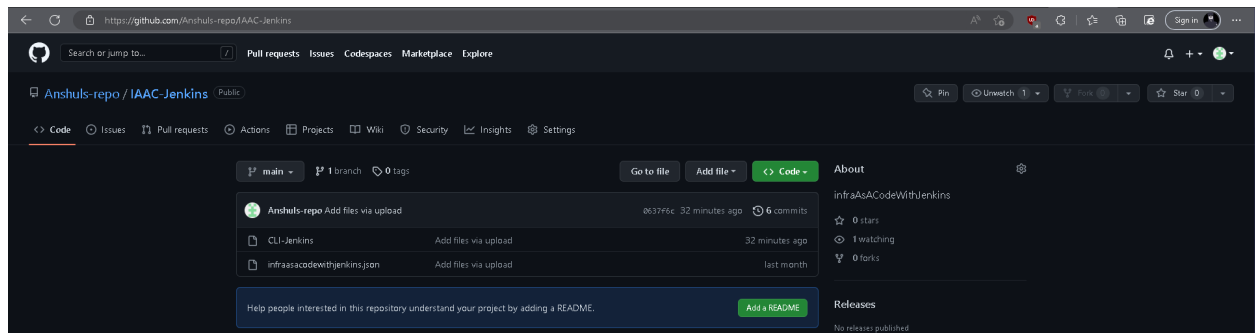


The image shows the Jenkins 'Enter an item name' dialog. At the top, there's a text input field with 'for-project-pipeline' entered. Below this, there's a list of job types with icons and descriptions:

- 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.
- Multibranch Pipeline**: Creates a set of Pipeline projects according to detected branches in one SCM repository.
- Organization Folder**: Creates a set of multibranch project subfolders by scanning for repositories.

Below the list, there's a section 'If you want to create a new item from other existing, you can use this option:' with a 'Copy from' input field and a 'Type to autocomplete' placeholder. At the bottom, there's an 'OK' button.

Straight away go to change 'Pipeline Configuration'. Here choose 'pipeline script from SCM'. Then choose GIT. Fill the details accordingly. We will be using this repo with Jenkins file and CloudFormation File.



Details has been filled according to the resources in GIT.

The screenshot shows the Jenkins Pipeline Configuration page for a job named 'for-project-pipeline'. The left sidebar has 'Configuration' selected, with sub-options for 'General', 'Advanced Project Options', and 'Pipeline'. The 'Pipeline' option is active. The main area is titled 'Pipeline' and contains the following fields:

- Definition:** A dropdown menu set to 'Pipeline script from SCM'.
- SCM:** A dropdown menu set to 'git'.
- Repositories:** A section with a 'Repository URL' field containing 'https://github.com/Anshul-repo/IAAC-Jenkins', a 'Credentials' dropdown set to 'for-project', and an 'Add' button. Below this is an 'Add Repository' button.
- Branches to build:** A section with a 'Branch Specifier (blank for \'any\')' field containing '*/main' and an 'Add Branch' button.
- Repository browser:** A dropdown menu set to 'Auto'.
- Additional behaviours:** A section with an 'Add +' button.
- Script Path:** A text field containing 'CU/Jenkins'.

At the bottom of the configuration area are 'Save' and 'Apply' buttons.

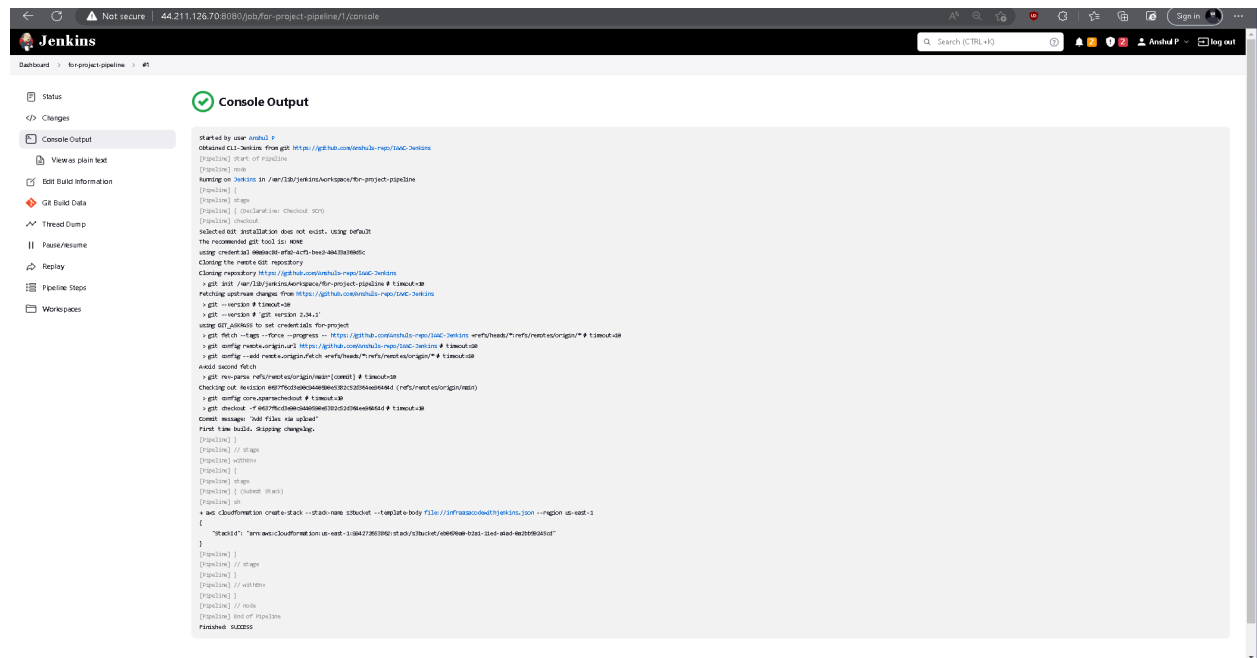
Then choose “Build now’ to build the job.

The screenshot shows the Jenkins Pipeline View page for the job 'for-project-pipeline'. The left sidebar has 'Build History' selected, showing 'No builds' and buttons for 'Align feed for all' and 'Align feed for failures'. The main area is titled 'Pipeline for-project-pipeline' and contains the following sections:

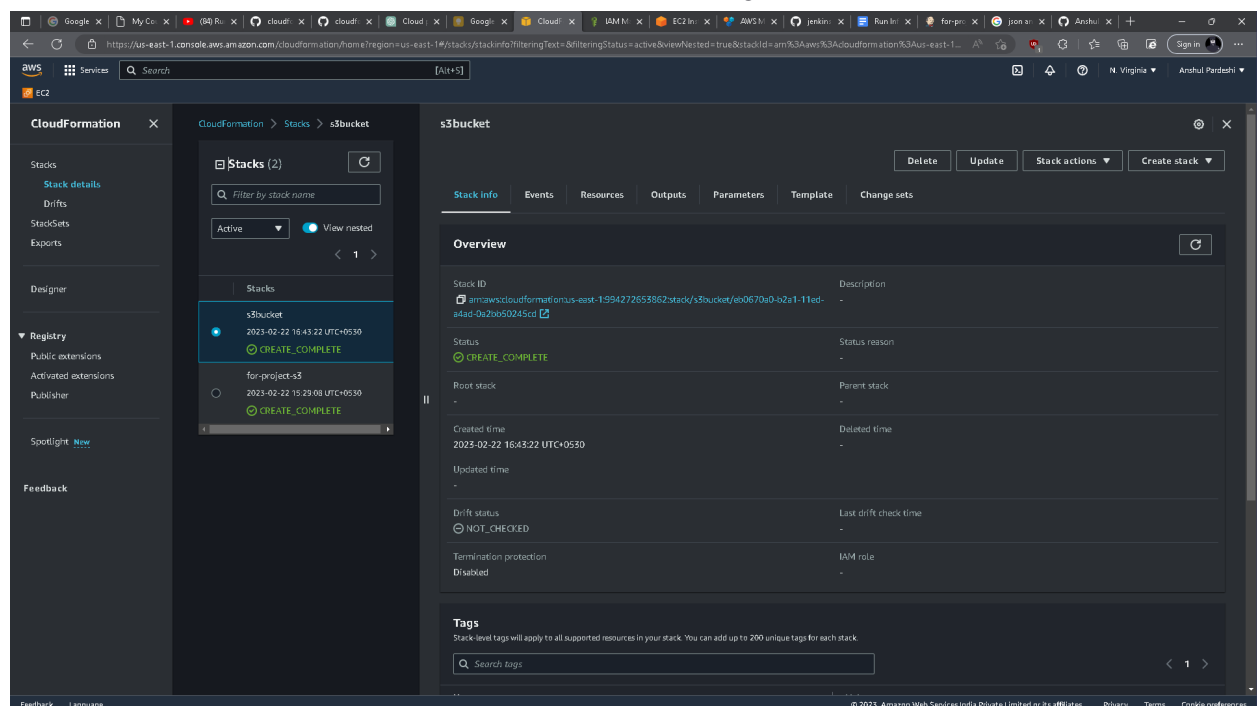
- Stage View:** A section with a message: 'No data available. This Pipeline has not yet run.'
- Permalinks:** A section with a message: 'No data available. This Pipeline has not yet run.'

At the top right of the main area are buttons for 'Add description' and 'Disable Project'. The bottom of the page shows the REST API endpoint '44211.126.70.8080/job/for-project-pipeline/build?delay=0sec' and the Jenkins version 'Jenkins 2.361.1'.

In logs you can see the job has been built successfully. Also The command in 'CLI-Jenkins' file is being executed.



The s3 Bucket has been created which was our end goal.



Conclusion: You can deploy any infrastructure by just changing the Code in “infraasacodewithjenkins.json” with respective code for the infrastructure that you want in github. The pipeline will do the job for you.