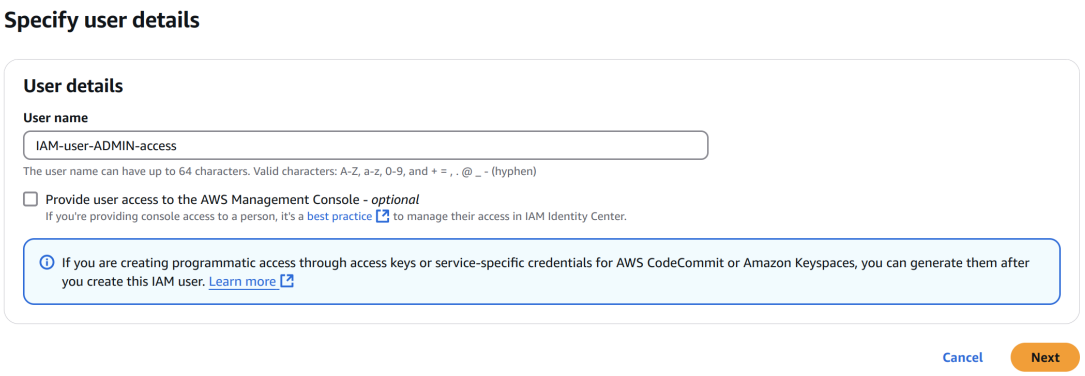
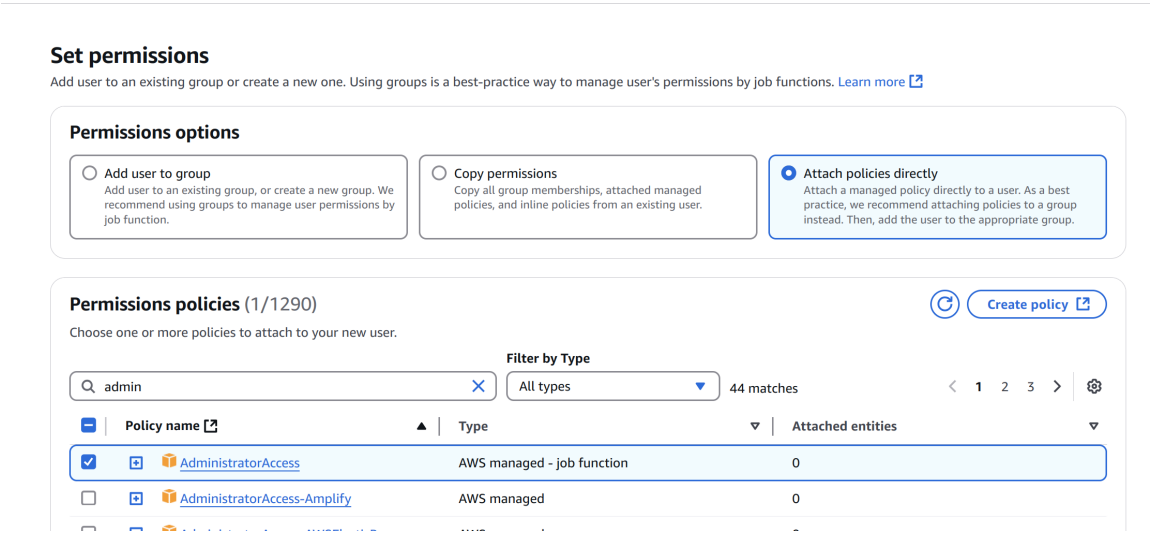
# CI/CD and Kubernetes Deployment Guide: Amazon Prime Clone

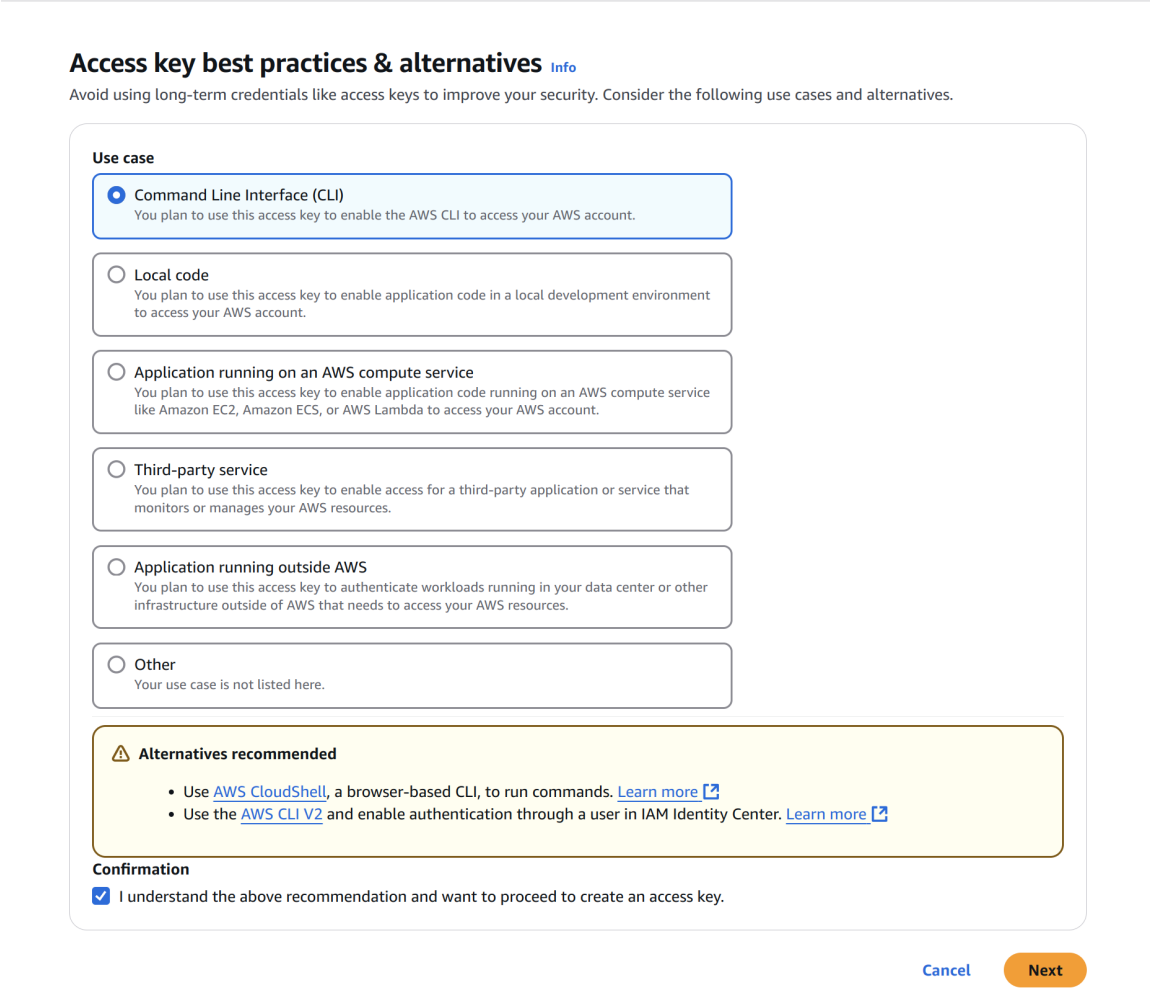
## github repo: [github.com/saisabari24/amazonPrime-CICID](github.com/saisabari24/amazonPrime-CICID)

1. Create a IAM user with Admin access and copy the access and secret keys for CLI

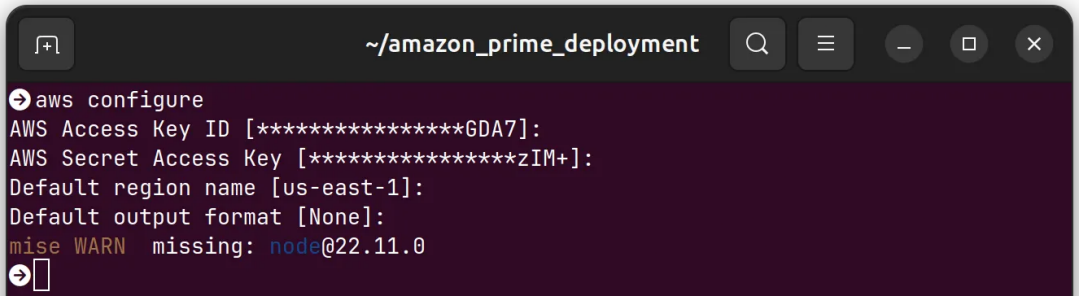
---



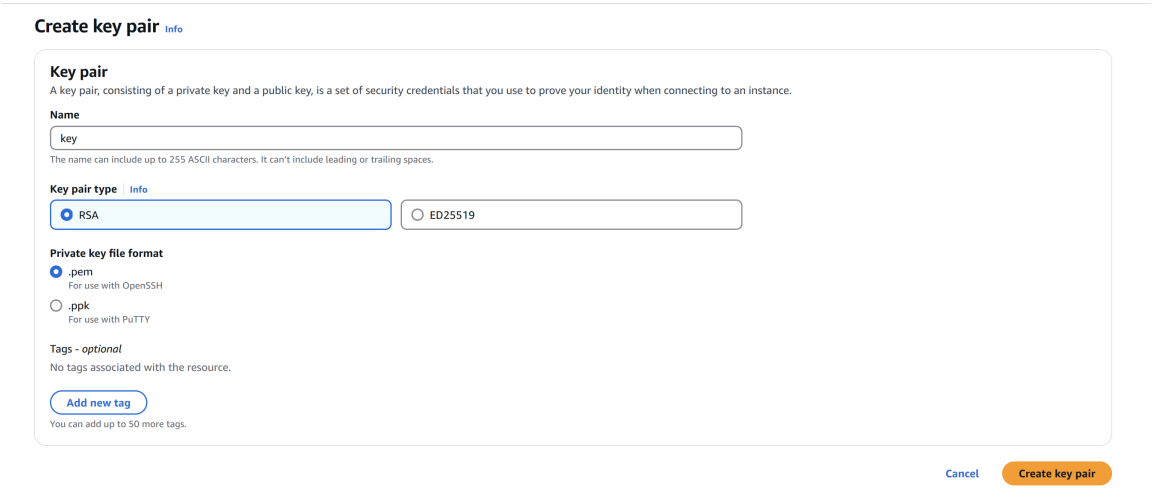



2. Configure AWS CLI in your local setup with the access and secret keys generated




3. Create a keypair “key.pem” for ec2 access



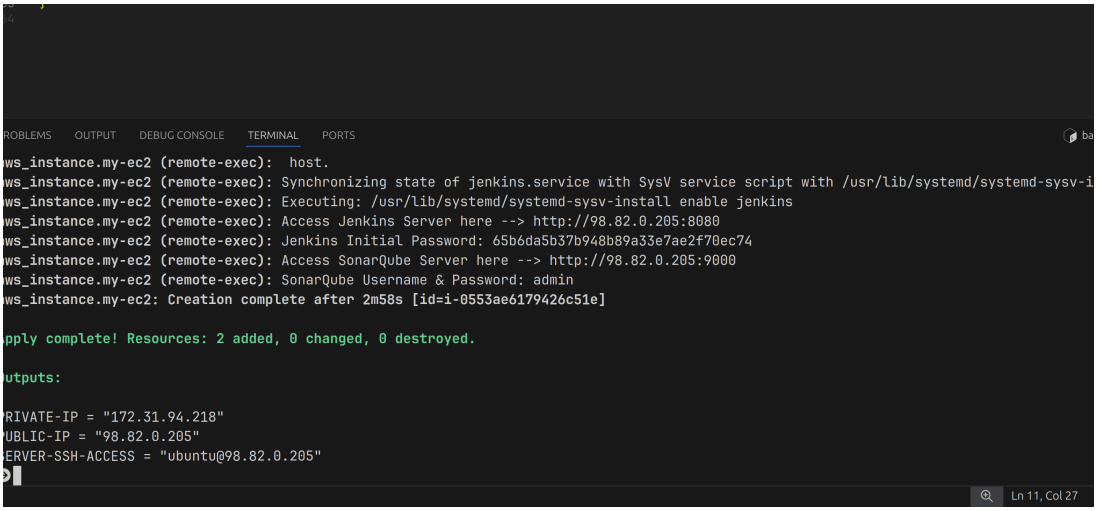

4. Clone the git repo into local and open in VS Code
5. Terraform EC2
   - cd to the terraform_code/ec2_server
   - Run “terraform init”  & “terraform apply —auto-approve ”
   - URL details will be displayed at the end of creation of resources

```
terraform {
  required_providers {
    aws = {
      source = "hashicorp/aws"
      version = "5.67.0"
    }
  }
}

provider "aws" {
  region = var.region_name
}

# STEP1: CREATE SG
resource "aws_security_group" "my-sg" {
  name        = "JENKINS-SERVER-SG"
  description = "Jenkins Server Ports"

  # Port 22 is required for SSH Access
  ingress {
    description     = "SSH Port"
    from_port       = 22
    to_port         = 22
    protocol        = "tcp"
    cidr_blocks = ["0.0.0.0/0"]
  }

  # Port 80 is required for HTTP
  ingress {
    description     = "HTTP Port"
    from_port       = 80
    to_port         = 80
    protocol        = "tcp"
    cidr_blocks = ["0.0.0.0/0"]
  }

  # Port 443 is required for HTTPS
  ingress {
    description     = "HTTPS Port"
    from_port       = 443
    to_port         = 443
    protocol        = "tcp"
```

```
cd ec2_server/
terraform init
Initializing the backend...
Initializing provider plugins...
- Finding hashicorp/aws versions matching "5.67.0"...
- Installing hashicorp/aws v5.67.0...
- Installed hashicorp/aws v5.67.0 (signed by HashiCorp)
Terraform has created a lock file .terraform.lock.hcl to record the provider
selections it made above. Include this file in your version control repository
so that Terraform can guarantee to make the same selections by default when
you run "terraform init" in the future.
```

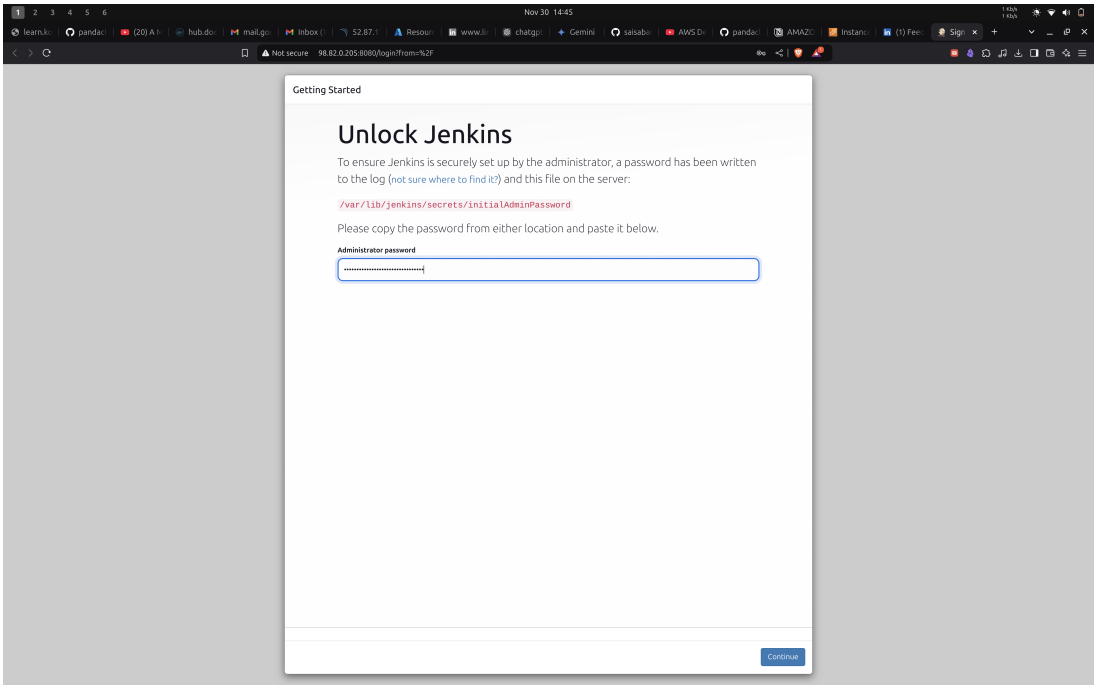
```
terraform {
  required_providers {
    aws = {
      source = "hashicorp/aws"
      version = "5.67.0"
    }
  }
}

provider "aws" {
  region = var.region_name
}

# STEP1: CREATE SG
resource "aws_security_group" "my-sg" {
  name        = "JENKINS-SERVER-SG"
  description = "Jenkins Server Ports"

  # Port 22 is required for SSH Access
  ingress {
    description     = "SSH Port"
    from_port       = 22
    to_port         = 22
    protocol        = "tcp"
    cidr_blocks = ["0.0.0.0/0"]
  }

  # Port 80 is required for HTTP
  ingress {
    description     = "HTTP Port"
    from_port       = 80
    to_port         = 80
    protocol        = "tcp"
    cidr_blocks = ["0.0.0.0/0"]
```

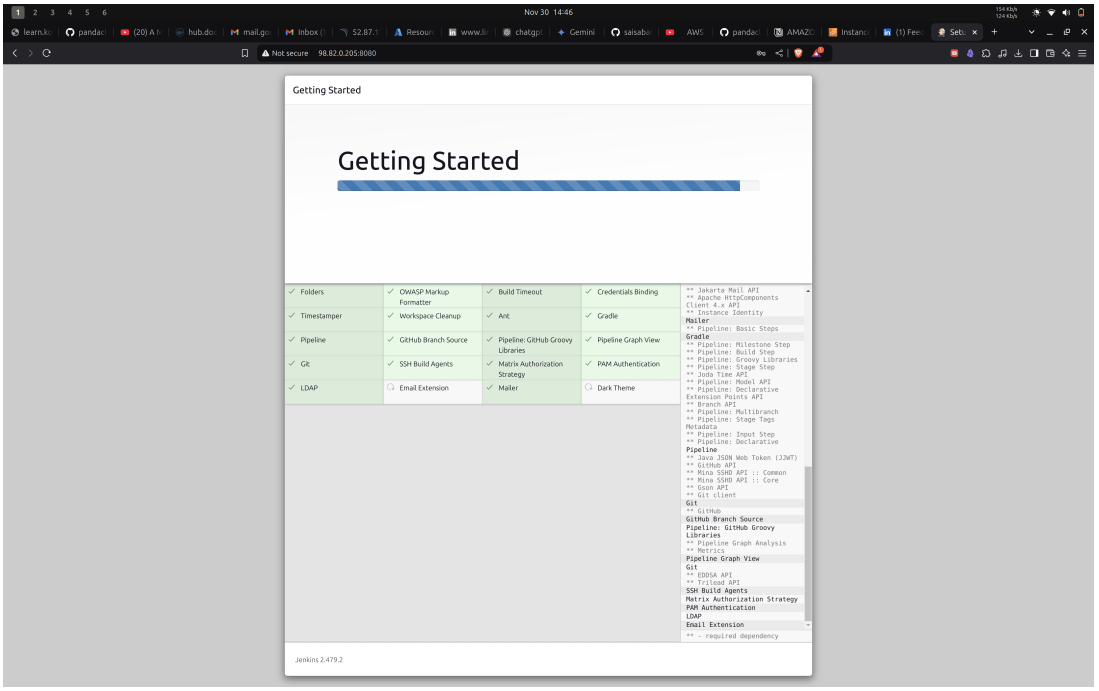
```
- Finding hashicorp/aws versions matching "5.67.0"...
terraform plan

Terraform used the selected providers to generate the following execution plan. Resource actions are indicated with the following symbols:
  + create

Terraform will perform the following actions:

  # aws_instance.my-ec2 will be created
  + resource "aws_instance" "my-ec2" {
      + ami                                  = "ami-0e86e20dae9224db8"
      + arn                                  = (known after apply)
      + associate_public_ip_address          = (known after apply)
      + availability_zone                    = (known after apply)
      + cpu_core_count                       = (known after apply)
      + cpu_threads_per_core                 = (known after apply)
      + disable_api_stop                     = (known after apply)
```

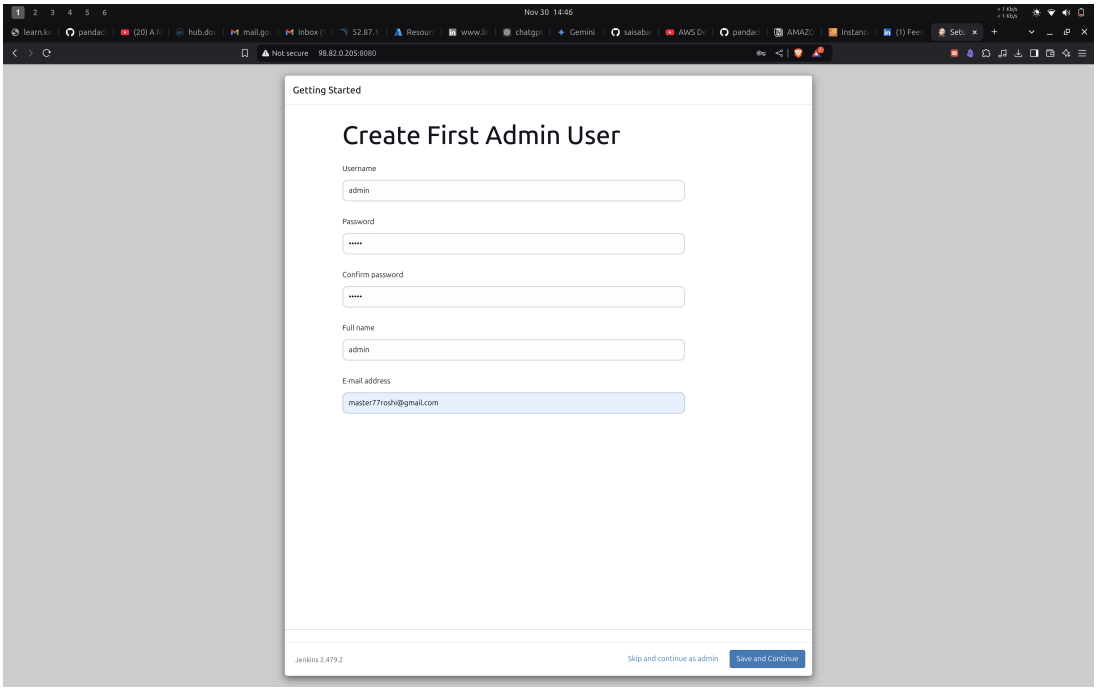



6. Jenkins

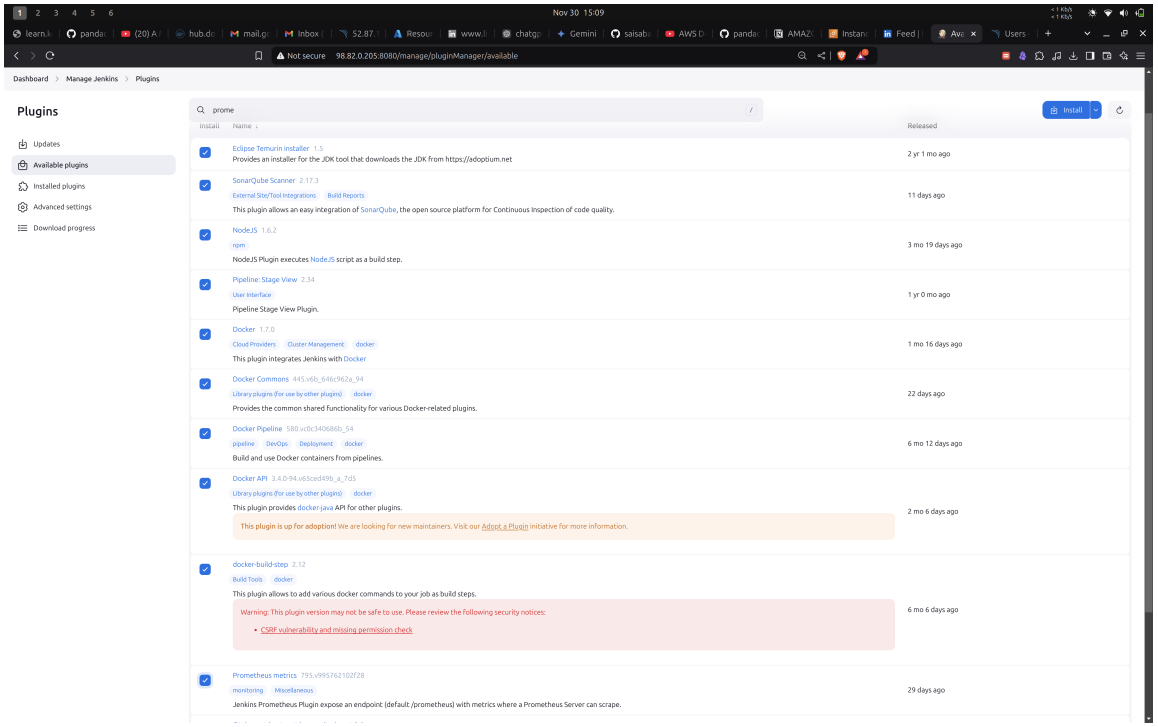
- Hit the url http://<public_IP_address_of_VM>:8080 to access Jenkins
- install suggested plugins
- change credentials
- Re-start Jenkins





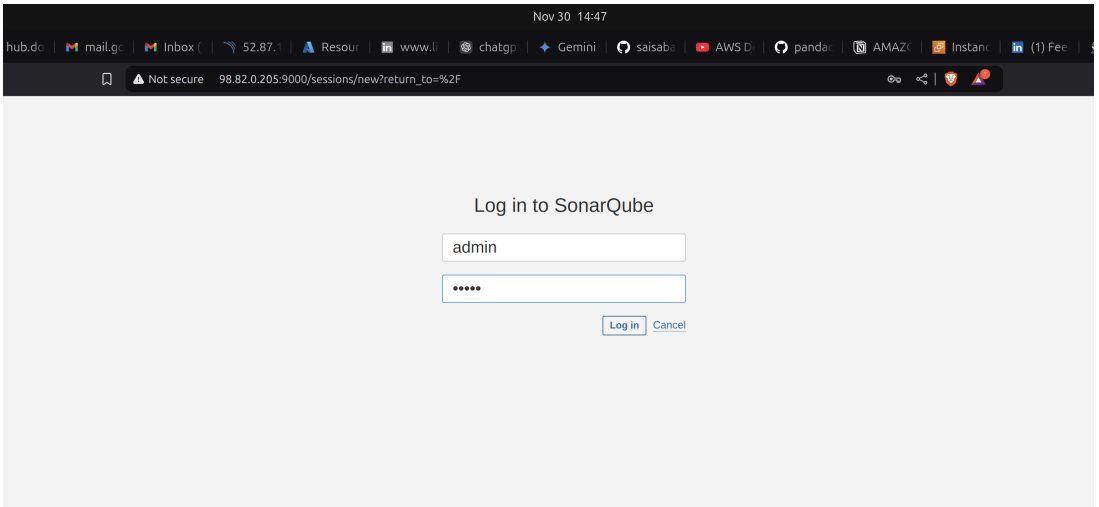



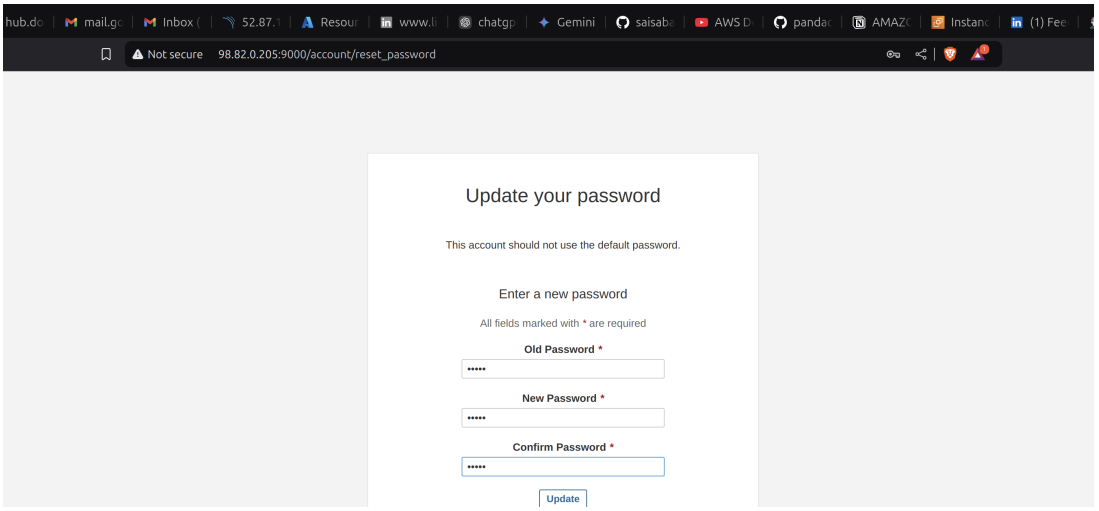

7. Goto → manage Jenkins →
   available  plugins and install the plugins seen below and restart  Jenkins

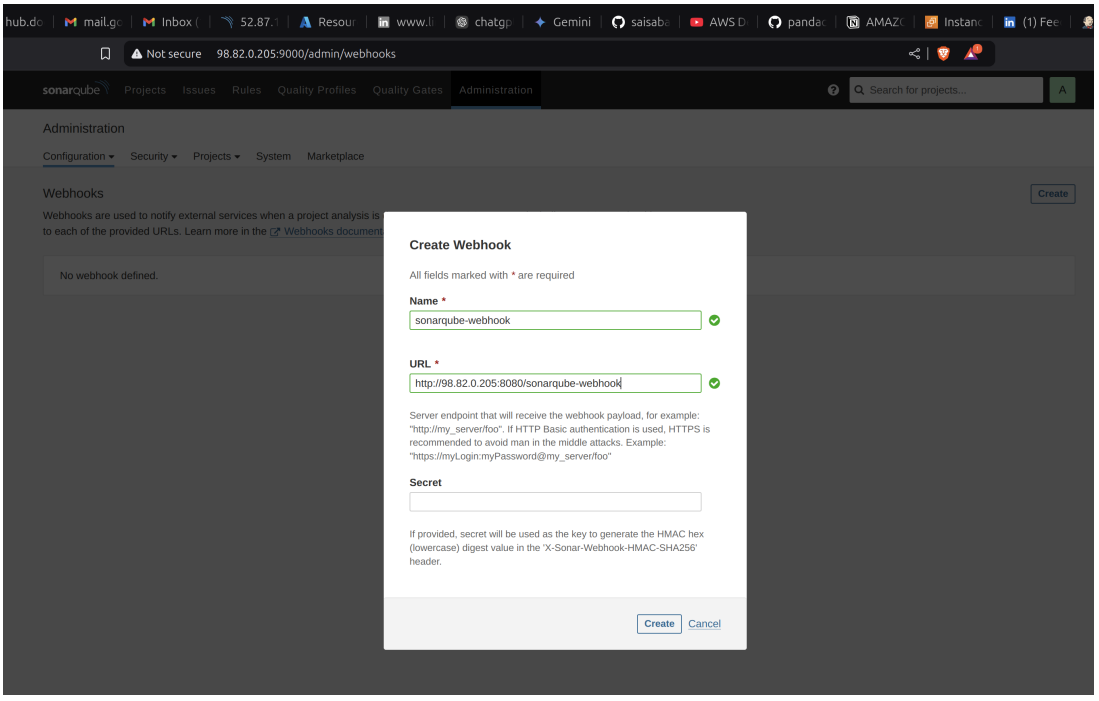


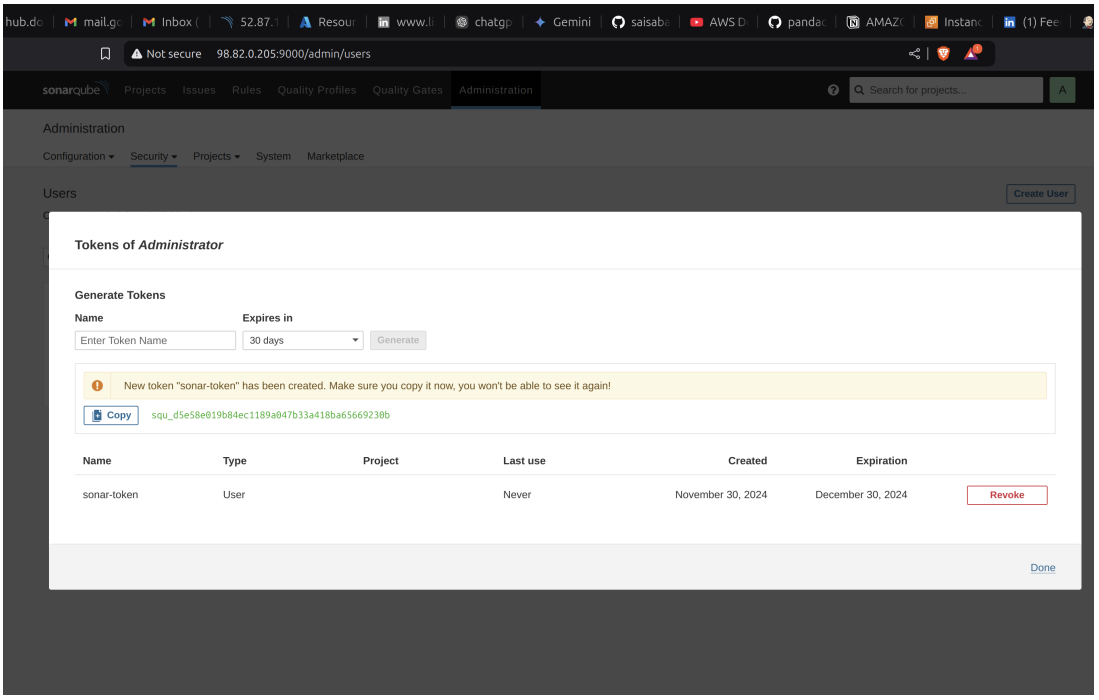

8. go to Sonar by using <public_IP>:9000 and  change credentials

   - goto→Adminstration →Configuration →Webooks and create a webhook for jenkins to create communication between Jenkins and Sonar
     Click on create and enter Name as “jenkins” and
     Url as “http://<public_IP_address_of_VM>:8080”  and save
   - goto →Administration→Security → Users and create a token for Jenkins
   - copy the token



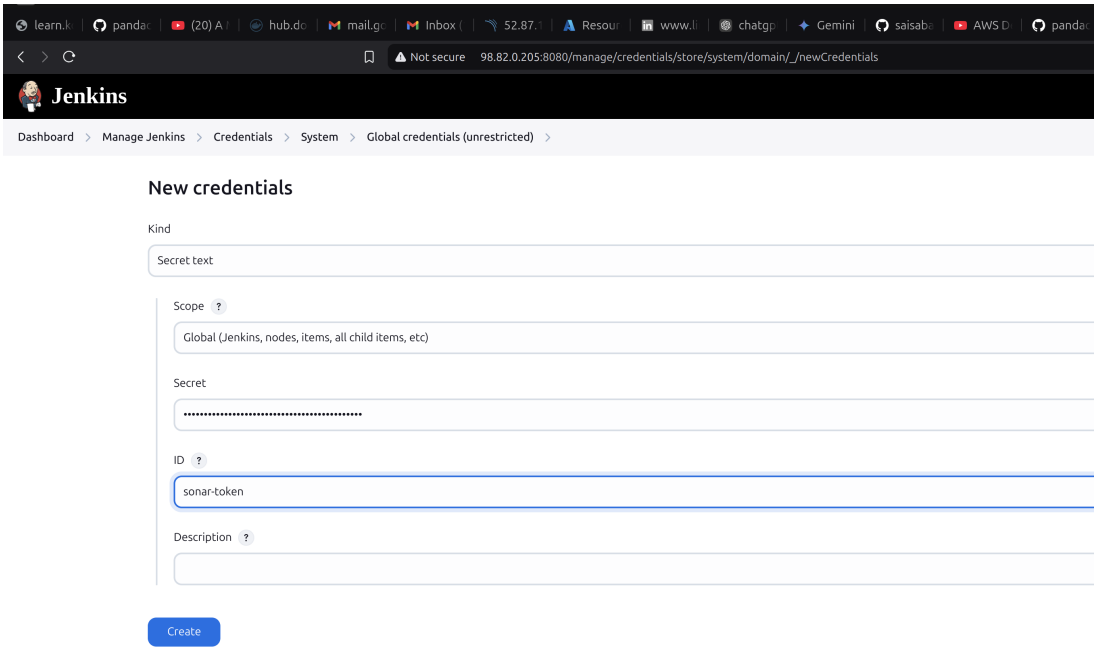

Not secure 98.82.0.205:9000/account/reset_password
Update your password
This account should not use the default password.
Enter a new password
All fields marked with * are required
Old Password *
New Password *
Confirm Password *
Update

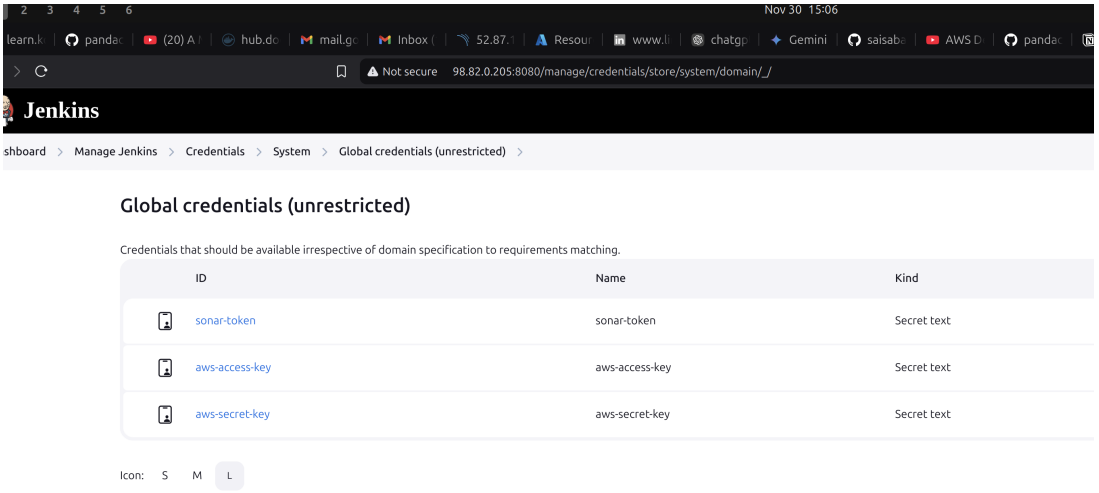

Not secure 98.82.0.205:9000/admin/webhooks
sonarqube
Projects
Issues
Rules
Quality Profiles
Quality Gates
Administration
Search for projects...
Administration
Configuration
Security
Projects
System
Marketplace
Webhooks
Create
No webhook defined.
Create Webhook
All fields marked with * are required
Name *
sonarqube-webhook
URL *
http://98.82.0.205:8080/sonarqube-webhook
Server endpoint that will receive the webhook payload, for example: "http://my_server/foo". If HTTP Basic authentication is used, HTTPS is recommended to avoid man in the middle attacks. Example: "https://myLogin:myPassword@my_server/foo"
Secret
If provided, secret will be used as the key to generate the HMAC hex (lowercase) digest value in the 'X-Sonar-Webhook-HMAC-SHA256' header.
Create
Cancel



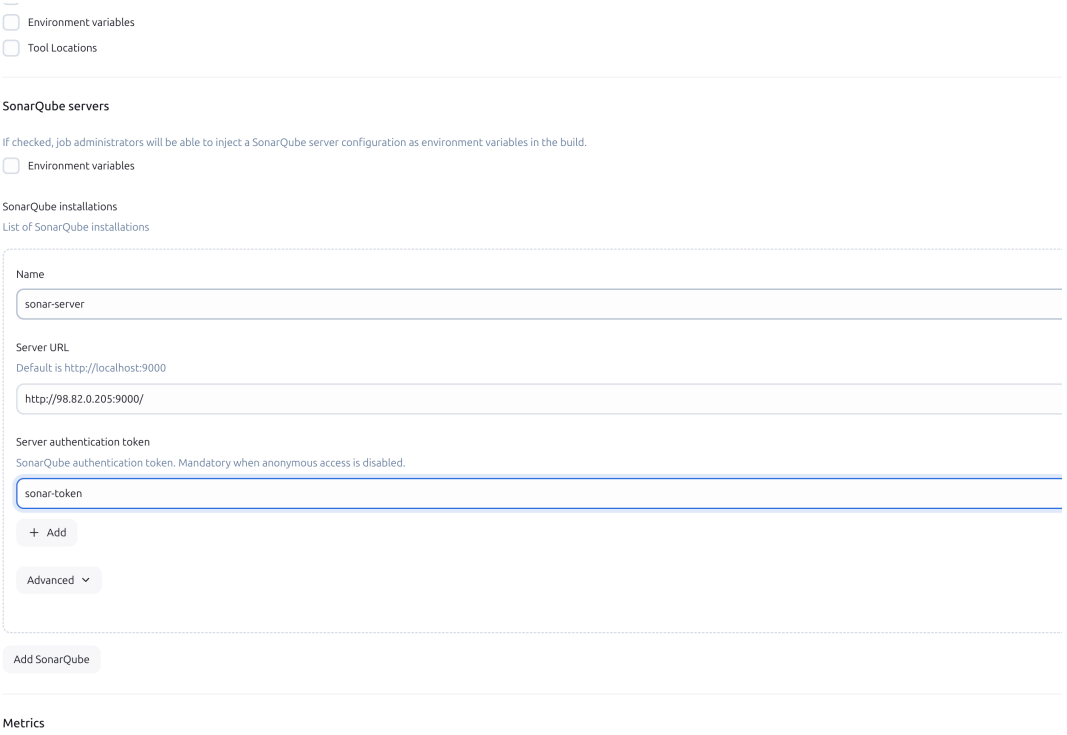

9. Jenkins Credentials

- goto → manage jenkins → credentials → Global →add credentials
- select Secret text in dropdown and paste the token copied from sonar into the secret field and ID as sonar-token and save
- add secret text types for aws access key and secret key and save



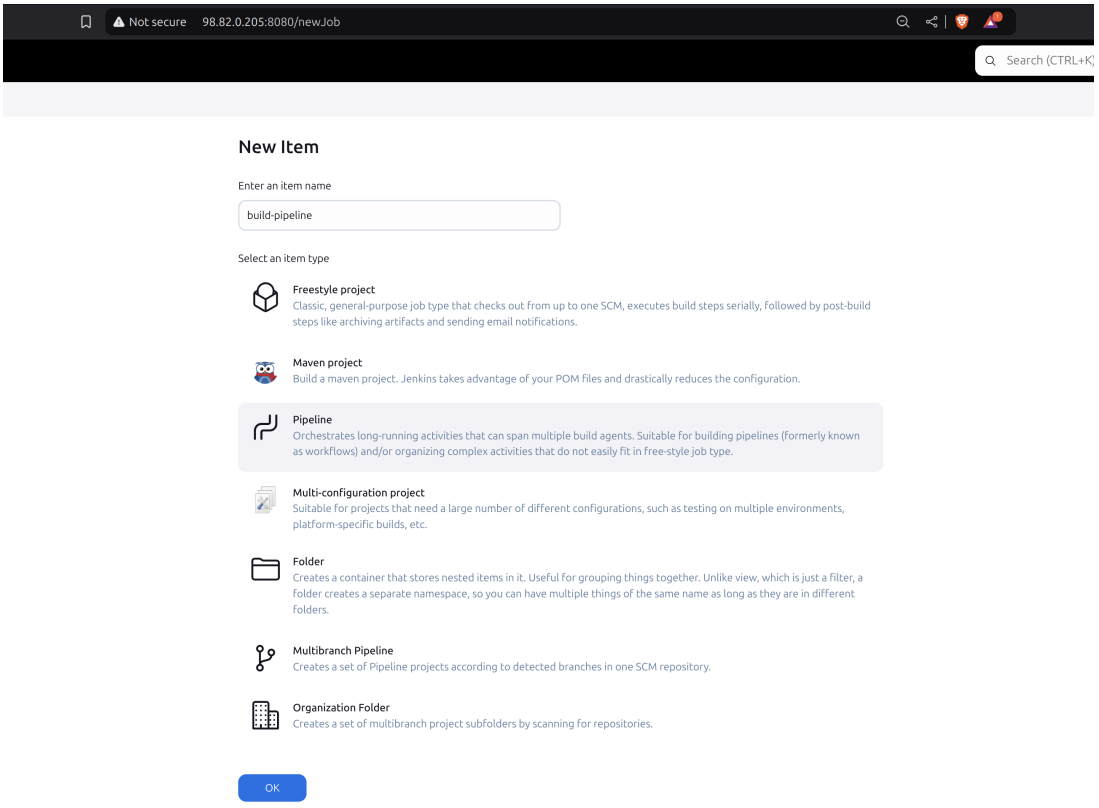


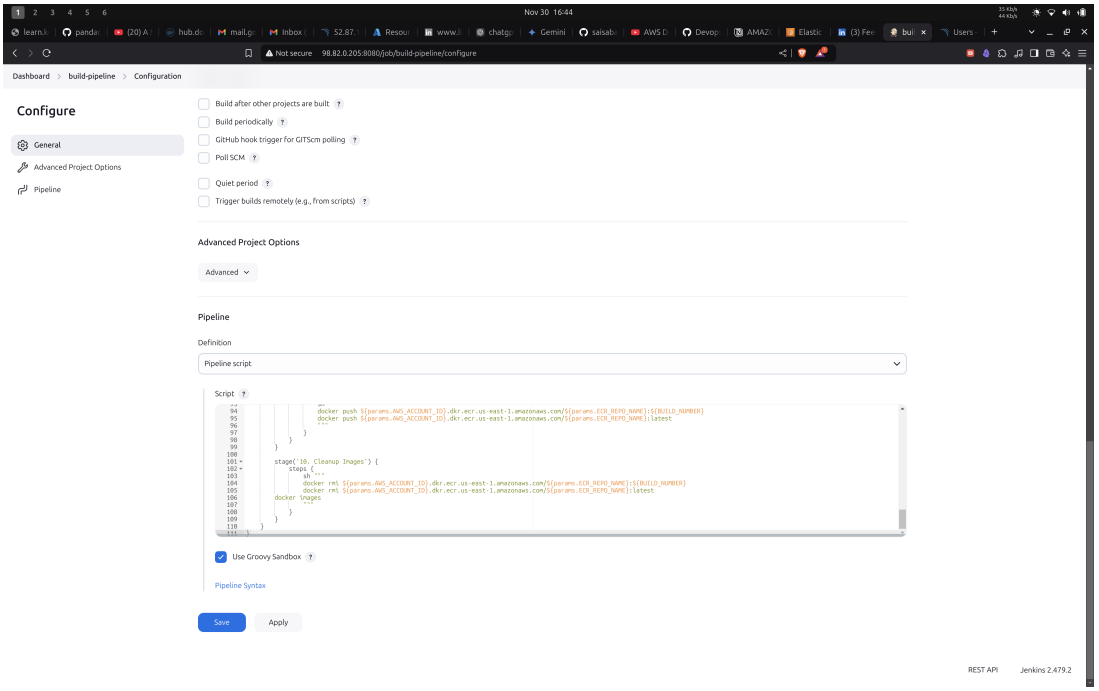

10. Jenkins Tools
    Add the specific versions for JDK17 [jdk 17.0.8] and node16 [node 16.20.0] and rest of the tools can be latest version [docker and sonar scanner]
11. Jenkins Systems
    - Search Sonar and setup with sonar url and sonar-token from credentials

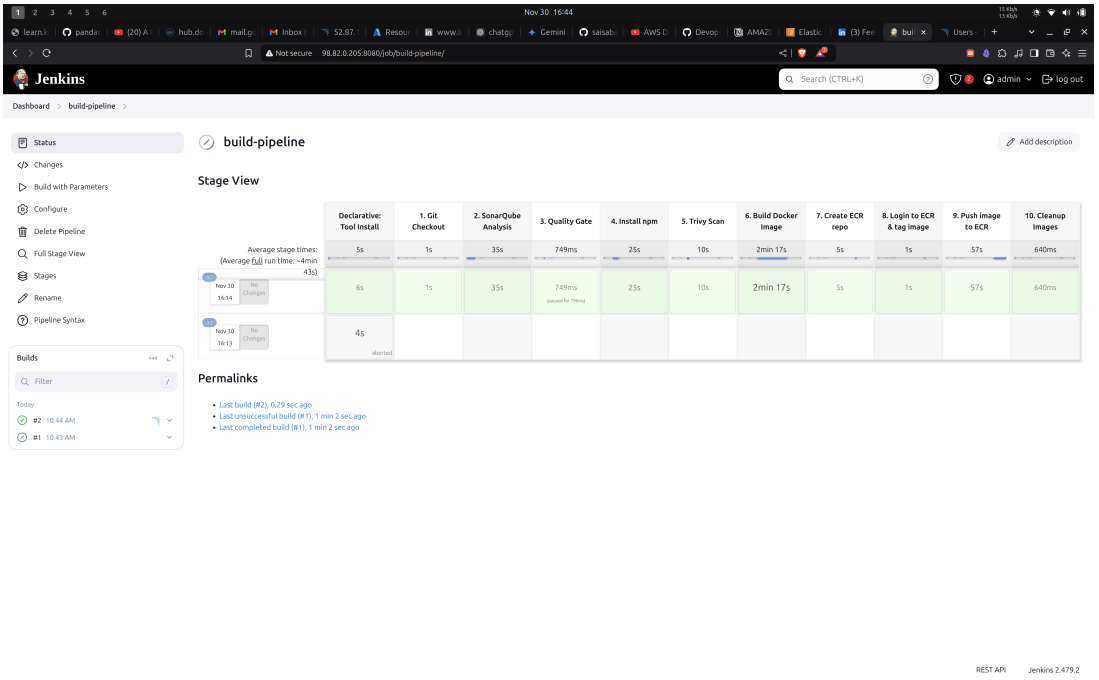


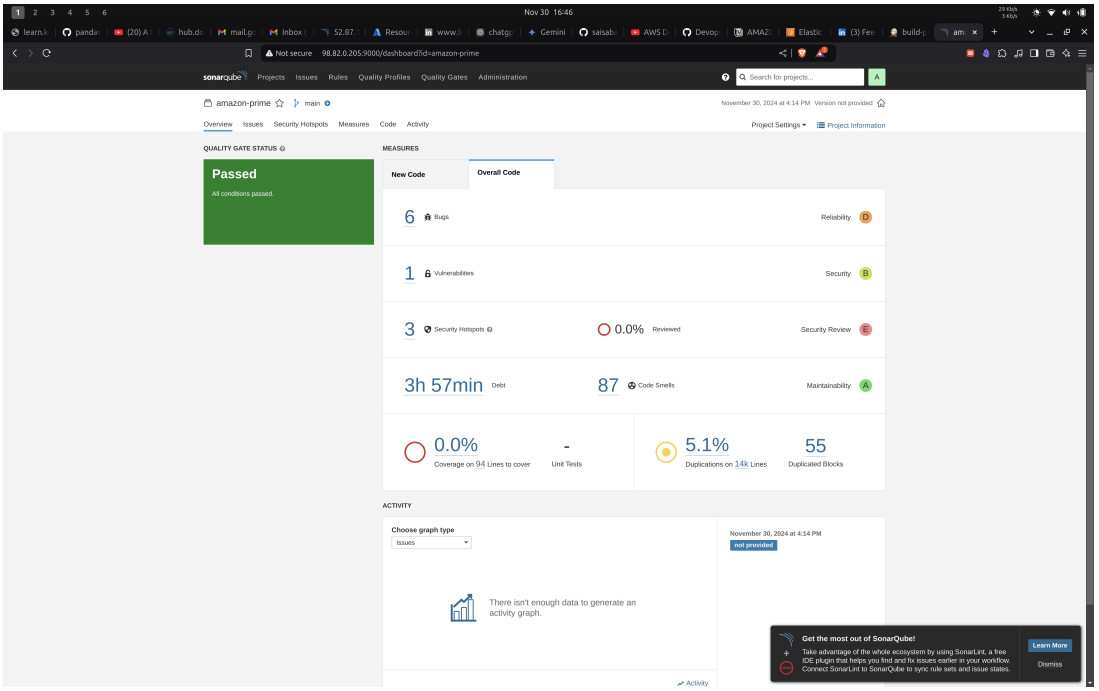

12. Build Pipeline

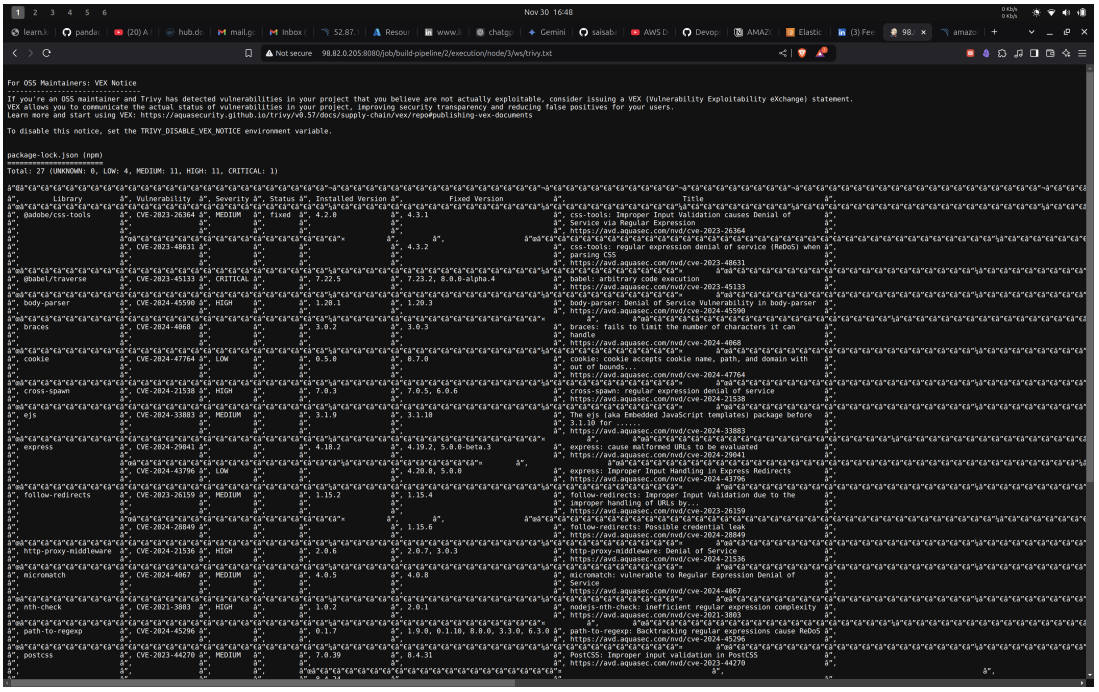
- Goto jenkins Dashboard → add new job/item
  enter Name as build-pipeline and type as Pipeline
- Click next
- Copy the build pipeline script from the github repository mentioned and paste in to Pipeline script
- Now goto → dashboard and click Build Now and abort for the first time and refresh
  now Click Build with Parameters and set the parameters



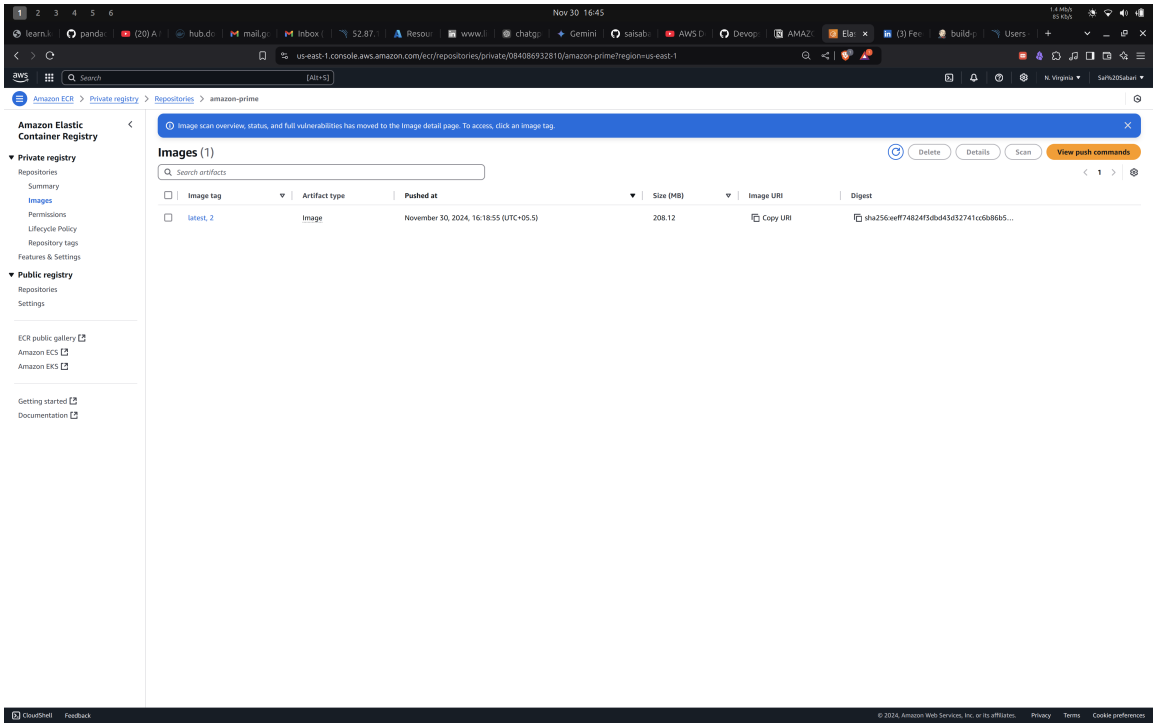

Nov 30 16:44
Not secure 98.82.0.205:8080/job/build-pipeline/configure
Dashboard > build-pipeline > Configuration
Configure
General
Advanced Project Options
Pipeline
Build after other projects are built
Build periodically
GitHub hook trigger for GITScm polling
Poll SCM
Quiet period
Trigger builds remotely (e.g., from scripts)
Advanced Project Options
Advanced
Pipeline
Definition
Pipeline script
Script
docker push ${params.AWS_ACCOUNT_ID}.dkr.ecr.us-east-1.amazonaws.com/${params.ECR_REPO_NAME}:${BUILD_NUMBER}
docker push ${params.AWS_ACCOUNT_ID}.dkr.ecr.us-east-1.amazonaws.com/${params.ECR_REPO_NAME}:latest
stage('10. Cleanup Images') {
steps {
docker rmi ${params.AWS_ACCOUNT_ID}.dkr.ecr.us-east-1.amazonaws.com/${params.ECR_REPO_NAME}:${BUILD_NUMBER}
docker rmi ${params.AWS_ACCOUNT_ID}.dkr.ecr.us-east-1.amazonaws.com/${params.ECR_REPO_NAME}:latest
Use Groovy Sandbox
Pipeline Syntax
Save
Apply
REST API
Jenkins 2.479.2


Nov 30 16:44
Not secure 98.82.0.205:8080/job/build-pipeline/
Jenkins
Search (CTRL+K)
admin
log out
Dashboard > build-pipeline >
Status
Changes
Build with Parameters
Configure
Delete Pipeline
Full Stage View
Stages
Rename
Pipeline Syntax
Builds
Filter
Today
#2 10:44 AM
#1 10:43 AM
build-pipeline
Add description
Stage View
Declarative: Tool Install
1. Git Checkout
2. SonarQube Analysis
3. Quality Gate
4. Install npm
5. Trivy Scan
6. Build Docker Image
7. Create ECR repo
8. Login to ECR & tag image
9. Push image to ECR
10. Cleanup Images
Average stage times:
(Average full run time: ~4min 43s)
5s 1s 35s 749ms 25s 10s 2min 17s 5s 1s 57s 640ms
#2 Nov 30 16:14 No Changes
6s 1s 35s 749ms 25s 10s 2min 17s 5s 1s 57s 640ms
#1 Nov 30 16:13 No Changes
4s
Permalinks
Last build (#2), 0.29 sec ago
Last unsuccessful build (#1), 1 min 2 sec ago
Last completed build (#1), 1 min 2 sec ago
REST API
Jenkins 2.479.2



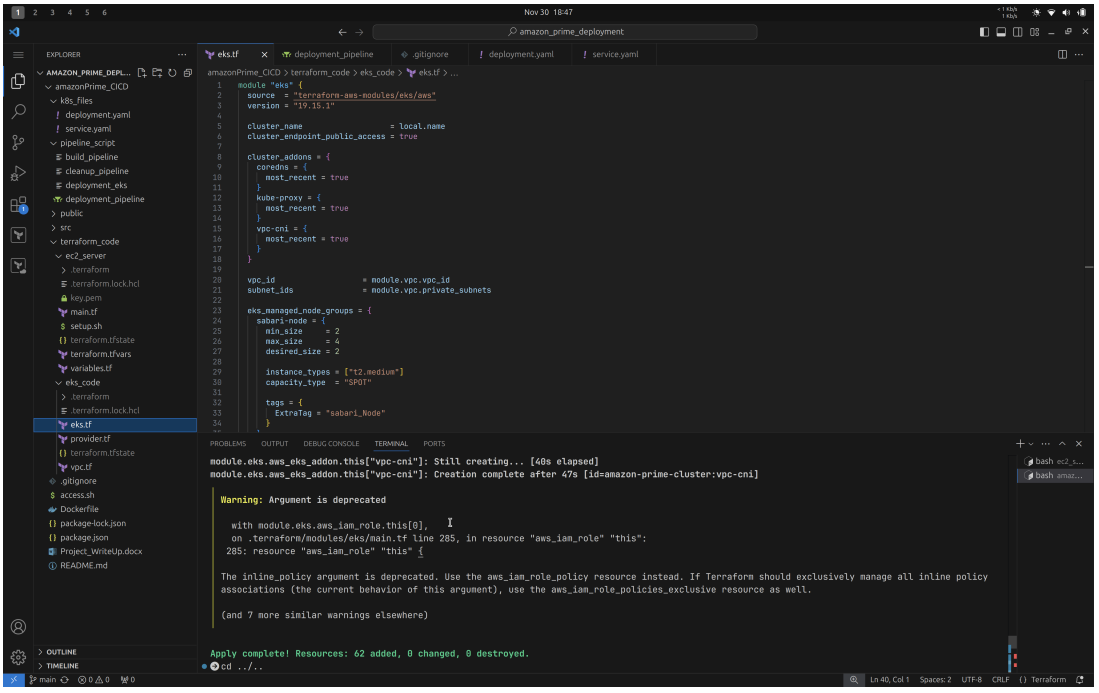



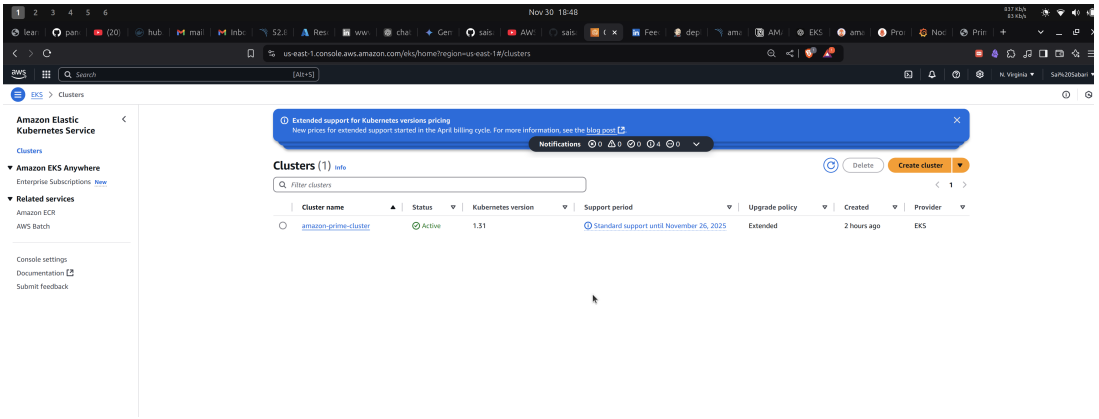

13. Amazon ECR
    Post build pipeline image will be pushed to Amazon ECR with build number and latest tag

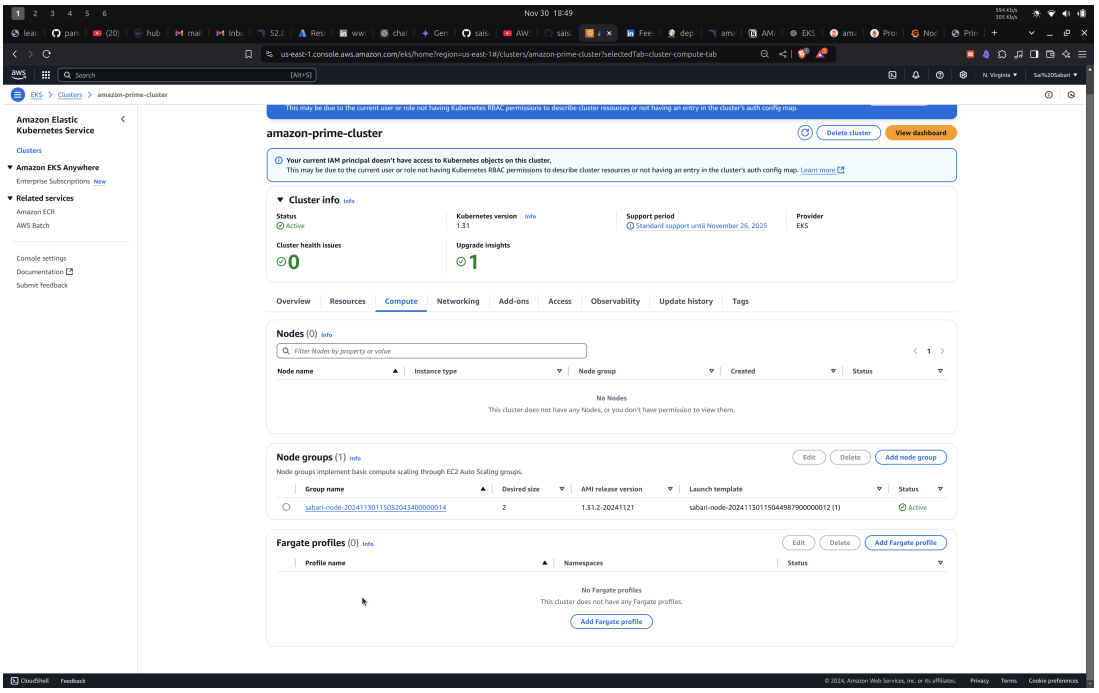


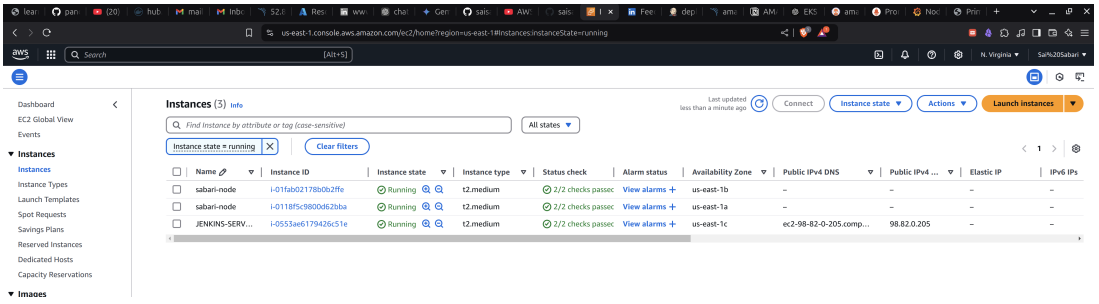

14. Terraform EKS

- in VS code cd to the terraform_code/eks_code
- Run “terraform init” & “terraform apply —auto-approve ”
- verify the resources created in EKS console and EC2 console



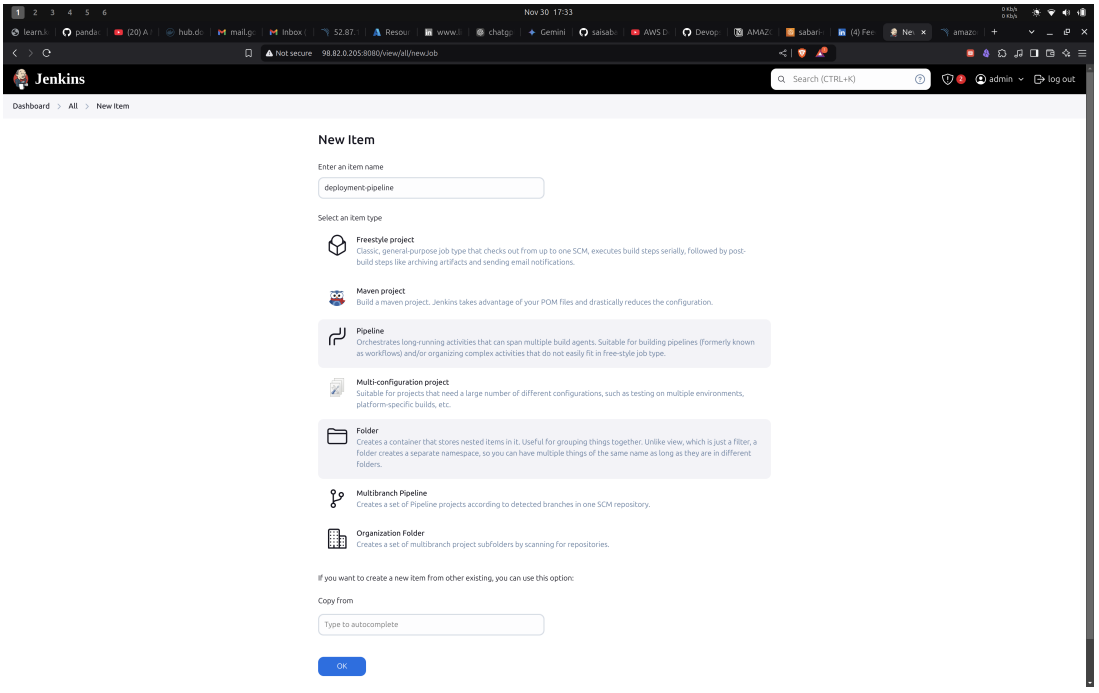


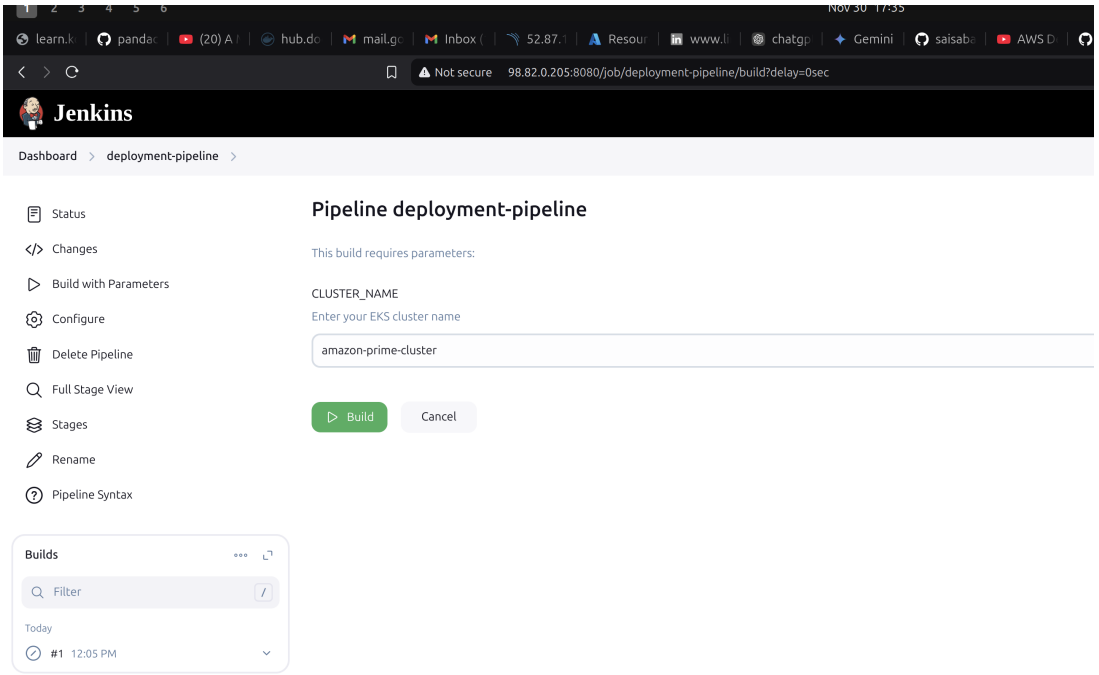



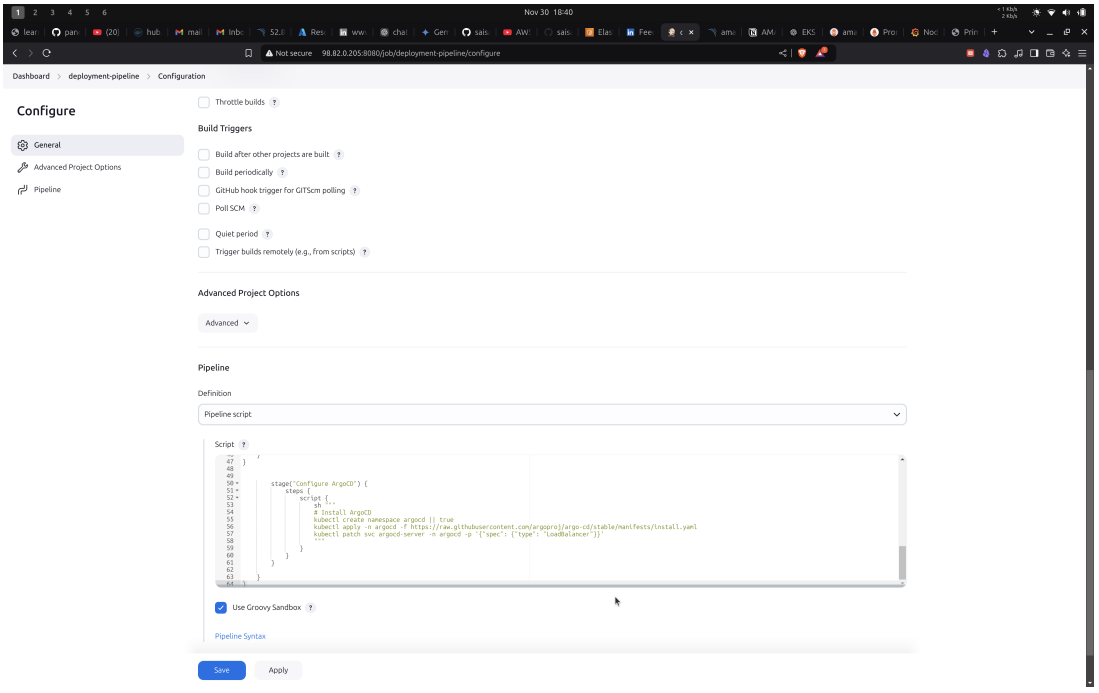



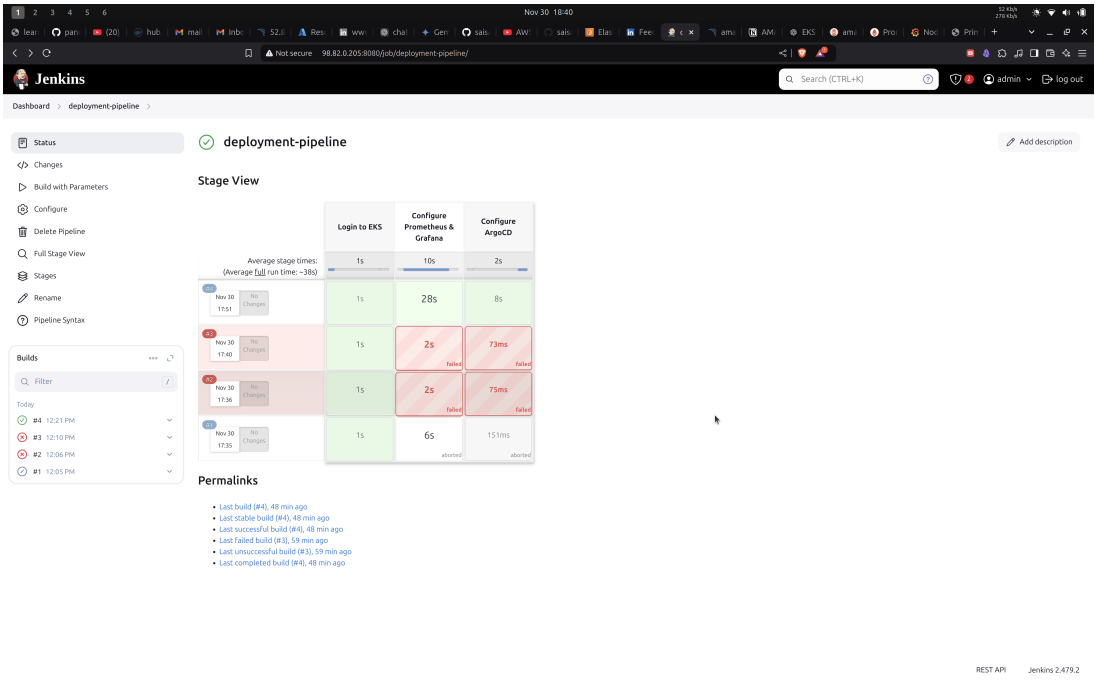

15. Deployment pipeline

- Goto Dashboard and create a new pipeline & name it “deployment-pipeline”
- copy the deployment-pipeline script from repo & paste it in Jenkins
- abort the first and Click “build parameters”


Jenkins
Dashboard > All > New Item
New Item
Enter an item name
deployment-pipeline
Select an item type
Freestyle project
Classic, general-purpose job type that checks out from up to one SCM, executes build steps serially, followed by post-build steps like archiving artifacts and sending email notifications.
Maven project
Build a maven project. Jenkins takes advantage of your POM files and drastically reduces the configuration.
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.
If you want to create a new item from other existing, you can use this option:
Copy from
Type to autocomplete
OK


Not secure 98.82.0.205:8080/job/deployment-pipeline/build?delay=0sec
Jenkins
Dashboard > deployment-pipeline >
Status
Changes
Build with Parameters
Configure
Delete Pipeline
Full Stage View
Stages
Rename
Pipeline Syntax
Builds
Filter
Today
#1 12:05 PM
Pipeline deployment-pipeline
This build requires parameters:
CLUSTER_NAME
Enter your EKS cluster name
amazon-prime-cluster
Build
Cancel



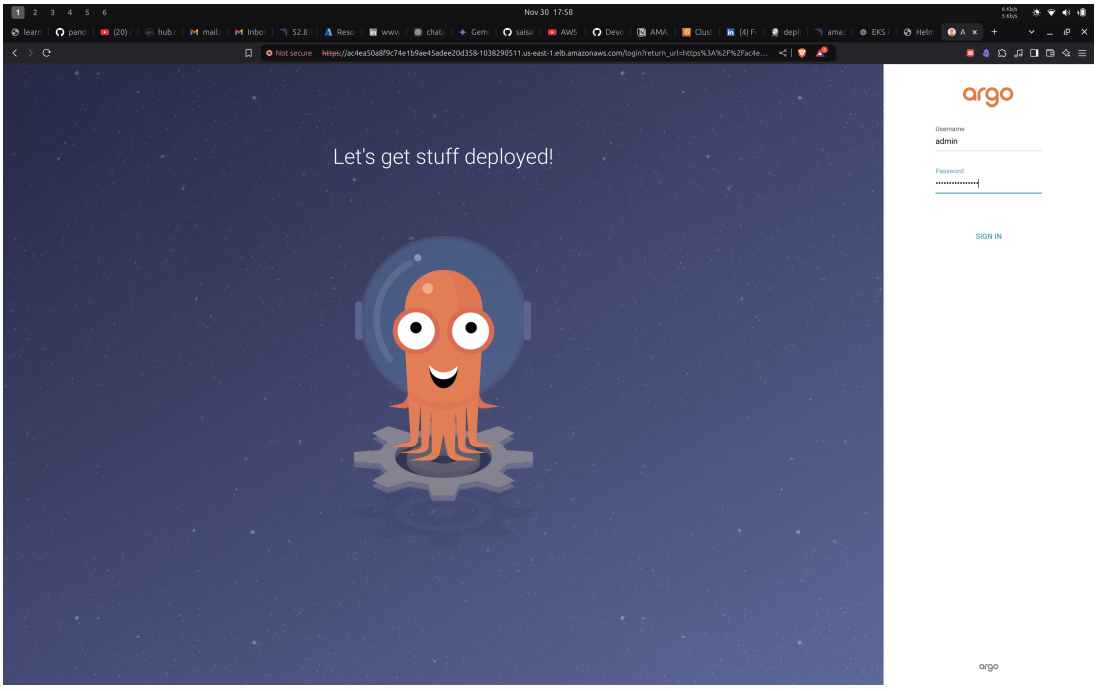

16. Access.sh

- SSH into jenkins server using key.pem and ubuntu@<public ip>
- create a file by entering in terminal “vi access.sh”
- Copy access.sh file from repo and paste it in the file created and save it by “:wq!”
- change access by “chmod a+x access.sh”
- execute access file by ./access.sh and enter

- find the load balancer details details in terminal

```
Default output format [None]:
Added new context arn:aws:eks:us-east-1:084086932810:cluster/amazon-prime-cluste
r to /home/ubuntu/.kube/config
------------------------
ArgoCD URL: ac4ea50a8f9c74e1b9ae45adee20d358-1038290511.us-east-1.elb.amazonaws.
com
ArgoCD User: admin
ArgoCD Initial Password: kfnJir28KZ8P2ARH

Prometheus URL: af72db763415c412b844d2ef1b366ad5-1141990950.us-east-1.elb.amazon
aws.com:9090

Grafana URL: a0fd5484f5b294f38882b3f0a32fc2fc-302995307.us-east-1.elb.amazonaws.
com
Grafana User: admin
Grafana Password: prom-operator
------------------------
```

17. ArgoCD
    - find the argoCD url and password in terminal , goto the url and enter creds
    - modify the ECR uri to your ECR uri under image in k8s_files/deployment.yaml
    - Click on “New App” and enter project details as shown and click create



CREATE CANCEL

GENERAL

Application Name
amazon-prime-app

Project Name
default

SYNC POLICY
Manual

☐ SET DELETION FINALIZER

SYNC OPTIONS

☐ SKIP SCHEMA VALIDATION
☐ PRUNE LAST
☐ RESPECT IGNORE DIFFERENCES
☐ AUTO-CREATE NAMESPACE
☐ APPLY OUT OF SYNC ONLY
☐ SERVER-SIDE APPLY

PRUNE PROPAGATION POLICY: foreground

☐ REPLACE ⚠
☐ RETRY

SOURCE

Repository URL
https://github.com/saisabari24/amazonPrime-CICID.git

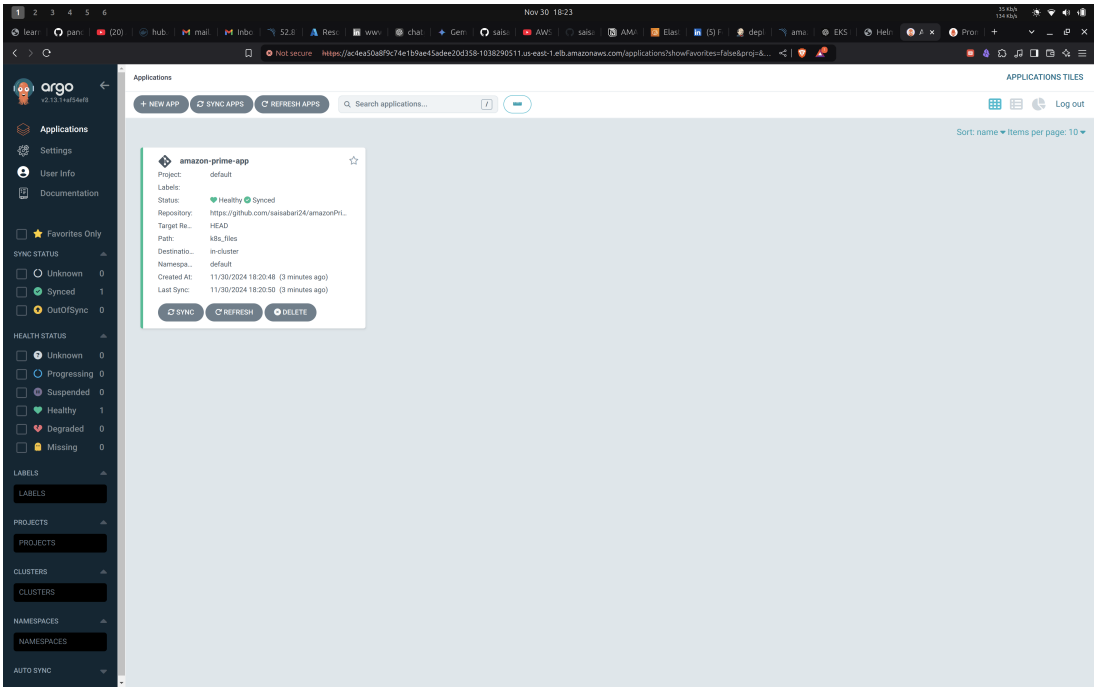
Revision
HEAD

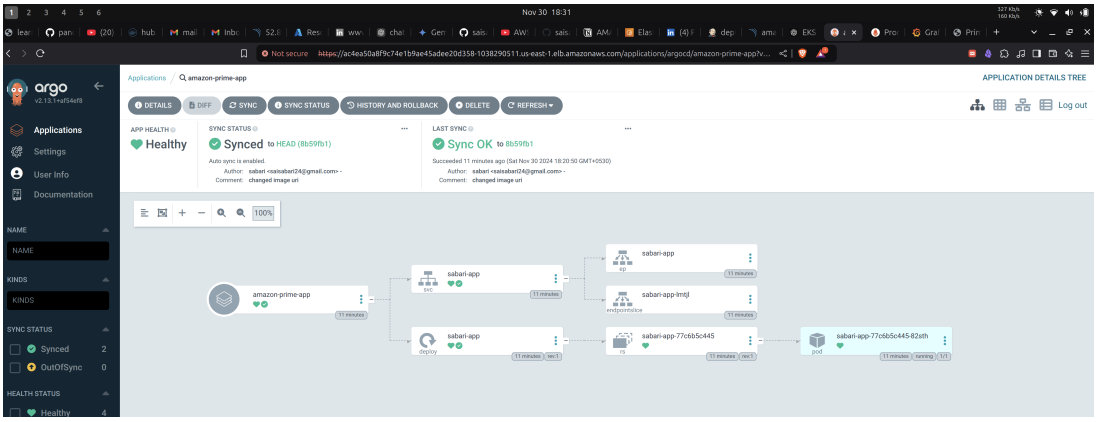
Path
k8s_files

DESTINATION

Cluster URL
https://kubernetes.default.svc

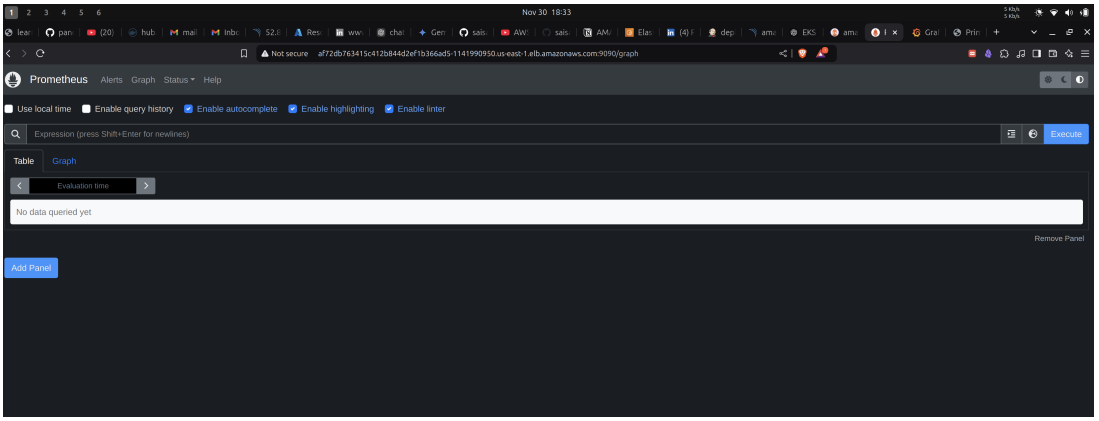
Namespace
default

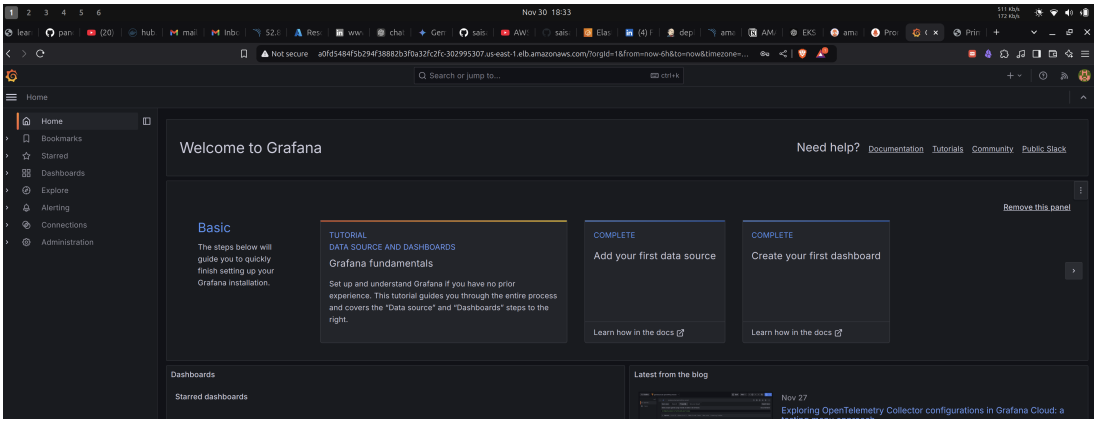


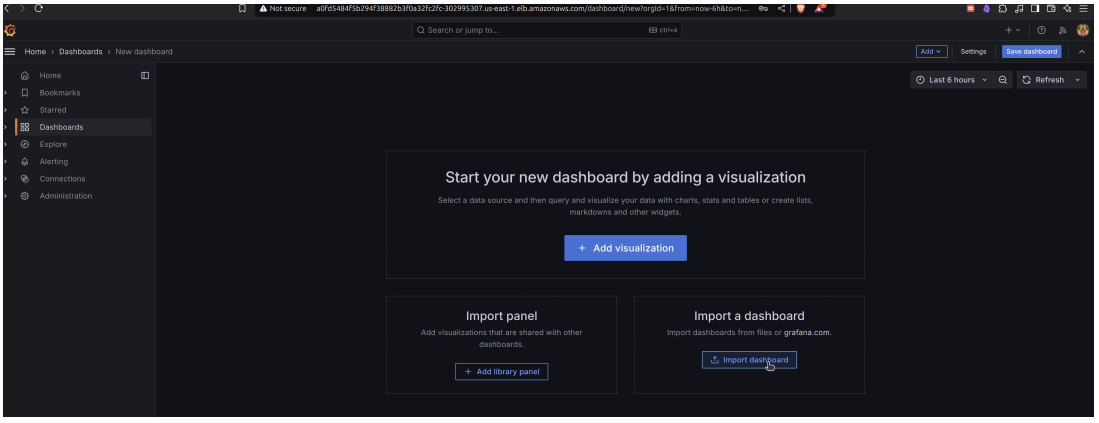



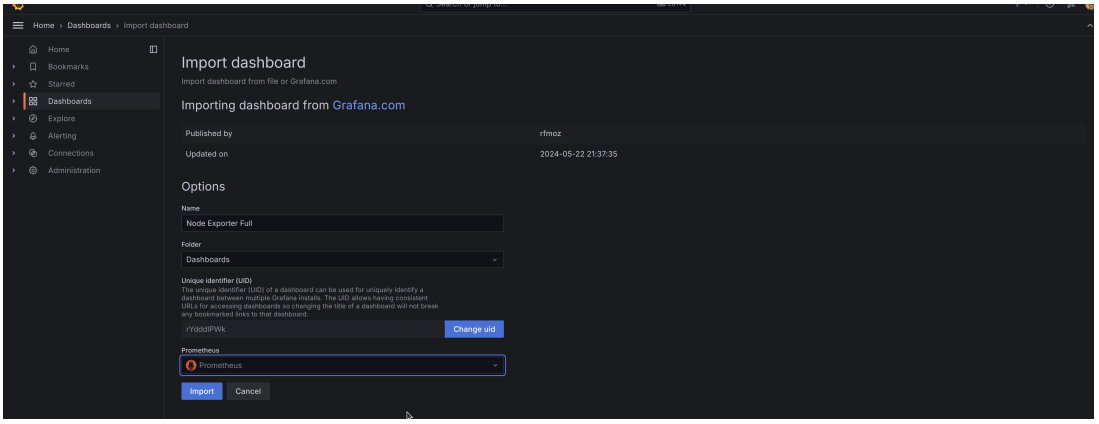

18. Promethues and Grafana

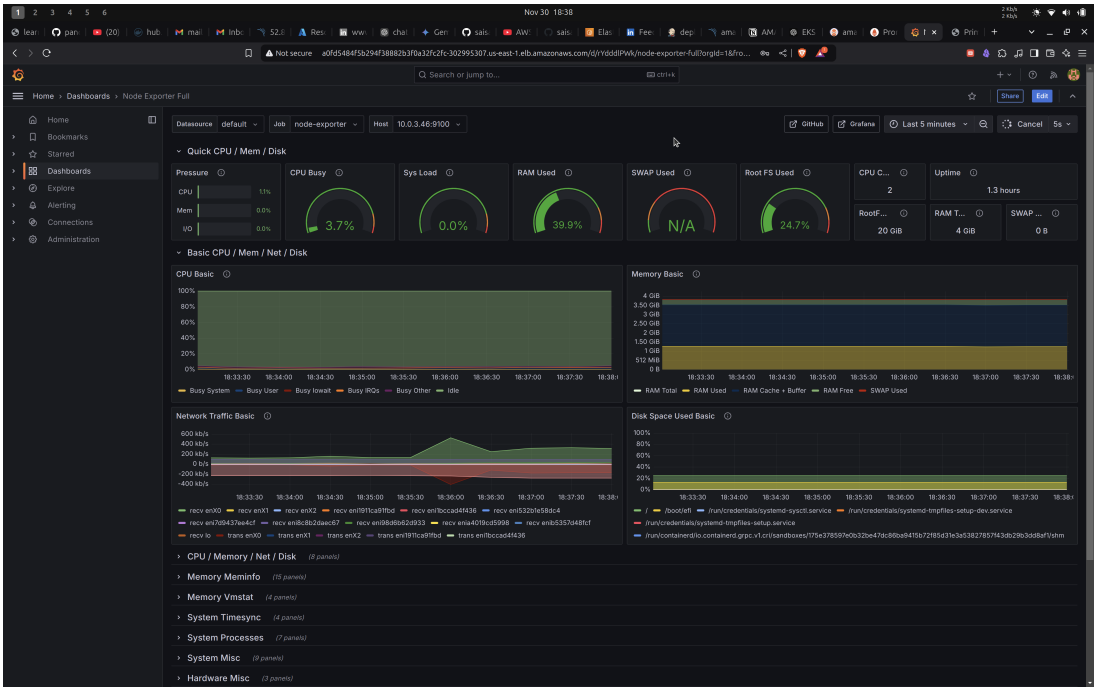
- goto Promethues and Grafana using the url & cred details seen in terminal
- in Grafana Click the “+” icon in the top right to add dashboard
- Now click “import dashboard” and enter the chart ID 1860 which import “Node Exporter Chart” dashboard and choose “Prometheus” as source and click Import
- modify grafana to meet your needs


Nov 30 18:33
Not secure af72db763415c412b844d2ef1b366ad5-1141990950.us-east-1.elb.amazonaws.com:9090/graph
Prometheus
Alerts
Graph
Status
Help
Use local time
Enable query history
Enable autocomplete
Enable highlighting
Enable linter
Expression (press Shift+Enter for newlines)
Execute
Table
Graph
Evaluation time
No data queried yet
Remove Panel
Add Panel


Nov 30 18:33
Not secure a0fd5484f5b294f38882b3f0a32fc2fc-302995307.us-east-1.elb.amazonaws.com/?orgId=1&from=now-6h&to=now&timezone=...
Search or jump to...
Home
Bookmarks
Starred
Dashboards
Explore
Alerting
Connections
Administration
Welcome to Grafana
Need help?
Documentation
Tutorials
Community
Public Slack
Remove this panel
Basic
The steps below will guide you to quickly finish setting up your Grafana installation.
TUTORIAL
DATA SOURCE AND DASHBOARDS
Grafana fundamentals
Set up and understand Grafana if you have no prior experience. This tutorial guides you through the entire process and covers the "Data source" and "Dashboards" steps to the right.
COMPLETE
Add your first data source
Learn how in the docs
COMPLETE
Create your first dashboard
Learn how in the docs
Dashboards
Starred dashboards
Latest from the blog
Nov 27
Exploring OpenTelemetry Collector configurations in Grafana Cloud: a

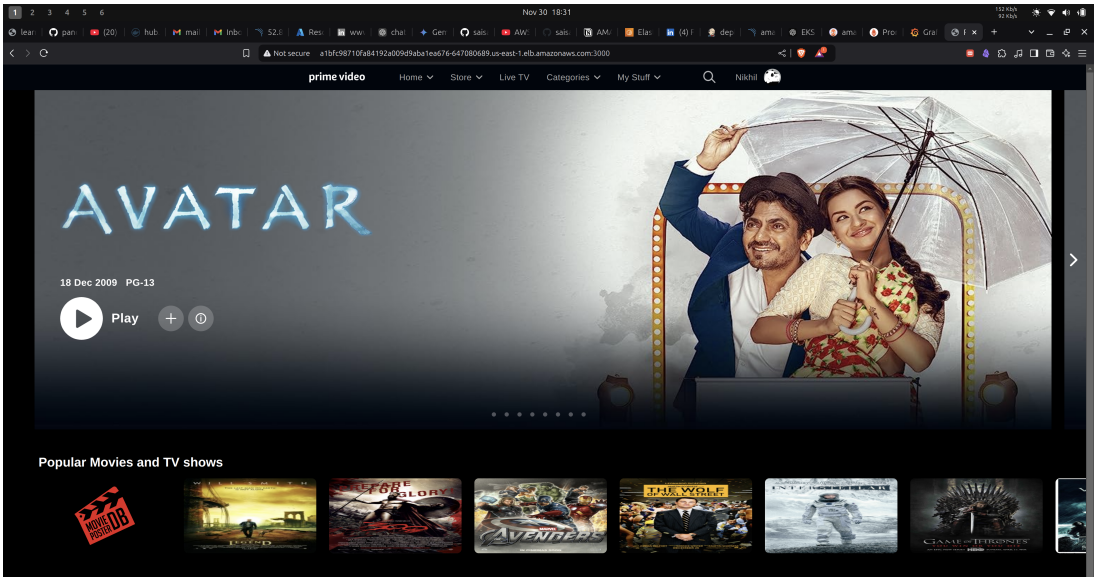

Not secure a0fd5484f5b294f38882b3f0a32fc2fc-302995307.us-east-1.elb.amazonaws.com/dashboard/new?orgId=1&from=now-6h&to=n...
Search or jump to...
Home > Dashboards > New dashboard
Add
Settings
Save dashboard
Last 6 hours
Refresh
Home
Bookmarks
Starred
Dashboards
Explore
Alerting
Connections
Administration
Start your new dashboard by adding a visualization
Select a data source and then query and visualize your data with charts, stats and tables or create lists, markdowns and other widgets.
+ Add visualization
Import panel
Add visualizations that are shared with other dashboards.
+ Add library panel
Import a dashboard
Import dashboards from files or grafana.com.
Import dashboard


Home > Dashboards > Import dashboard
Home
Bookmarks
Starred
Dashboards
Explore
Alerting
Connections
Administration
Import dashboard
Import dashboard from file or Grafana.com
Importing dashboard from Grafana.com
Published by
rfmoz
Updated on
2024-05-22 21:37:35
Options
Name
Node Exporter Full
Folder
Dashboards
Unique identifier (UID)
The unique identifier (UID) of a dashboard can be used for uniquely identify a dashboard between multiple Grafana installs. The UID allows having consistent URLs for accessing dashboards so changing the title of a dashboard will not break any bookmarked links to that dashboard.
rYdddlPWk
Change uid
Prometheus
Prometheus
Import
Cancel




19. Application endpoint

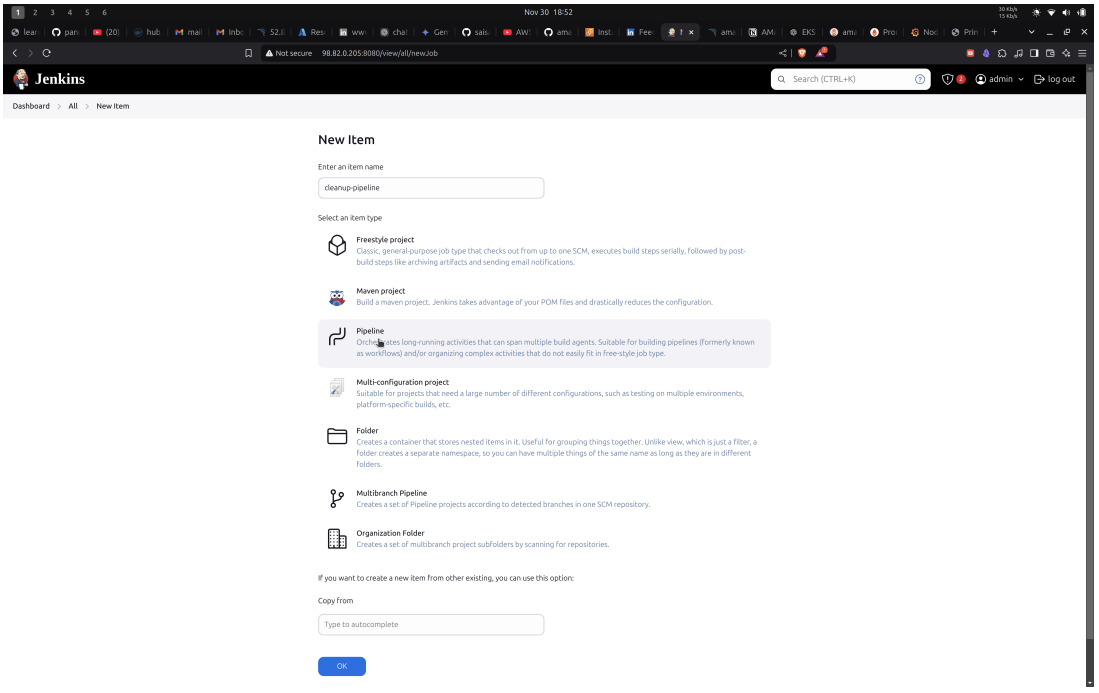
- to get the application end point run “kubectl get svc” and see the application endpoint as loadbalancer
- also can fetch it from ArgoCD by clicking the application loadbalancer icon

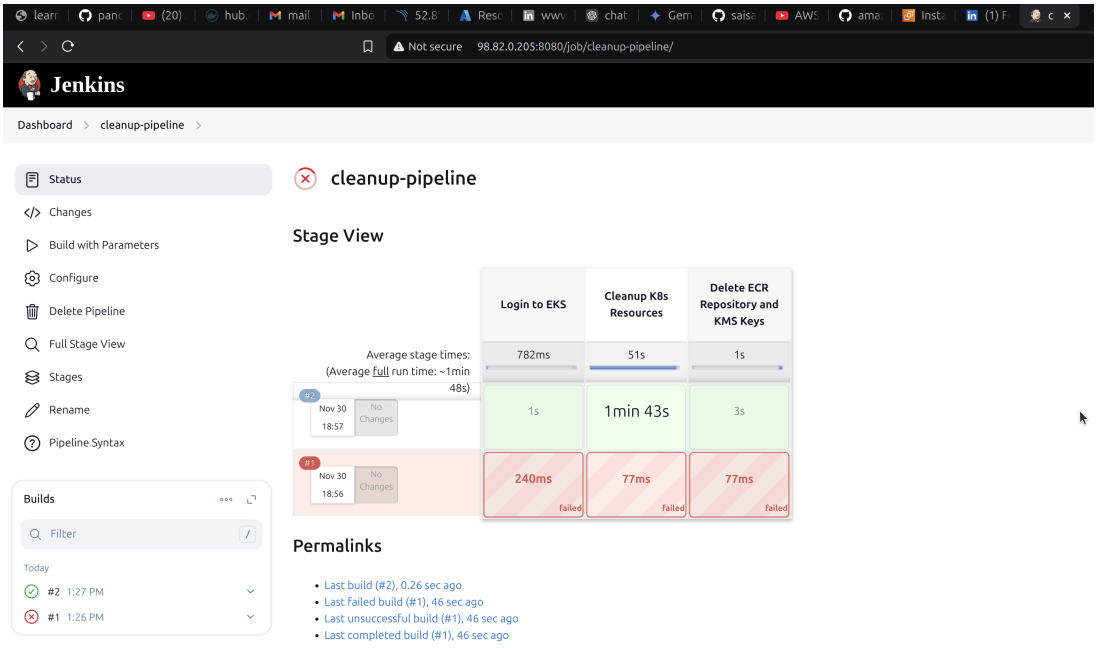



20. CleanUp-pipeline

- run the cleanup-pipeline script to delete the applications installed in eks and nodes

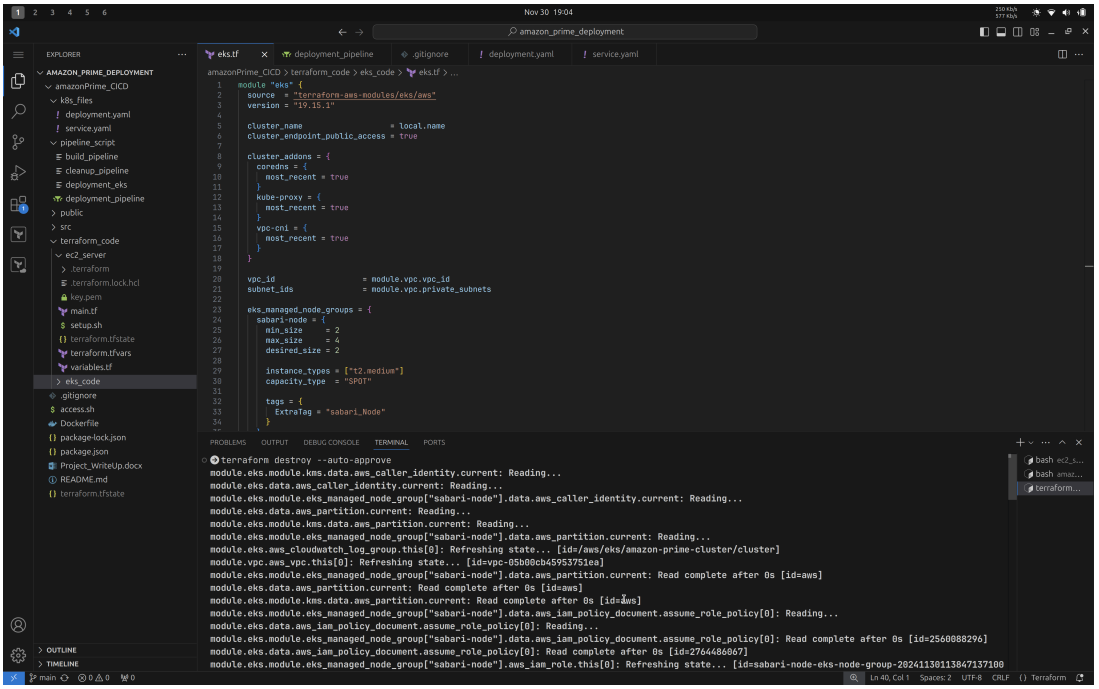
- go to jenkins create a pipeline called “cleanup-pipeline”
- copy script from “cleanup-pipeline” and paste it and click build now
- this deletes the helm promethues grafana argocd




21. Terraform Destroy

- goto terraform_code/ec2_server and enter “terraform destroy —auto-approve”

this takes 2-3 minutes to delete jenkins server

- goto terraform_code/eks_code and enter “terraform destroy —auto-approve”
  his takes 5-10 minutes to delete eks cluster & nodes & other resources created