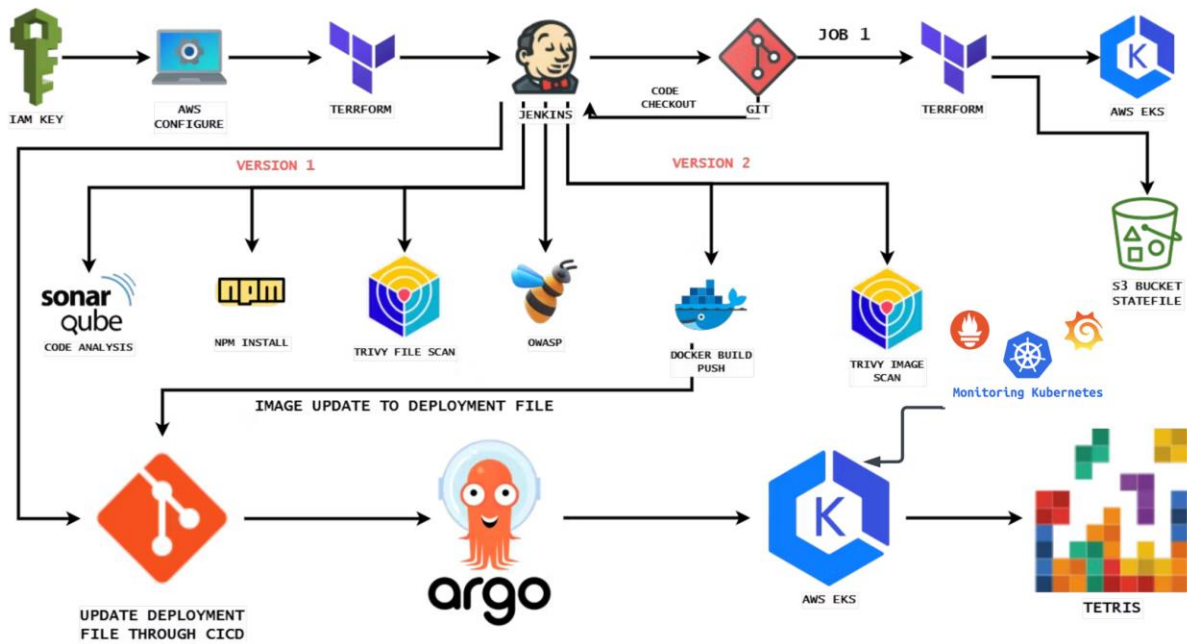


Automating Tetris Deployments Using DevSecOps



GitHub REPOSITORIES :

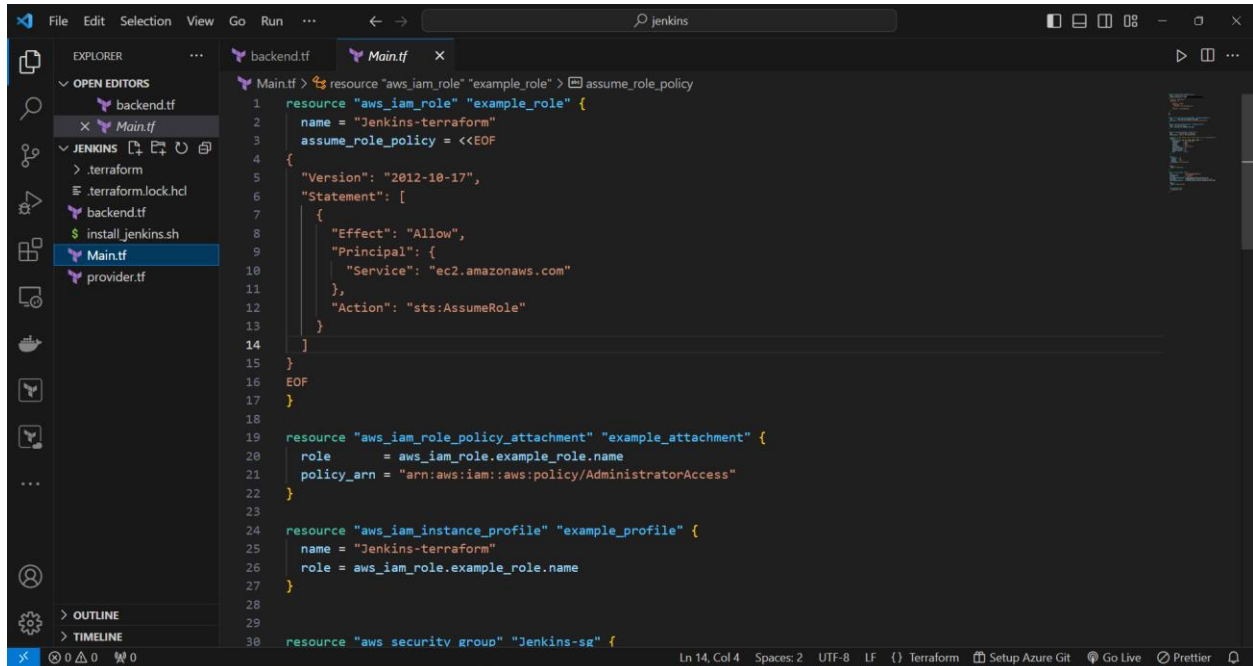
TETRIS-VERSION1 : <https://github.com/Narasimha76/Tetrisv1>

TETRIS-VERSION2 : <https://github.com/Narasimha76/TetrisV2>

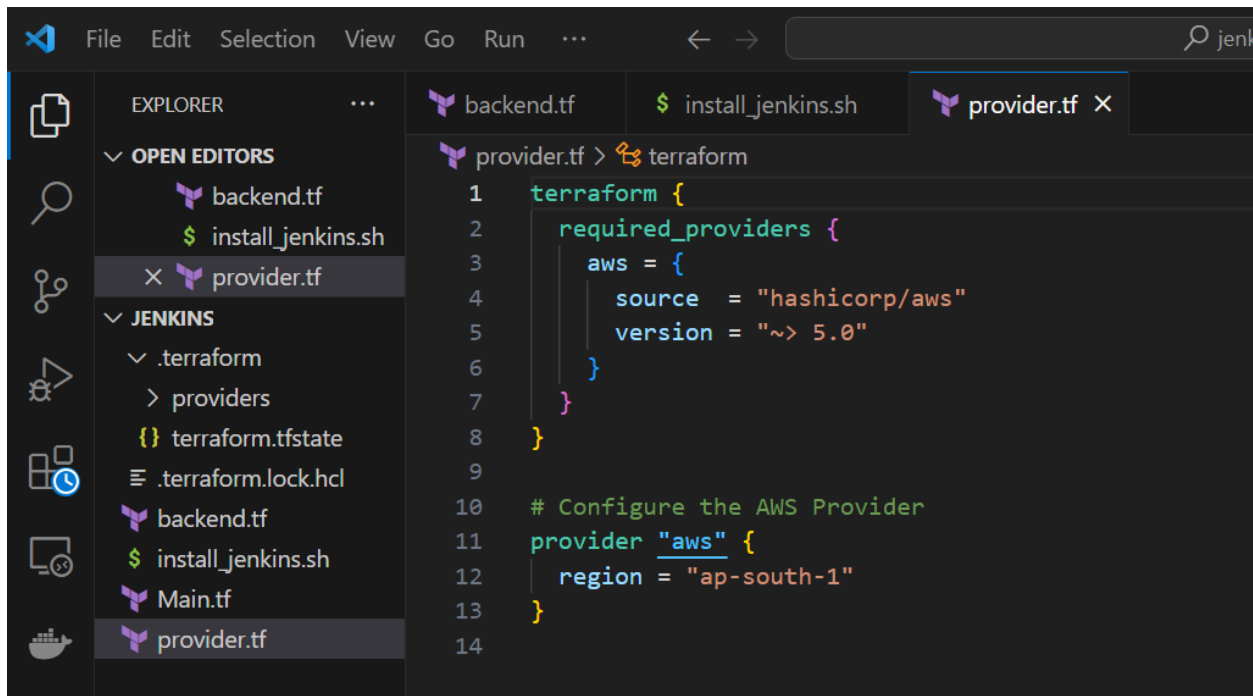
TETRIS_MANIFEST : <https://github.com/Narasimha76/Tetris-manifest>

Step1: Terraform Provisioning

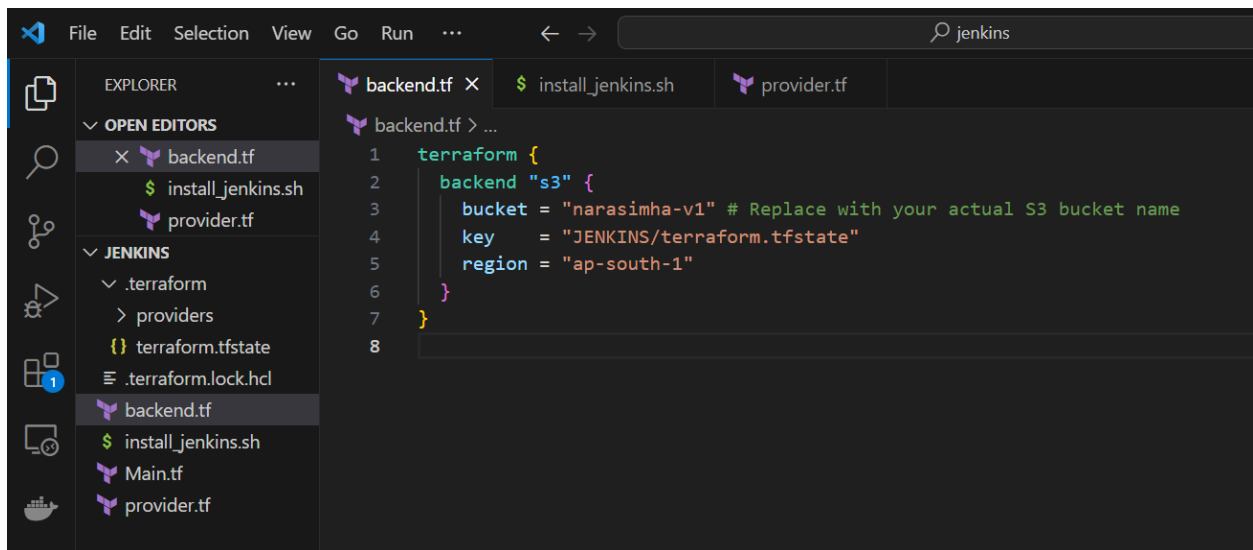
- Provision an AWS EC2 instance using Terraform, incorporating a bash script. This script orchestrates the installation of Docker, SonarQube, Trivy, OWASP, Terraform, AWS CLI, Jenkins with JDK, and Kubectl on the EC2 instance.



```
1 resource "aws_iam_role" "example_role" {
2   name = "Jenkins-terraform"
3   assume_role_policy = <<EOF
4
5   "Version": "2012-10-17",
6   "Statement": [
7     {
8       "Effect": "Allow",
9       "Principal": {
10        "Service": "ec2.amazonaws.com"
11      },
12      "Action": "sts:AssumeRole"
13    }
14  ]
15 }
16 EOF
17 }
18
19 resource "aws_iam_role_policy_attachment" "example_attachment" {
20   role       = aws_iam_role.example_role.name
21   policy_arn = "arn:aws:iam::aws:policy/AdministratorAccess"
22 }
23
24 resource "aws_iam_instance_profile" "example_profile" {
25   name = "Jenkins-terraform"
26   role = aws_iam_role.example_role.name
27 }
28
29
30 resource "aws_security_group" "Jenkins-sg" {
```

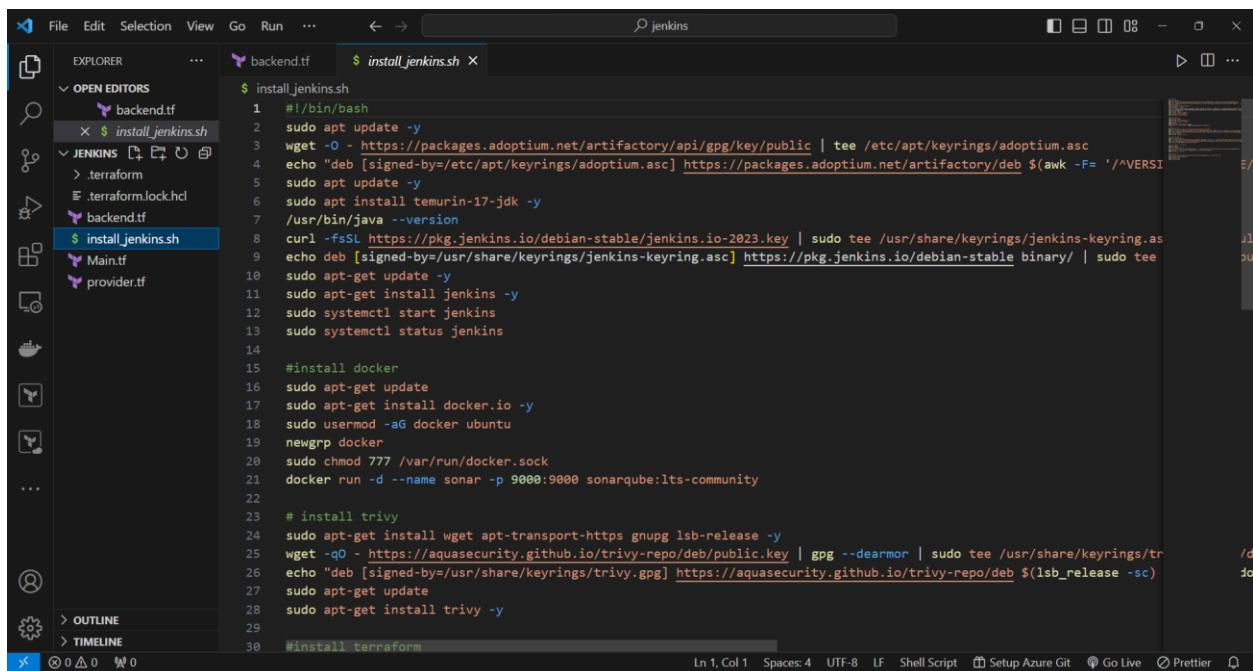


```
1 terraform {
2   required_providers {
3     aws = {
4       source = "hashicorp/aws"
5       version = "~> 5.0"
6     }
7   }
8 }
9
10 # Configure the AWS Provider
11 provider "aws" {
12   region = "ap-south-1"
13 }
14
```



This screenshot shows the VS Code interface with a Terraform configuration file named `backend.tf` open. The Explorer sidebar on the left shows the project structure, including `install_jenkins.sh` and `provider.tf`. The main editor displays the following Terraform code:

```
1 terraform {
2   backend "s3" {
3     bucket = "narasimha-v1" # Replace with your actual S3 bucket name
4     key    = "JENKINS/terraform.tfstate"
5     region = "ap-south-1"
6   }
7 }
8
```



This screenshot shows the VS Code interface with a shell script named `install_jenkins.sh` open. The Explorer sidebar on the left shows the project structure, including `backend.tf` and `provider.tf`. The main editor displays the following shell script code:

```
1 #!/bin/bash
2 sudo apt update -y
3 wget -O - https://packages.adoptium.net/artifactory/api/gpg/key/public | tee /etc/apt/keyrings/adoptium.asc
4 echo "deb [signed-by=/etc/apt/keyrings/adoptium.asc] https://packages.adoptium.net/artifactory/deb $(awk -F= '/^VERSION/' {print $2})" | tee /etc/apt/sources.list.d/adoptium.list
5 sudo apt update -y
6 sudo apt install temurin-17-jdk -y
7 /usr/bin/java --version
8 curl -fsSL https://pkg.jenkins.io/debian-stable/jenkins.io-2023.key | sudo tee /usr/share/keyrings/jenkins-keyring.asc
9 echo deb [signed-by=/usr/share/keyrings/jenkins-keyring.asc] https://pkg.jenkins.io/debian-stable binary/ | sudo tee /etc/apt/sources.list.d/jenkins.list
10 sudo apt-get update -y
11 sudo apt-get install jenkins -y
12 sudo systemctl start jenkins
13 sudo systemctl status jenkins
14
15 #install docker
16 sudo apt-get update
17 sudo apt-get install docker.io -y
18 sudo usermod -aG docker ubuntu
19 newgrp docker
20 sudo chmod 777 /var/run/docker.sock
21 docker run -d --name sonar -p 9000:9000 sonarqube:lts-community
22
23 # install trivy
24 sudo apt-get install wget apt-transport-https gnupg lsb-release -y
25 wget -qO - https://aquasecurity.github.io/trivy-repo/deb/public.key | gpg --dearmor | sudo tee /usr/share/keyrings/trivy.gpg
26 echo "deb [signed-by=/usr/share/keyrings/trivy.gpg] https://aquasecurity.github.io/trivy-repo/deb $(lsb_release -sc) main" | sudo tee /etc/apt/sources.list.d/trivy.list
27 sudo apt-get update
28 sudo apt-get install trivy -y
29
30 #install terraform
```

- Apply commands `terraform init` and `terraform apply -auto-approve`
- Then the resource is creating in the aws.

Step2: Jenkins Pipeline for EKS

- Create a pipeline to provision an EKS cluster and node group using Terraform.
- Check all the scripts that are installed in the ec2 Server

```
root@Narasimha: ~
ubuntu@ip-172-31-40-200: ~
ubuntu@ip-172-31-40-200: ~

ubuntu@ip-172-31-40-200:~$ docker --version
Docker version 24.0.5, build 24.0.5-0ubuntu1~22.04.1
ubuntu@ip-172-31-40-200:~$ jenkins --version
2.426.3
ubuntu@ip-172-31-40-200:~$ terraform --version
Terraform v1.7.1
on linux_amd64
ubuntu@ip-172-31-40-200:~$ trivy --version
Version: 0.48.3
ubuntu@ip-172-31-40-200:~$ aws --version
aws-cli/2.15.15 Python/3.11.6 Linux/6.2.0-1017-aws exe/x86_64.ubuntu.22 prompt/off
ubuntu@ip-172-31-40-200:~$ kubectl
kubectl controls the Kubernetes cluster manager.

Find more information at: https://kubernetes.io/docs/reference/kubectl/

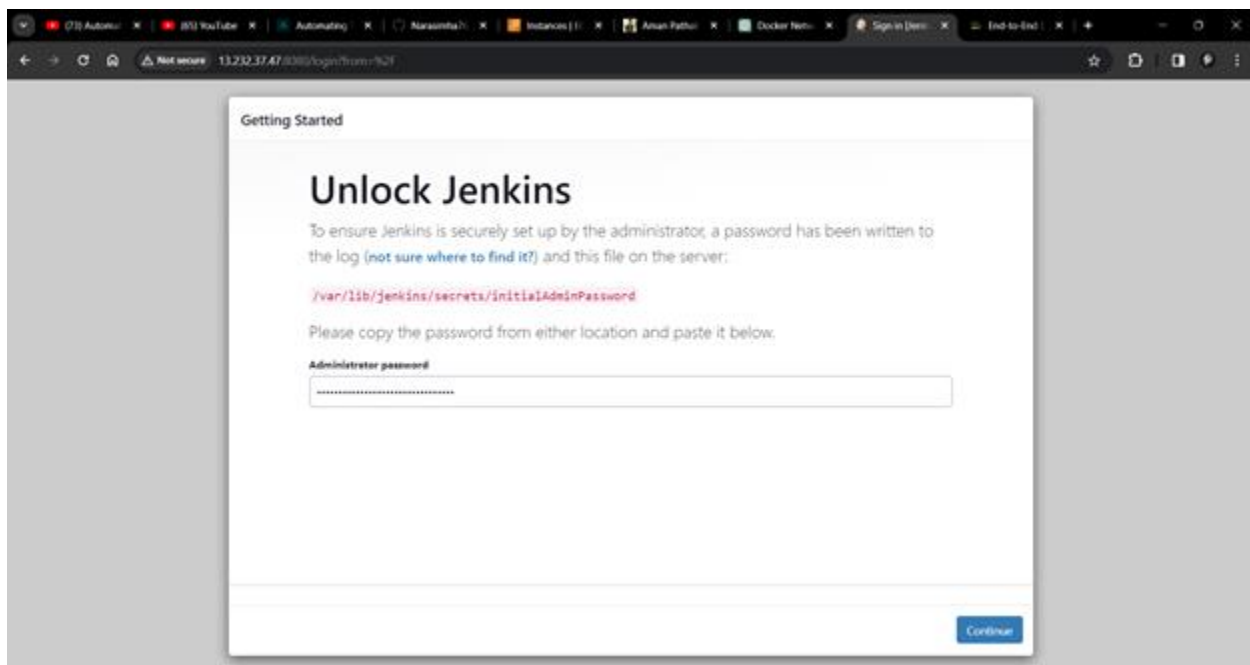
Basic Commands (Beginner):
  create      Create a resource from a file or from stdin
  expose       Take a replication controller, service, deployment or pod and expose it as a new Kubernetes service
  run         Run a particular image on the cluster
  set         Set specific features on objects

Basic Commands (Intermediate):
  explain     Get documentation for a resource
  get         Display one or many resources
  edit        Edit a resource on the server
  delete      Delete resources by file names, stdin, resources and names, or by resources and label selector

Deploy Commands:
  rollout     Manage the rollout of a resource
  scale       Set a new size for a deployment, replica set, or replication controller
  autoscale   Auto-scale a deployment, replica set, stateful set, or replication controller

Cluster Management Commands:
  certificate Modify certificate resources
```

- Configure the Jenkins



- Create a pipeline to provision an EKS cluster and node group using Terraform.

Dashboard > terraform-eks > Configuration

Configure

General

Advanced Project Options

Pipeline

☐ Pipeline speed/durability override ?

☐ Preserve stashes from completed builds ?

☒ This project is parameterised ?

Choice Parameter ?

Name ?

action

Choices ?

apply
destroy

Description ?

Plain text [Preview](#)

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

- Eks provision terraform code is present in git-hub

```

1 pipeline{
2   agent any
3   stages {
4     stage('Checkout from Git'){
5       steps{
6         git branch: 'main', url: 'https://github.com/Narasimha76/Tetrisv1.git'
7       }
8     }
9     stage('Terraform version'){
10      steps{
11        sh 'terraform --version'
12      }
13    }
14    stage('Terraform init'){
15      steps{
16        dir('Eks-terraform') {
17          sh 'terraform init'
18        }
19      }
20    }
21    stage('Terraform validate'){
22      steps{
23        dir('Eks-terraform') {
24          sh 'terraform validate'
25        }
26      }
27    }
28    stage('Terraform plan'){
29      steps{
30        dir('Eks-terraform') {
31          sh 'terraform plan'
32        }
33      }
34    }
35    stage('Terraform apply/destroy'){
36      steps{
37        dir('Eks-terraform') {
38          sh 'terraform ${action} --auto-approve'
39        }
40      }
41    }
42  }
43 }

```

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

Pipeline terraform-eks

This build requires parameters:

action

apply

Build

Cancel

terraform-eks

Add description

Disable Project

Stage View

Average stage times:
(Average full run time: ~10min 8s)

	Checkout from Git	Terraform version	Terraform init	Terraform validate	Terraform plan	Terraform apply/destroy
	2s	641ms	5s	1s	2s	4min 55s
#2 Jan 31 15:14 3 commits	976ms	348ms	8s	3s	4s	9min 50s
#1 Jan 31 15:09 No Changes	4s	935ms	1s failed	90ms failed	54ms failed	55ms failed

- Eks cluster has been created in AWS

EKS > Clusters

New Kubernetes versions are available for 1 cluster.

Clusters (1) Info

Delete

Add cluster ▼

Filter clusters

< 1 >

	Cluster name	Status	Kubernetes version	Support type	Provider
	EKS_CLOUD	Active	1.28 Update now	Standard support until November 2024	EKS

Instances (2) [Info](#)

Find Instance by attribute or tag (case-sensitive)

Any state

Instance state = running X Clear filters

<input type="checkbox"/>	Name	Instance ID	Instance state	Instance type	Status check	Alarm status	Availability Zone	Public IPv4 DNS
<input type="checkbox"/>	Jenkins-argo	i-07f68d8f4ddb1d54c	Running	t2.medium	2/2 checks passed	View alarms	ap-south-1a	ec2-13-232-37-4
<input type="checkbox"/>		i-0efc2f0c750af55e1	Running	t2.medium	Initializing	View alarms	ap-south-1b	ec2-43-204-235-

Step3: Creating Full DevSecOps Pipeline

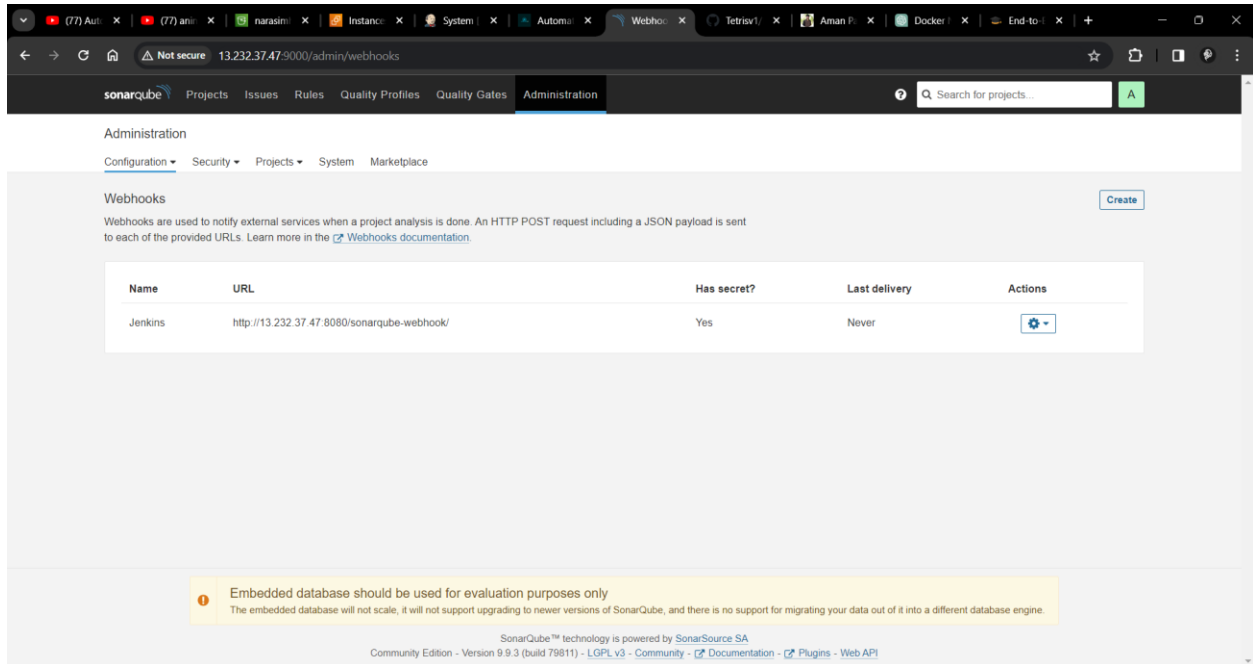
1. Checkout the repository.
2. SonarQube analysis, including quality gate.
3. Install dependencies using npm.
4. OWASP Scan with Dependency Check.
5. Trivy Scan for file analysis.
6. Build Docker image and push to Docker Hub.
7. Trivy scan Docker image.
8. Trigger another pipeline (Manifest Pipeline).

We need some plugins to complete this process

1. Eclipse Temurin installer
2. Sonarqube Scanner
3. NodeJs
4. Owasp Dependency-Check
5. Docker, Docker Commons, Docker Pipeline, Docker API, Docker-build-step

Add the tools in the Jenkins UI for JDK, Docker, NodeJS, Sonarqube, OWSAP(DependencyCheck)

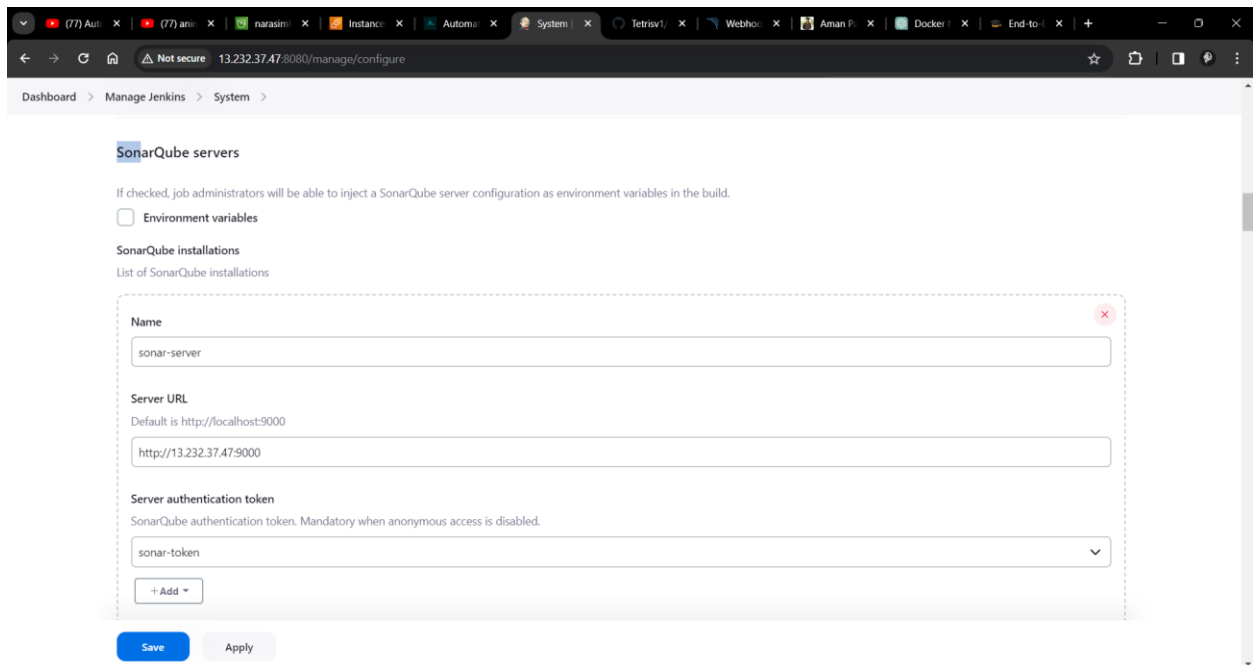
- Configure the Sonarqube by creating user token and that token to the Jenkins credentials and taken the credentials and create the system for the Jenkins in the Jenkins UI.



The screenshot shows the SonarQube Administration interface, specifically the 'Webhooks' section. The breadcrumb trail is 'Administration > Configuration > Security > Projects > System > Marketplace'. The 'Webhooks' section has a 'Create' button and a description: 'Webhooks are used to notify external services when a project analysis is done. An HTTP POST request including a JSON payload is sent to each of the provided URLs. Learn more in the [Webhooks documentation](#).' Below this is a table with one entry for 'Jenkins'.

Name	URL	Has secret?	Last delivery	Actions
Jenkins	http://13.232.37.47:8080/sonarqube-webhook/	Yes	Never	Configure

At the bottom, there is a yellow warning box: 'Embedded database should be used for evaluation purposes only. The embedded database will not scale, it will not support upgrading to newer versions of SonarQube, and there is no support for migrating your data out of it into a different database engine.' Below the warning box, it says 'SonarQube™ technology is powered by SonarSource SA. Community Edition - Version 9.9.3 (build 79811) - LGPL v3 - Community - [Documentation](#) - [Plugins](#) - [Web API](#)'.




The screenshot shows the SonarQube Configuration interface, specifically the 'SonarQube servers' section. The breadcrumb trail is 'Dashboard > Manage Jenkins > System >'. The 'SonarQube servers' section has a description: 'If checked, job administrators will be able to inject a SonarQube server configuration as environment variables in the build.' Below this is a checkbox labeled 'Environment variables'. The 'SonarQube installations' section has a description: 'List of SonarQube installations'. Below this is a form with three fields: 'Name' (sonar-server), 'Server URL' (http://13.232.37.47:9000), and 'Server authentication token' (sonar-token). There is a '+ Add' button and 'Save' and 'Apply' buttons at the bottom.


- Now let's create an pipeline

Enter an item name


>> 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.

**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.


- Add the Jenkins Script in the pipeline which is present in my git repo

Dashboard > Tetris-V1 > Configuration

Configure

 General

 Advanced Project Options

 Pipeline

Pipeline

Definition

Pipeline script

Script ?

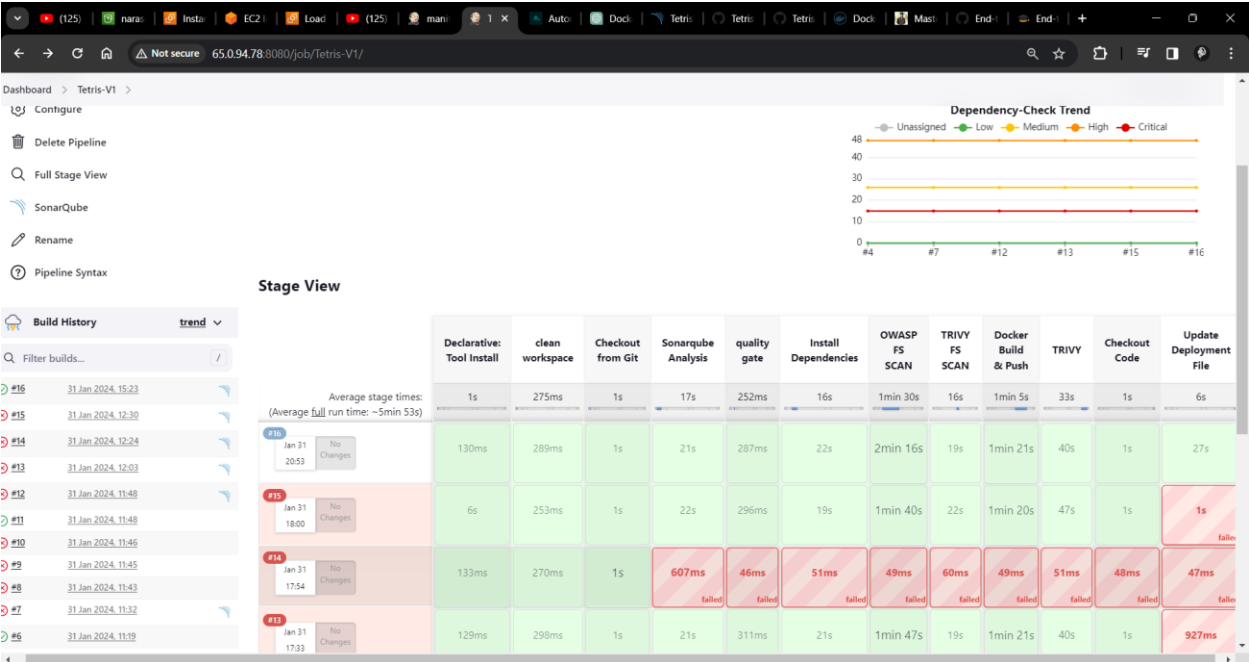
```
1 pipeline(  
2   agent any  
3   tools(  
4     jdk 'jdk17'  
5     nodejs 'node16'  
6   )  
7   environment {  
8     SCANNER_HOME=tool 'sonar-scanner'  
9   }  
10  stages {  
11    stage('clean workspace'){  
12      steps(  
13        cleanWs()  
14      )  
15    }  
16    stage('Checkout from Git'){  
17      steps(  
18        git branch: 'main', url: 'https://github.com/larasimha76/TetrisV1.git'  
19      )  
20    }  
21    stage("Sonarqube Analysis "){  
22      steps(  
23        withSonarQubeEnv('sonar-server') {  
24          sh "'${SCANNER_HOME}/bin/sonar-scanner' -Dsonar.projectName=TetrisVersion1.0 \\  
25            -Dsonar.projectKey=TetrisVersion1.0 '"  
26        }  
27      )  
28    }  
29    stage("quality gate"){  
30      steps {
```

try sample Pipeline...

Save

Apply

- Now run the pipeline

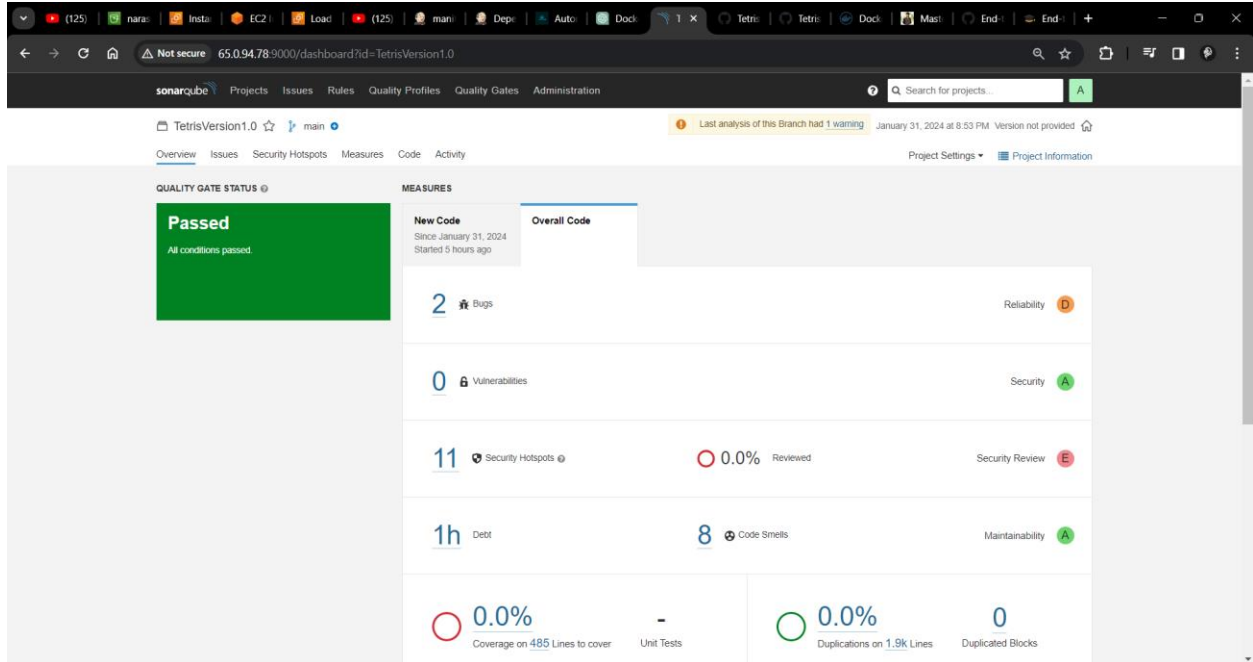


- Dependency-Check Results for the pipeline

The screenshot shows the Jenkins Pipeline Dependency-Check Results page for build #16. The page displays a table of vulnerabilities found in the pipeline's dependencies. The table is organized into columns for 'File Name', 'Vulnerability', 'Severity', and 'Weakness'. The 'Severity' column shows a distribution of 15 High, 48 Medium, and 26 Low severity issues. The 'Weakness' column lists the specific CVEs and CWEs associated with each vulnerability. The table is paginated, showing 1 of 9 results.

File Name	Vulnerability	Severity	Weakness
@babel/preset-env:7.12.1	NVD CVE-2023-45133	High	CWE-697
ansi-html:0.0.7	NVD CVE-2021-23424	High	NVD-CWE-noinfo
ansi-regex:4.1.0	NVD CVE-2021-3807	High	CWE-1333
ansi-regex:5.0.0	NVD CVE-2021-3807	High	CWE-1333
async:2.6.3	NVD CVE-2021-43138	High	CWE-1321
browsersify-sign:4.2.1	NVD CVE-2023-46234	High	CWE-347
browserslist:4.14.2	NVD CVE-2021-23364	Medium	CWE-1333
browserslist:4.16.3	NVD CVE-2021-23364	Medium	CWE-1333
css-what:3.4.2	OSSINDEX CVE-2022-21222	High	CWE-1333
decode-uri-component:0.2.0	NVD CVE-2022-38778	Medium	CWE-20

- Sonarqube check for the pipeline



Step4: Create a Manifest Pipeline

- Add this code to the Jenkins pipeline

Narasimha76 Create Jenkinsfile

```

1  pipeline{
2      agent any
3      environment {
4          GIT_REPO_NAME = "Tetris-manifest"
5          GIT_USER_NAME = "Narasimha76"
6      }
7      stages {
8          stage('Checkout Code') {
9              steps {
10                 git branch: 'main', url: 'https://github.com/Narasimha76/Tetris-manifest.git'
11             }
12         }
13         stage('Update Deployment File') {
14             steps {
15                 script {
16                     withCredentials([string(credentialsId: 'github', variable: 'GITHUB_TOKEN')]) {
17                         NEW_IMAGE_NAME = "narasimhaswamy76/tetrisv1:latest" #update your image here
18                         sh "sed -i 's|image: .*|image: $NEW_IMAGE_NAME|' deployment.yml"
19                         sh 'git add deployment.yml'
20                         sh "git commit -m 'Update deployment image to $NEW_IMAGE_NAME'"
21                         sh "git push https://$GITHUB_TOKEN@github.com:$GIT_USER_NAME/$GIT_REPO_NAME HEAD:main"
22                     }
23                 }
24             }
25         }
26     }
27 }

```

Step5: Install ArgoCD on AWS EKS and configure it

- Before installation add the eks cluster to your CLI

```
</body></html>
ubuntu@ip-172-31-40-200:~$ aws eks update-kubeconfig --name EKS_CLOUD --region ap-south-1
Added new context arn:aws:eks:ap-south-1:862547479026:cluster/EKS_CLOUD to /home/ubuntu/.kube/
config
ubuntu@ip-172-31-40-200:~$ kubectl get nodes
NAME                                STATUS    ROLES    AGE    VERSION
ip-172-31-14-253.ap-south-1.compute.internal Ready    <none>   5h52m  v1.28.5-eks-5e0fdde
ubuntu@ip-172-31-40-200:~$ kubectl get namespaces
NAME              STATUS    AGE
default           Active    5h57m
kube-node-lease   Active    5h57m
kube-public        Active    5h57m
kube-system        Active    5h57m
ubuntu@ip-172-31-40-200:~$
```

- Install the ArgoCD

```
ubuntu@ip-172-31-40-200:~$ kubectl create namespace argocd
namespace/argocd created
ubuntu@ip-172-31-40-200:~$ kubectl get namespaces
NAME              STATUS    AGE
argocd            Active    2s
default           Active    5h58m
kube-node-lease   Active    5h58m
kube-public        Active    5h58m
kube-system        Active    5h58m
ubuntu@ip-172-31-40-200:~$ kubectl create namespace argocd
Error from server (AlreadyExists): namespaces "argocd" already exists
ubuntu@ip-172-31-40-200:~$ kubectl apply -n argocd -f https://raw.githubusercontent.com/argoproj/argo-cd/v2.4.7/manifests/install.yaml
customresourcedefinition.apiextensions.k8s.io/applications.argoproj.io created
customresourcedefinition.apiextensions.k8s.io/applicationsets.argoproj.io created
customresourcedefinition.apiextensions.k8s.io/appprojects.argoproj.io created
serviceaccount/argocd-application-controller created
serviceaccount/argocd-applicationset-controller created
serviceaccount/argocd-dex-server created
serviceaccount/argocd-notifications-controller created
serviceaccount/argocd-redis created
serviceaccount/argocd-repo-server created
serviceaccount/argocd-server created
```

```
ubuntu@ip-172-31-40-200:~$ kubectl get all -n argocd
NAME                                READY    STATUS    RESTARTS   AGE
pod/argocd-application-controller-0  1/1      Running   0           42s
pod/argocd-applicationset-controller-749957bcc9-7f4tr  1/1      Running   0           42s
pod/argocd-dex-server-b6c8d496b-jmdj6  1/1      Running   0           42s
pod/argocd-notifications-controller-5fff689764-mf6x7  1/1      Running   0           42s
pod/argocd-redis-55c6985b84-q45k5  1/1      Running   0           42s
pod/argocd-repo-server-d7666859-cbw9r  1/1      Running   0           42s
pod/argocd-server-6984446998-8zlmq  1/1      Running   0           42s

NAME                                TYPE          CLUSTER-IP    EXTERNAL-IP    PORT(S)          AGE
service/argocd-applicationset-controller  ClusterIP    10.100.106.144 <none>         7000/TCP,8080/TCP 42s
service/argocd-dex-server                 ClusterIP    10.100.1.181   <none>         5556/TCP,5557/TCP,5558/TCP 42s
service/argocd-metrics                    ClusterIP    10.100.5.23    <none>         8082/TCP          42s
service/argocd-notifications-controller-metrics ClusterIP    10.100.201.70  <none>         9001/TCP          42s
service/argocd-redis                      ClusterIP    10.100.97.8    <none>         6379/TCP          42s
service/argocd-repo-server                 ClusterIP    10.100.207.69  <none>         8081/TCP,8084/TCP 42s
service/argocd-server                     ClusterIP    10.100.141.29  <none>         80/TCP,443/TCP   42s
service/argocd-server-metrics              ClusterIP    10.100.223.40  <none>         8083/TCP          42s

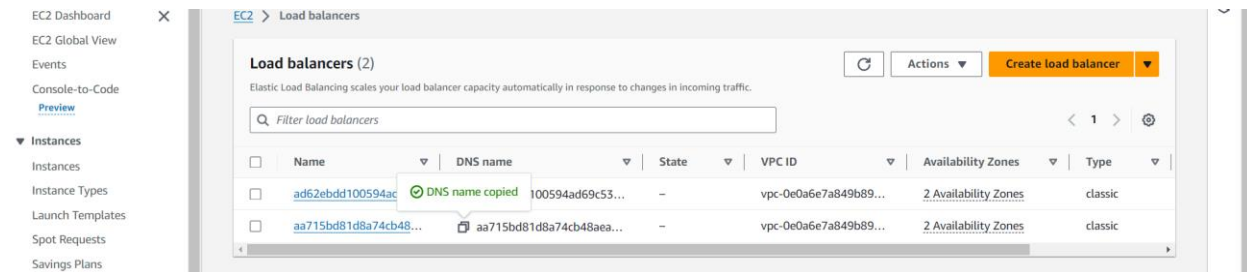
NAME                                READY    UP-TO-DATE    AVAILABLE    AGE
deployment.apps/argocd-applicationset-controller  1/1      1              1            42s
deployment.apps/argocd-dex-server                 1/1      1              1            42s
deployment.apps/argocd-notifications-controller  1/1      1              1            42s
deployment.apps/argocd-redis                      1/1      1              1            42s
deployment.apps/argocd-repo-server                 1/1      1              1            42s
deployment.apps/argocd-server                     1/1      1              1            42s

NAME                                DESIRED    CURRENT    READY    AGE
replicaset.apps/argocd-applicationset-controller-749957bcc9  1          1          1        42s
replicaset.apps/argocd-dex-server-b6c8d496b  1          1          1        42s
replicaset.apps/argocd-notifications-controller-5fff689764  1          1          1        42s
replicaset.apps/argocd-redis-55c6985b84  1          1          1        42s
replicaset.apps/argocd-repo-server-d7666859  1          1          1        42s
replicaset.apps/argocd-server-6984446998  1          1          1        42s

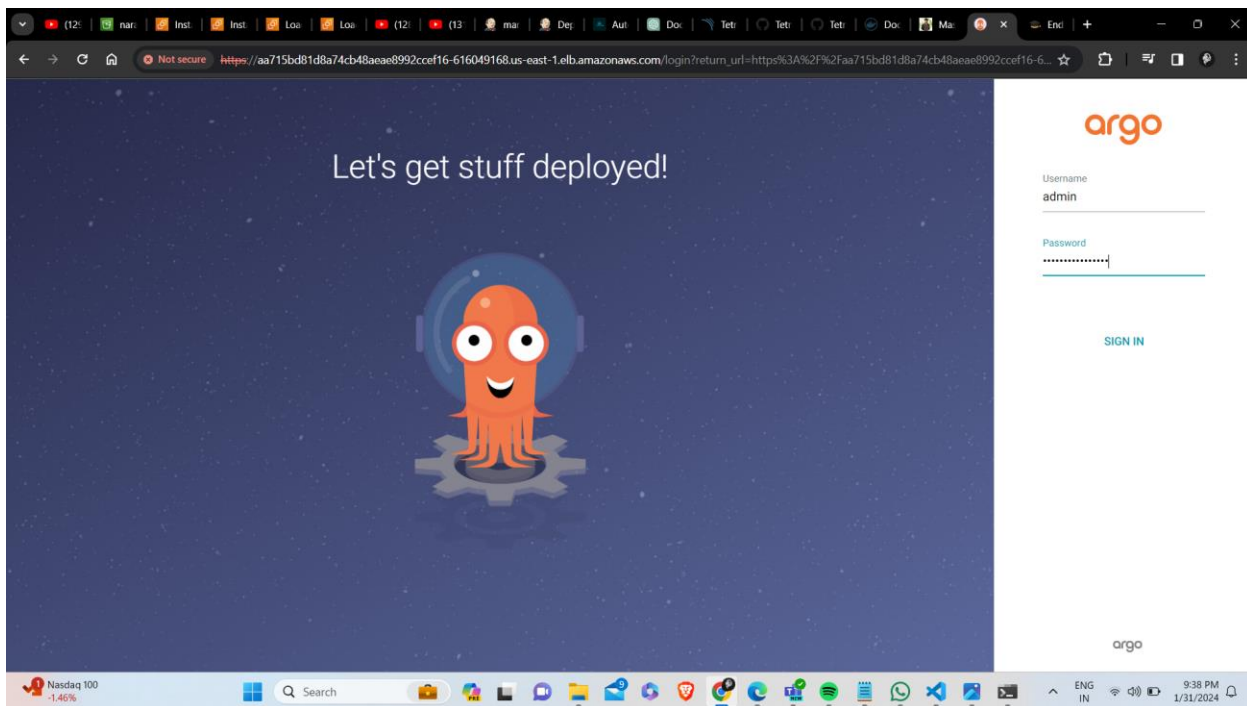
NAME                                READY    AGE
statefulset.apps/argocd-application-controller  1/1      42s
```

- Now modify the argocd service to loadbalancer

```
ubuntu@ip-172-31-40-200:~$ kubectl patch svc argocd-server -n argocd -p '{"spec": {"type": "LoadBalancer"}}'
service/argocd-server patched
ubuntu@ip-172-31-40-200:~$
ubuntu@ip-172-31-40-200:~$
```



- Now copy the dns of the loadbalancer and paste in the web



- Take the password from the secrets and decode it into bas64

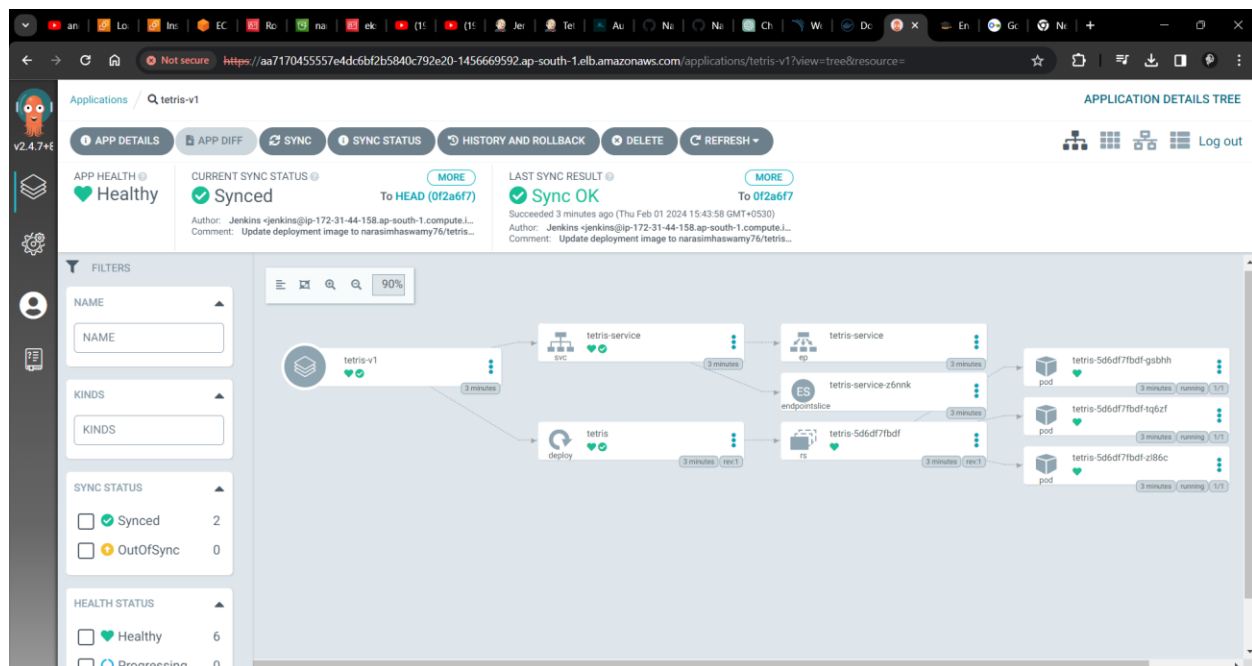
```
Command Prompt  X  ubuntu@ip-172-31-40-200: ~  X  +  v
ubuntu@ip-172-31-40-200:~$ kubectl get secrets -n argocd
NAME                                TYPE      DATA      AGE
argocd-initial-admin-secret         Opaque    1           17m
argocd-notifications-secret         Opaque    0           17m
argocd-secret                       Opaque    5           17m
ubuntu@ip-172-31-40-200:~$ kubectl edit secret argocd-initial-admin-secret -n argocd
```

```
Command Prompt  X  ubuntu@ip-172-31-40-200: ~  X  +  v
## Please edit the object below. Lines beginning with a '#' will be ignored,
# and an empty file will abort the edit. If an error occurs while saving this file will be
# reopened with the relevant failures.
#
apiVersion: v1
data:
  password: VEhDQk11ampoT0pqT1IxWA==
kind: Secret
metadata:
  creationTimestamp: "2024-01-31T15:48:55Z"
  name: argocd-initial-admin-secret
  namespace: argocd
  resourceVersion: "52179"
  uid: feb5149d-3255-4a1b-9058-2d77802f879c
type: Opaque
~
~
~
~
```

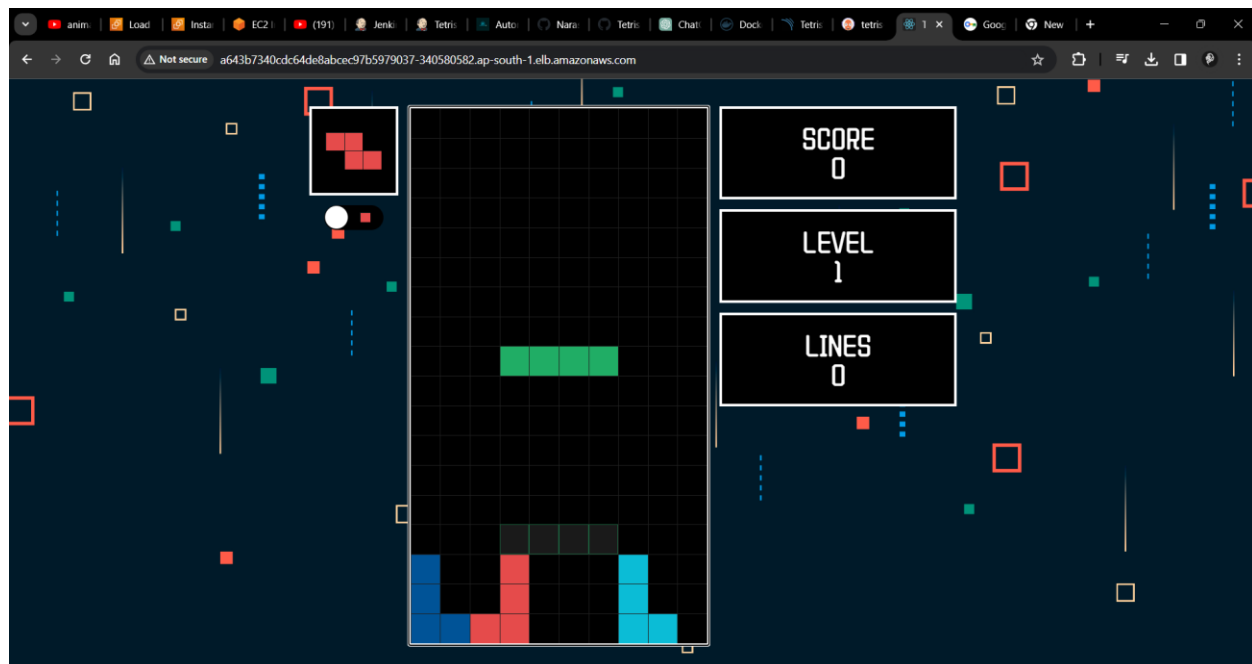
```
Command Prompt  X  ubuntu@ip-172-31-40-200: ~  X  +  v
ubuntu@ip-172-31-40-200:~$ kubectl get secrets -n argocd
NAME                                TYPE      DATA      AGE
argocd-initial-admin-secret         Opaque    1           17m
argocd-notifications-secret         Opaque    0           17m
argocd-secret                       Opaque    5           17m
ubuntu@ip-172-31-40-200:~$ kubectl edit secret argocd-initial-admin-secret -n argocd
Edit cancelled, no changes made.
ubuntu@ip-172-31-40-200:~$ echo VEhDQk11ampoT0pqT1IxWA== | base64 --decode
THCBMujjhOJjORIXubuntu@ip-172-31-40-200:~$
```

Step6: Use ArgoCD to synchronize AWS EKS deployments with manifest changes in git repo

- Create a project in the argocd with the connection of git-hub manifest repo and define the destination eks cluster in the aws eks.



- It will monitors the manifest repo when ever the change occur in the repo it will keep the eks cluster in the desired state
- After deploying the pods and load balancer service in the eks cluster then we can access the game through the service endpoint



Step7: Install Prometheus, Grafana, and Node Exporter using Helm on AWS EKS.

- First install the helm in the eks cluster.
- Install the grafana and node exporter and grafana using helm commands in the eks cluster.
- After installation edit the Prometheus file to add the targets of node exporter

```
Command Prompt x ubuntu@ip-172-31-35-16:/etc x ubuntu@ip-172-31-44-158:~ x + x x
scrape_interval: 15s # Set the scrape interval to every 15 seconds. Default is every 1 minute.
evaluation_interval: 15s # Evaluate rules every 15 seconds. The default is every 1 minute.
# scrape_timeout is set to the global default (10s).

# Alertmanager configuration
alerting:
  alertmanagers:
    - static_configs:
        - targets:
            # - alertmanager:9093

# Load rules once and periodically evaluate them according to the global 'evaluation_interval'.
rule_files:
  # - "first_rules.yml"
  # - "second_rules.yml"

# A scrape configuration containing exactly one endpoint to scrape:
# Here it's Prometheus itself.
scrape_configs:
  # The job name is added as a label `job=<job_name>` to any timeseries scraped from this config.
  - job_name: "prometheus"

    # metrics_path defaults to '/metrics'
    # scheme defaults to 'http'.

    static_configs:
      - targets: ["localhost:9090"]
  - job_name: 'node_exporter'
    static_configs:
      - targets: ['localhost:9100']
  - job_name: 'narasimha'
    metrics_path: '/metrics'
    static_configs:
      - targets: ['65.0.76.193:9100']
```

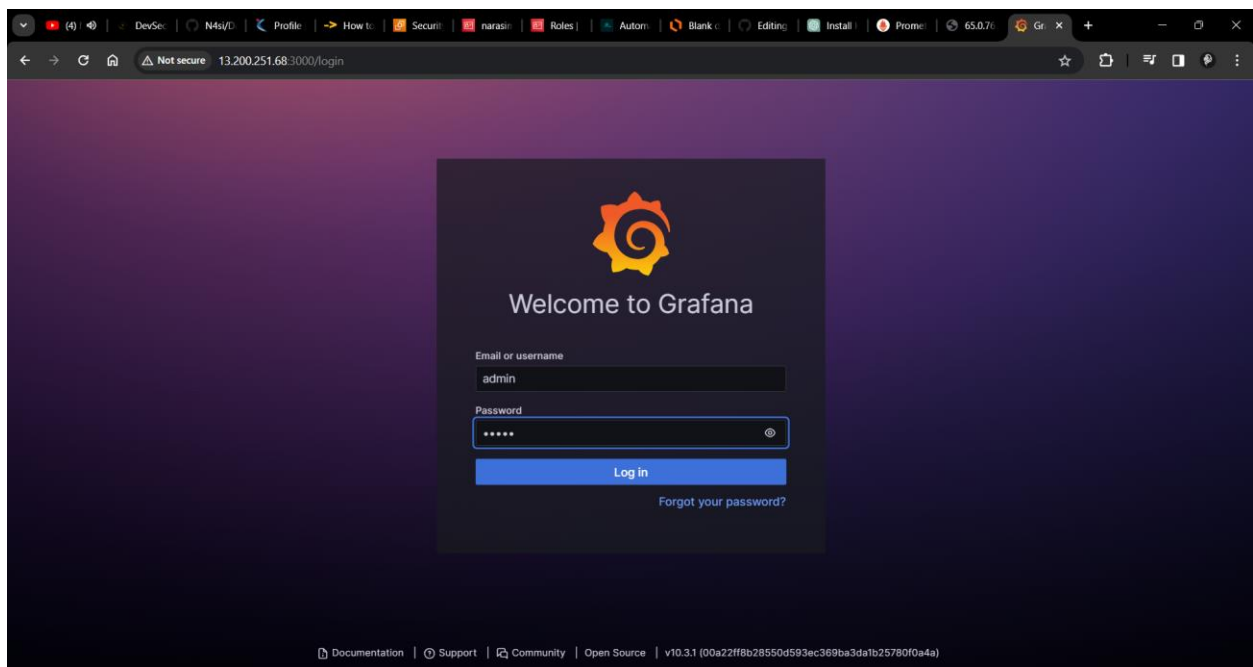
The screenshot shows the Prometheus web interface. At the top, there's a navigation bar with 'Prometheus', 'Alerts', 'Graph', 'Status', and 'Help'. Below this is a search bar and filter buttons for 'All', 'Unhealthy', and 'Collapse All'. The main content area lists three target groups:

- narasimha (1/1 up)**: Shows a table with one target at `http://65.0.76.193:9100/metrics` with state 'UP', labels `instance="65.0.76.193:9100"` and `job="narasimha"`, last scrape at 3.421s ago, and duration 32.527ms.
- node_exporter (1/1 up)**: Shows a table with one target at `http://localhost:9100/metrics` with state 'UP', labels `instance="localhost:9100"` and `job="node_exporter"`, last scrape at 8.947s ago, and duration 13.694ms.
- prometheus (1/1 up)**: Shows a table with one target at `http://localhost:9090/metrics` with state 'UP', labels `instance="localhost:9090"` and `job="prometheus"`, last scrape at 6.566s ago, and duration 4.768ms.

- Node Exporter metrics :

```
# HELP go_gc_duration_seconds A summary of the pause duration of garbage collection cycles.
# TYPE go_gc_duration_seconds summary
go_gc_duration_seconds{quantile="0"} 2.1445e-05
go_gc_duration_seconds{quantile="0.25"} 3.6411e-05
go_gc_duration_seconds{quantile="0.5"} 4.0065e-05
go_gc_duration_seconds{quantile="0.75"} 4.4421e-05
go_gc_duration_seconds{quantile="1"} 0.000115871
go_gc_duration_seconds_sum 0.003473487
go_gc_duration_seconds_count 81
# HELP go_goroutines Number of goroutines that currently exist.
# TYPE go_goroutines gauge
go_goroutines 9
# HELP go_info Information about the Go environment.
# TYPE go_info gauge
go_info{version="go1.21.4"} 1
# HELP go_memstats_alloc_bytes Number of bytes allocated and still in use.
# TYPE go_memstats_alloc_bytes gauge
go_memstats_alloc_bytes 2.454152e+06
# HELP go_memstats_alloc_bytes_total Total number of bytes allocated, even if freed.
# TYPE go_memstats_alloc_bytes_total counter
go_memstats_alloc_bytes_total 7.079792e+07
# HELP go_memstats_buck_hash_sys_bytes Number of bytes used by the profiling bucket hash table.
# TYPE go_memstats_buck_hash_sys_bytes gauge
go_memstats_buck_hash_sys_bytes 1.463999e+06
# HELP go_memstats_frees_total Total number of frees.
# TYPE go_memstats_frees_total counter
go_memstats_frees_total 514547
# HELP go_memstats_gc_sys_bytes Number of bytes used for garbage collection system metadata.
# TYPE go_memstats_gc_sys_bytes gauge
go_memstats_gc_sys_bytes 4.277808e+06
# HELP go_memstats_heap_alloc_bytes Number of heap bytes allocated and still in use.
# TYPE go_memstats_heap_alloc_bytes gauge
go_memstats_heap_alloc_bytes 2.454152e+06
# HELP go_memstats_heap_idle_bytes Number of heap bytes waiting to be used.
# TYPE go_memstats_heap_idle_bytes gauge
go_memstats_heap_idle_bytes 4.071424e+06
# HELP go_memstats_heap_inuse_bytes Number of heap bytes that are in use.
# TYPE go_memstats_heap_inuse_bytes gauge
go_memstats_heap_inuse_bytes 3.8912e+06
# HELP go_memstats_heap_objects Number of allocated objects.
# TYPE go_memstats_heap_objects gauge
go_memstats_heap_objects 20778
# HELP go_memstats_heap_released_bytes Number of heap bytes released to OS.
# TYPE go_memstats_heap_released_bytes gauge
go_memstats_heap_released_bytes 2.760704e+06
# HELP go_memstats_heap_sys_bytes Number of heap bytes obtained from system.
# TYPE go_memstats_heap_sys_bytes gauge
```

Grafana :



- Login with grafana and add the the datasource as the Prometheus and create a dashboard with that Prometheus datasource
- Then the dashboard of the eks cluster will be displayed

