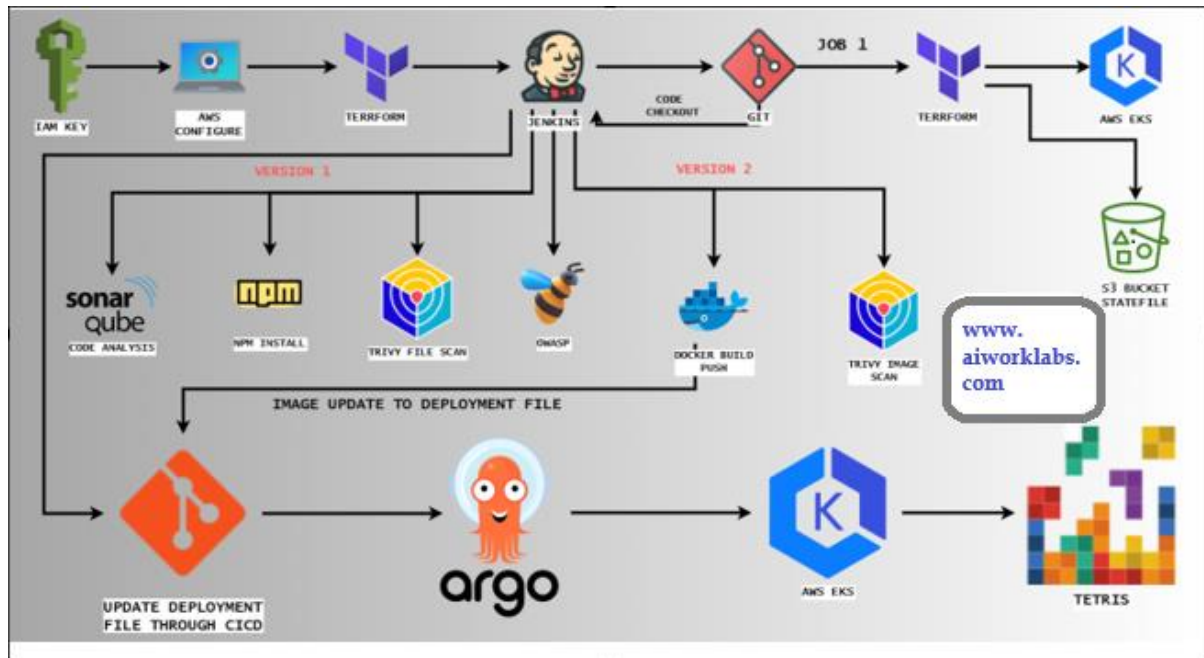


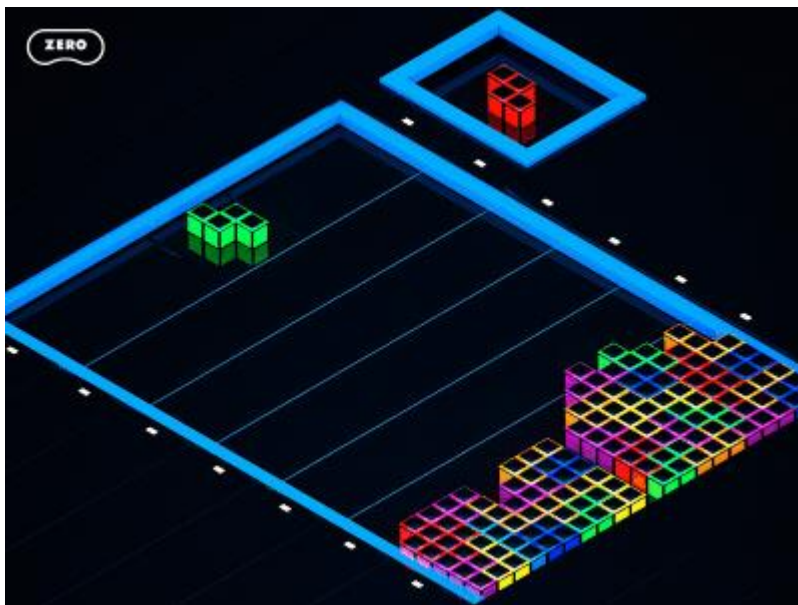
Rajinikanth DevSecOps Projects



Automating Tetris Deployments: DevSecOps with ArgoCD, Terraform, and Jenkins for Two Game Versions

Welcome to the world of DevSecOps automation, where cutting-edge

technologies merge to streamline the deployment of one of the most beloved classic games, Tetris. In this blog, we'll embark on a fascinating journey that unveils the power of ArgoCD, Terraform, and Jenkins in orchestrating a seamless and secure deployment pipeline for not one, but two distinct versions of the Tetris game. Imagine a scenario where your Tetris application effortlessly transitions through various stages of development, from infrastructure provisioning to continuous integration and delivery, all while maintaining the highest standards of security. Join us as we delve into the details of how these sophisticated tools work together to create an automated, efficient, and foolproof DevOps workflow, capable of handling multiple game versions, designed to impress even the most discerning tech enthusiasts.



GitHub REPOSITORIES

TETRIS-VERSION1

<https://github.com/RajinikanthVadla/Tetris-V1.git>

Prerequisites:

1. **AWS Account:** To get started, you'll need an active AWS account. Ensure that you have access and permission to create and manage AWS resources.
2. **AWS CLI:** Install the AWS Command Line Interface (CLI) on your local machine and configure it with your AWS credentials. This is essential for managing your AWS resources.
3. **IAM User and Key Pair:** Create an IAM (Identity and Access Management) user with the necessary permissions to provision resources on AWS. Additionally, generate an IAM Access Key and Secret Access Key for programmatic access. Ensure that you securely manage these credentials.
4. **S3 Bucket:** Set up an S3 bucket to store your Terraform state files. This bucket is crucial for maintaining the state of your infrastructure and enabling collaboration.
5. **Terraform:** Install Terraform on your local machine. Terraform is used for provisioning infrastructure as code and managing AWS resources. Make sure to configure Terraform to work with your AWS credentials and your S3 bucket for state

Step1: How to install and setup Terraform on Windows

Download Terraform:

Visit the official Terraform website: terraform.io/downloads.html

Verify the Installation:

Open a new Command Prompt or PowerShell window.

Type `terraform -version` and press Enter. This command should display the Terraform version, confirming that Terraform is installed and in your PATH.

```
PS C:\Users\Rajinikanth\Downloads> terraform -version
Terraform v1.7.0
on windows_386

Your version of Terraform is out of date! The latest version
is 1.7.1. You can update by downloading from https://www.terraform.io/downloads.html
PS C:\Users\Rajinikanth\Downloads> |
```

Step2: Download the AWS CLI Installer:

Visit the AWS CLI Downloads page: aws.amazon.com/cli

Under “Install the AWS CLI,” click on the “64-bit” link to download the AWS CLI installer for Windows

Verify the Installation:

Open a Command Prompt or PowerShell window.

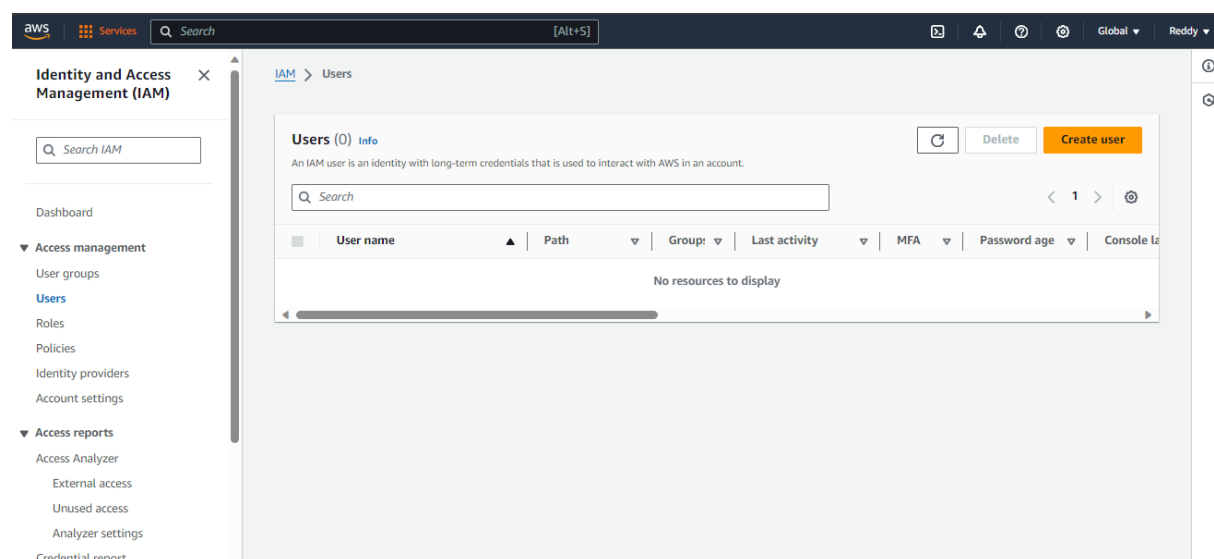
Type **aws --version** and press Enter. This command should display the AWS CLI version, confirming that the installation was successful.

```
PS C:\Users\Rajinikanth\Downloads> aws --version
aws-cli/2.9.0 Python/3.9.11 windows/10 exe/AMD64 prompt/off
PS C:\Users\Rajinikanth\Downloads> |
```

Step3: create an IAM user

Navigate to the **AWS console**

Click the “Search” field.



Create user Rajinikanth_Vadla and click next

Step 2

Set permissions

Step 3

Review and create

Step 4

Retrieve password

User details

User name

Rajinikanth_Vadla

The user name can have up to 64 characters. Valid characters: A-Z, a-z, 0-9, and +, -, @, _ (hyphen)

☒ Provide user access to the AWS Management Console - optional

If you're providing console access to a person, it's a best practice to manage their access in IAM Identity Center.

Are you providing console access to a person?

User type

☒ Specify a user in Identity Center - Recommended

We recommend that you use Identity Center to provide console access to a person. With Identity Center, you can centrally manage user access to their AWS accounts and cloud applications.

☐ I want to create an IAM user

We recommend that you create IAM users only if you need to enable programmatic access through access keys, service-specific credentials for AWS CodeCommit or Amazon Keyspaces, or a backup credential for emergency account access.

☐ If you are creating programmatic access through access keys or service-specific credentials for AWS CodeCommit or Amazon Keyspaces, you can generate them after you create this IAM user. [Learn more](#)

Cancel

Next

Then attach a policy as per below image

Services

Search

[Alt+5]

Global

IAM > Users > Create user

Step 1

Specify user details

Step 2

Set permissions

Step 3

Review and create

Step 4

Retrieve password

Set permissions

Add user to an existing group or create a new one. Using groups is a best-practice way to manage user's permissions by job functions. [Learn more](#)

Permissions options

☐ Add user to group

Add user to an existing group, or create a new group. We recommend using groups to manage user permissions by job function.

☐ Copy permissions

Copy all group memberships, attached managed policies, and inline policies from an existing user.

☒ Attach policies directly

Attach a managed policy directly to a user. As a best practice, we recommend attaching policies to a group instead. Then, add the user to the appropriate group.

Permissions policies (1/1174)

Choose one or more policies to attach to your new user.

Search

Filter by Type

All types

<

1

2

3

4

5

6

7

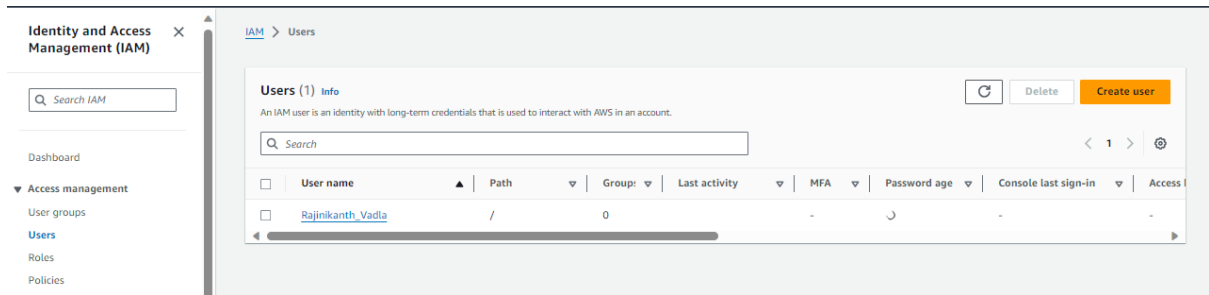
...

59

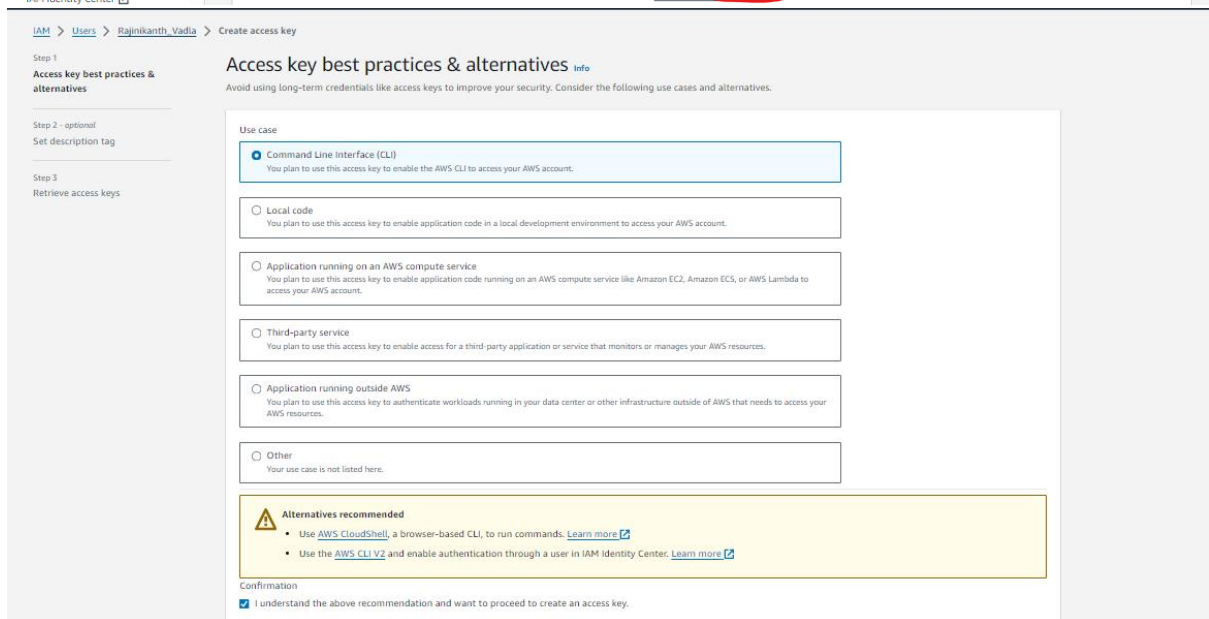
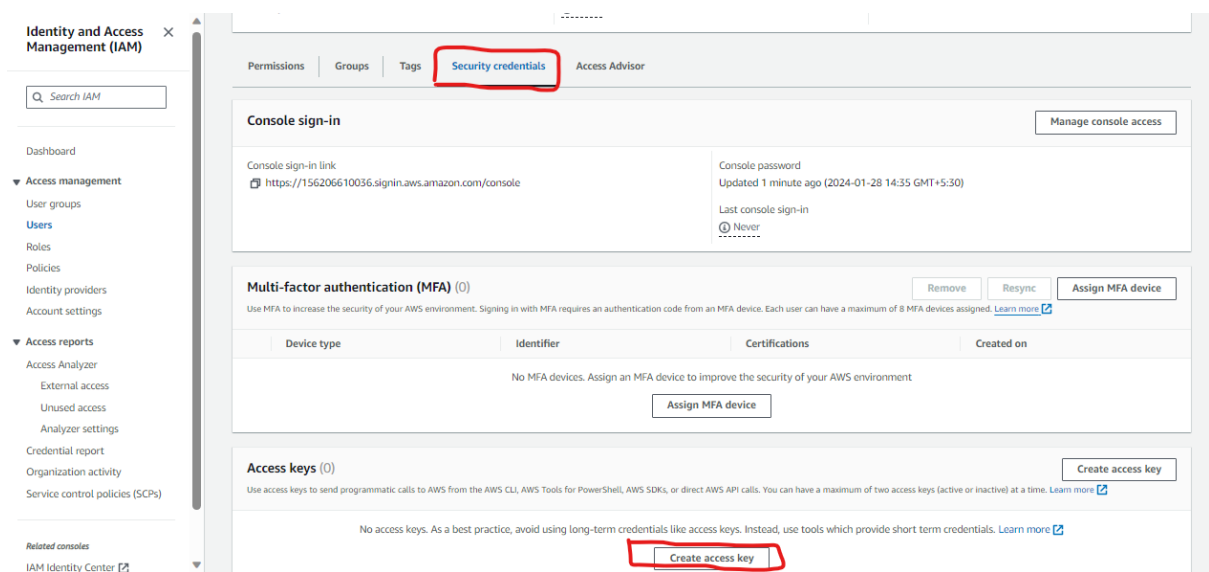
>

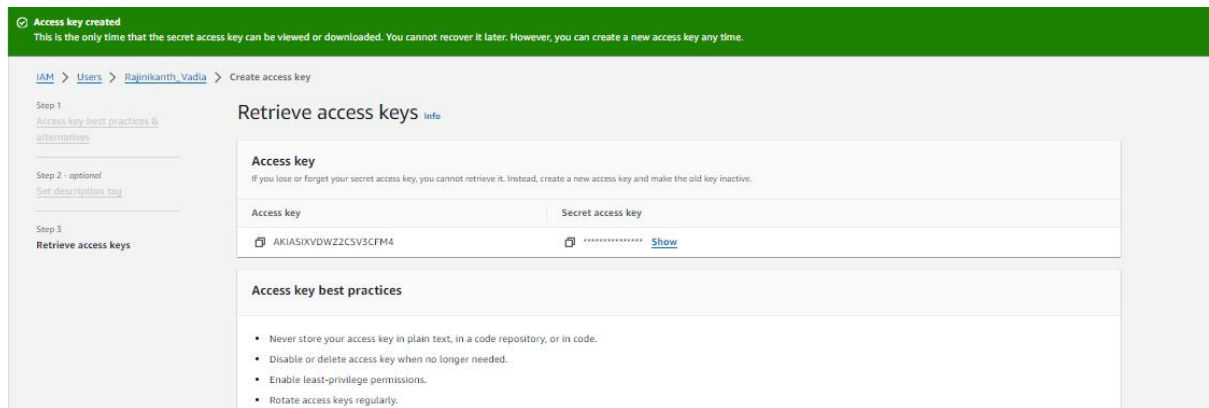
	Policy name	Type	Attached entities
<input type="checkbox"/>	AccessAnalyzerServiceRolePolicy	AWS managed	0
<input checked="" type="checkbox"/>	AdministratorAccess	AWS managed - job function	0
<input type="checkbox"/>	AdministratorAccess-Amplify	AWS managed	0
<input type="checkbox"/>	AdministratorAccess-AWSElastic...	AWS managed	0

Once user created click on user

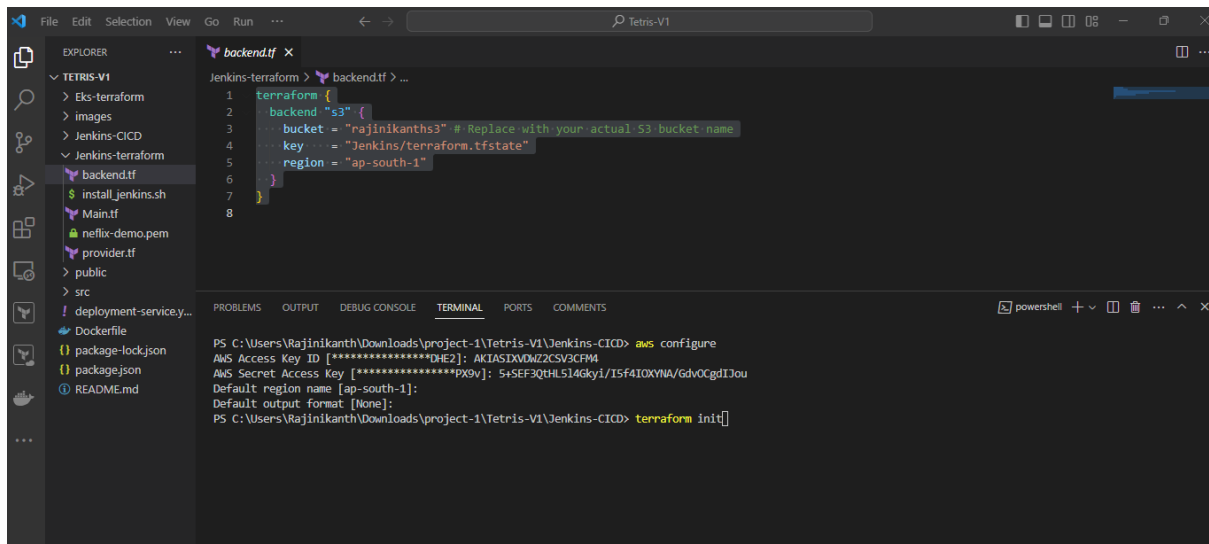


Create an access key for this user

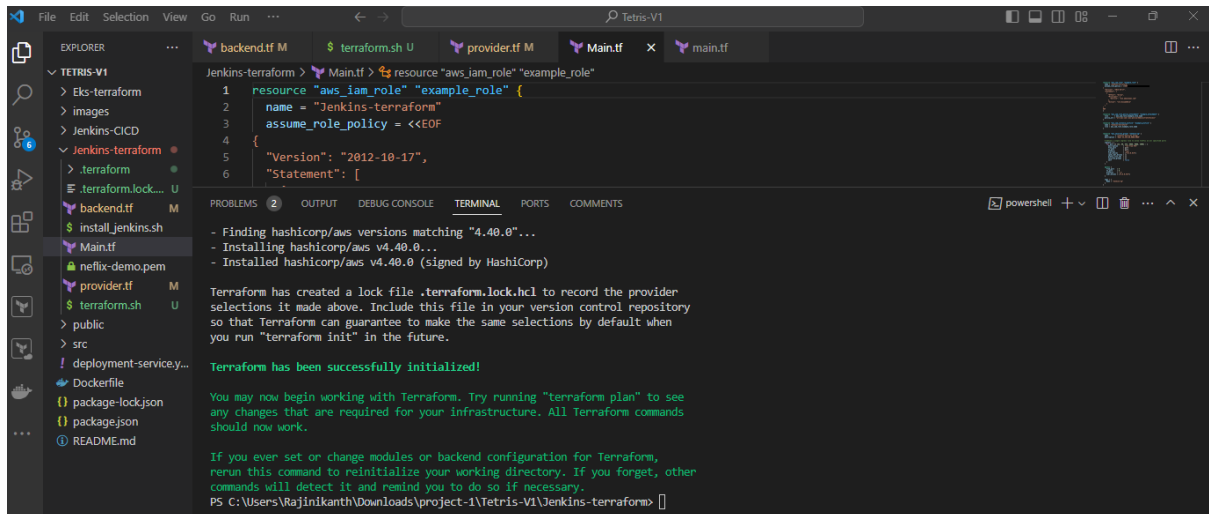




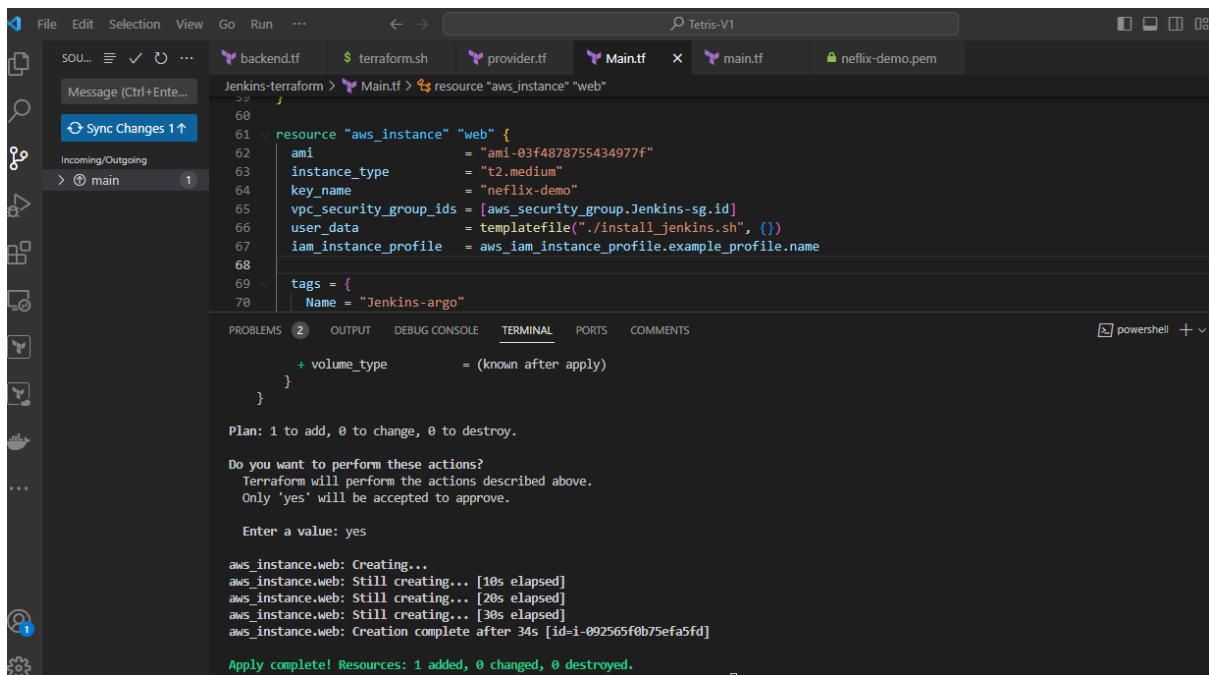
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

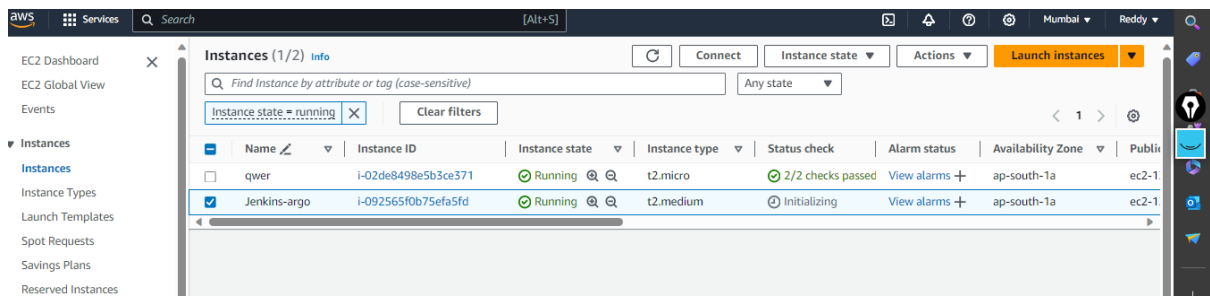


Clone repo and go to terraform code folder open terminal before that you have to create S3 bucket in AWS



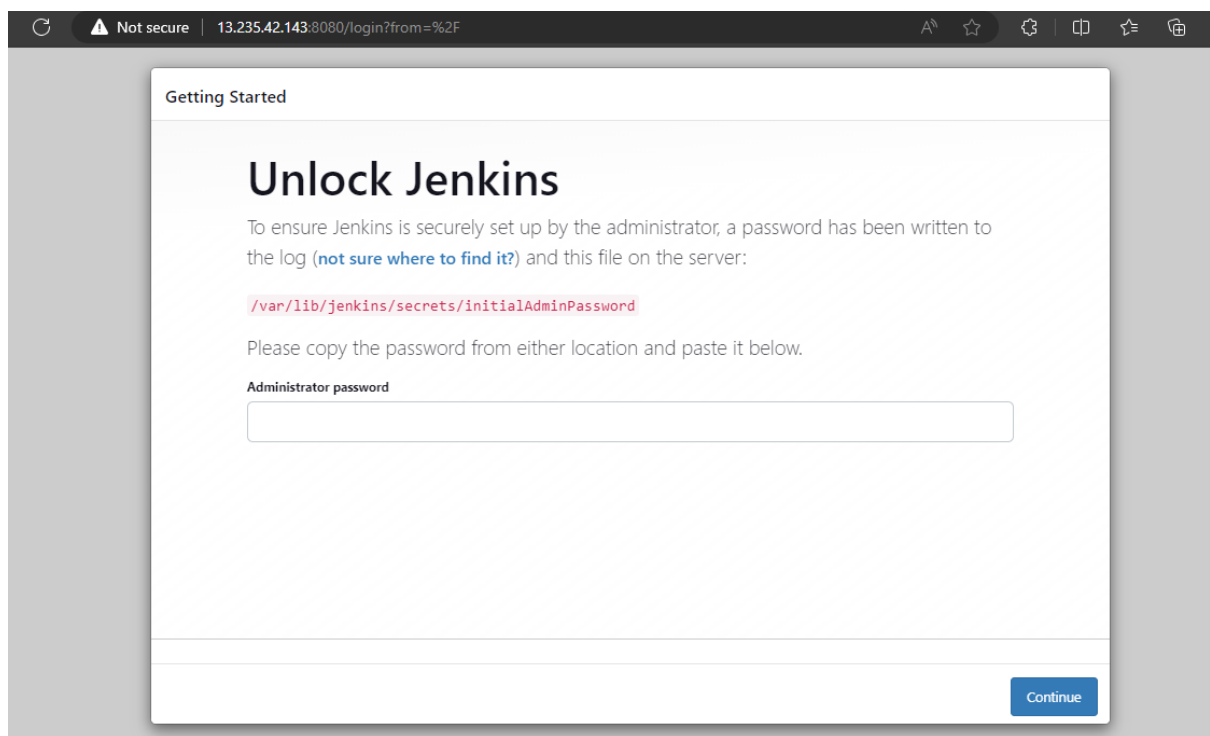
Then terraform apply it create a server in AWS console





connect to Jenkins from this public IP

Connect to instance and get the Jenkins password



Not secure | 13.235.42.143:8080

Getting Started

Create First Admin User

Username

Password

Confirm password

Full name

Jenkins 2.426.3

[Skip and continue as admin](#) [Save and Continue](#)

Open SonarQube same IP but different port number

← → ↻ Not secure | 13.235.42.143:9000/sessions/new?return_to=%2F

Log in to SonarQube

admin

[Log in](#) [Cancel](#)

Then update your password

The check the versions

```
root@ip-172-31-44-207:/home/ubuntu# trivy --version
aws --version
terraform --version
kubect1 version
Version: 0.48.3
aws-cli/2.15.15 Python/3.11.6 Linux/6.2.0-1017-aws exe/x86_64.ubuntu.22 prompt/off
Terraform v1.7.1
on linux_amd64
Client Version: v1.29.1
Kustomize Version: v5.0.4-0.20230601165947-6ce0bf390ce3
Error from server (Forbidden): <html><head><meta http-equiv='refresh' content='1;url=/login?from=%2Fversion%3Ftimeout%3D32s' /><script>>window.location.replace('/login?from=%2Fversion%3Ftimeout%3D32s');</script></head><body style='background-color:white; color:white;'>

Authentication required
<!--
-->

</body></html>
root@ip-172-31-44-207:/home/ubuntu#
```

That is done now go to Jenkins and add a terraform plugin to provision the AWS EKS using the Pipeline Job.

Go to Jenkins dashboard -> Manage Jenkins -> Plugins

Available Plugins, Search for Terraform and install it.



after installation

Now come back to Manage Jenkins -> Tools

Add the terraform in Tools

```
root@ip-172-31-44-207:/home/ubuntu# which terraform
```

```
/usr/bin/terraform
```

```
root@ip-172-31-44-207:/home/ubuntu#
```

Terraform installations

Add Terraform

≡ Terraform

Name

Install directory

☐ Install automatically ?

Add Terraform

Save Apply


Apply and save.

CHANGE YOUR S3 BUCKET NAME IN THE [BACKEND.TF](https://backend.tf)


Now create a new job for the Eks provision

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.

**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.

I want to do this with build parameters to apply and destroy while building only.

you have to add this inside job like the below image

✓ This project is parameterized ?

≡ Choice Parameter ?

Name ?

action

Choices ?

apply
destroy

Description ?

Plain text [Preview](#)

Save Apply

Let's add a pipeline

add script

```
pipeline{
  agent any
  stages {
    stage('Checkout from Git'){
      steps{
        git branch: 'main', url:
'https://github.com/RajinikanthVadla/Tetris-V1.git'
      }
    }
  }
}
```

```
stage('Terraform version'){
    steps{
        sh 'terraform --version'
    }
}

stage('Terraform init'){
    steps{
        dir('Eks-terraform') {
            sh 'terraform init'
        }
    }
}

stage('Terraform validate'){
    steps{
        dir('Eks-terraform') {
            sh 'terraform validate'
        }
    }
}

stage('Terraform plan'){
    steps{
        dir('Eks-terraform') {
```

```

        sh 'terraform plan'

    }

}

stage('Terraform apply/destroy'){

    steps{

        dir('Eks-terraform') {

            sh 'terraform ${action} --auto-approve'

        }

    }

}

}

```

← → ↻ ⚠ Not secure 13.235.42.143:8080/job/terraform-Eks/ 🔍 ⌵ 🔄 📄 ⭐ 🗂 🔄 📄

Stage Logs (Terraform init) ✕

🔍 Shell Script -- terraform init (self time 1s)

```

+ terraform init

B[0mB[1mInitializing the backend...B[0m
B[31mB[31mB[0mB[0m
B[31mB[0m B[0mB[1mB[31mError: B[0mB[0mB[1mFailed to get existing workspaces: Unable to list objects in S3 bucket "ratedatastore": operation error S3: ListObjectsV2, https response error StatusCo
de: 403, RequestID: Z2HFD3XD3A4NWCY, HostID: nhZwBCR3d1/cFDj210hU2nPgA71a9TcegeVCXyEL+1/BwzozJxfG095xdJK013jkmUckjyT9qRg=, api error AccessDenied: Access DeniedB[0m
B[31mB[0m B[0m
B[31mB[0m B[0m B[0mB[0m
B[31m B[0mB[0m
B[0mB[0m

```

Build History trend ▾

Filter builds... /

#2

Jan 28, 2024, 12:19 PM

#1

Jan 28, 2024, 12:17 PM

#2

Jan 28 17:49

1 commit

#1

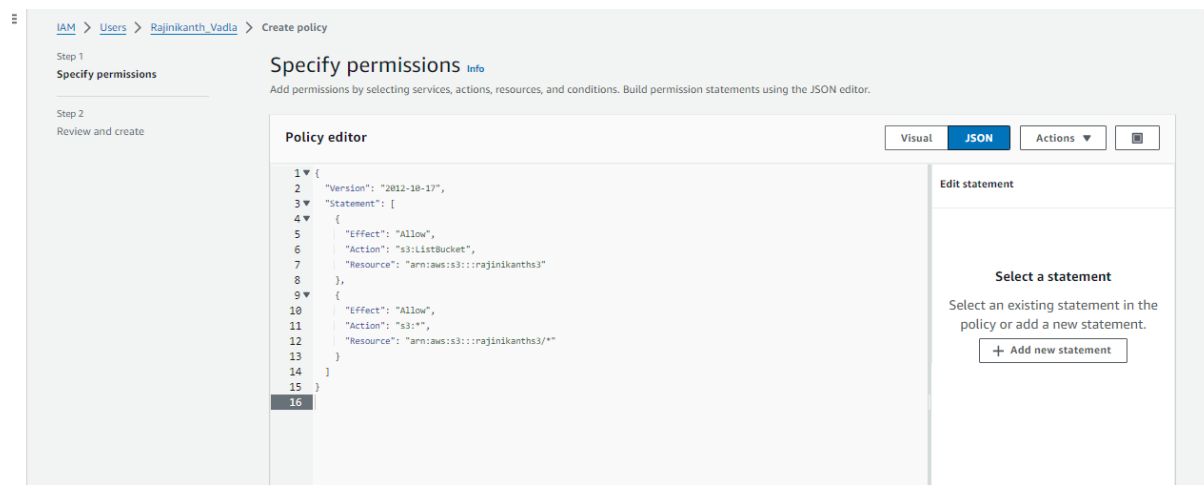
Jan 28 17:47

No Changes

929ms	345ms	1s failed	50ms failed	48ms failed	49ms failed
5s	947ms	1s failed	107ms failed	68ms failed	56ms failed

Permalinks

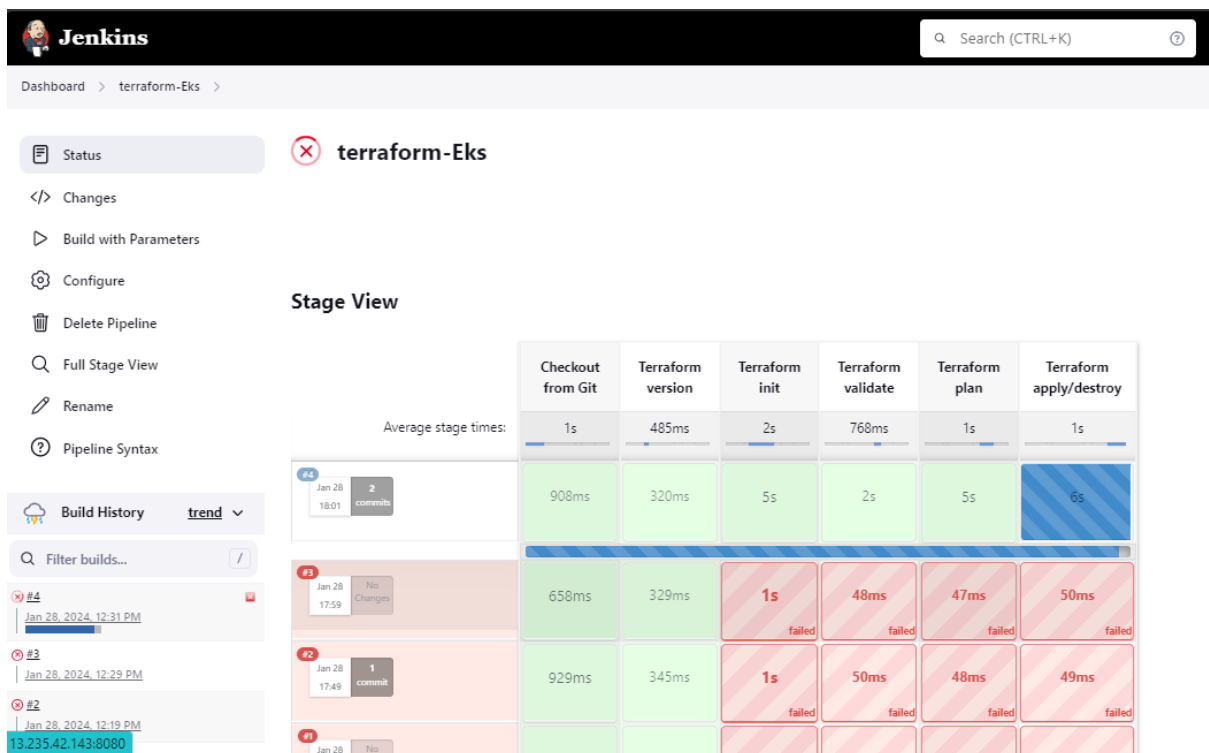
if its get failed ad in line policy in IAM user



```
{
  "Version": "2012-10-17",
  "Statement": [
    {
      "Effect": "Allow",
      "Action": "s3:ListBucket",
      "Resource": "arn:aws:s3:::rajinikanths3"
    },
    {
      "Effect": "Allow",
      "Action": "s3:*",
      "Resource": "arn:aws:s3:::rajinikanths3/*"
    }
  ]
}
```

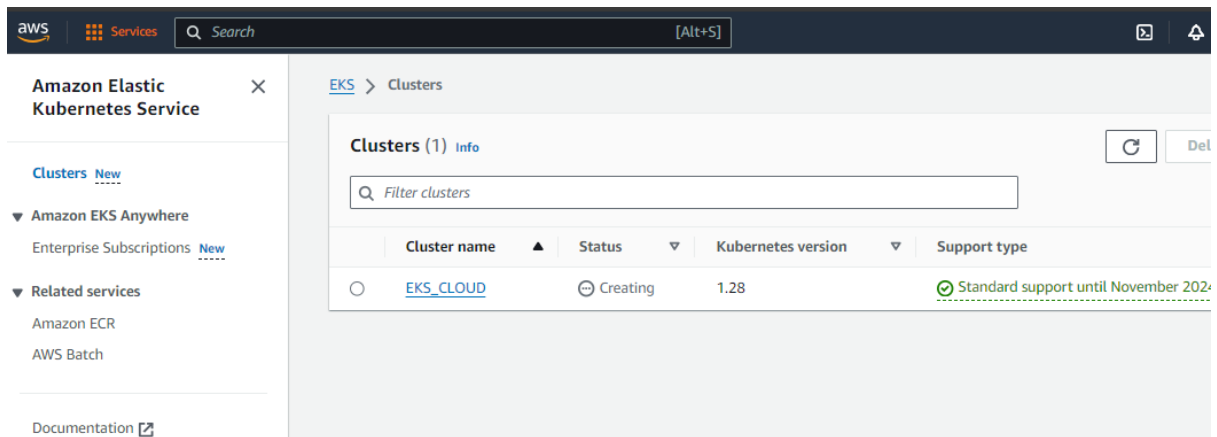


```
}  
  
]  
  
}
```



it will take 10-15 mins time to create EKS cluster

Then we need to do dome configs in Jenkins



after created cluster go to Jenkins

Now let's build Tetris version 1

We need some plugins to complete this process

Go to Jenkins dashboard

Manage Jenkins -> Plugins -> Available Plugins

Search for the Below Plugins

Eclipse Temurin installer

Sonarqube Scanner

NodeJs

Owasp Dependency-Check

Docker

Docker Commons

Docker Pipeline

Docker API

Docker-build-step

<input checked="" type="checkbox"/>	Eclipse Temurin installer 1.5 Provides an installer for the JDK tool that downloads the JDK from https://adoptium.net <div>This plugin is up for adoption! We are looking for new maintainers. Visit our Adopt a Plugin initiative for more information.</div>	1 yr 0 mo ago
<input checked="" type="checkbox"/>	SonarQube Scanner 2.16.1 External Site/Tool Integrations Build Reports This plugin allows an easy integration of SonarQube , the open source platform for Continuous Inspection of code quality.	15 days ago
<input checked="" type="checkbox"/>	NodeJS 1.6.1 npm NodeJS Plugin executes NodeJS script as a build step.	2 mo 10 days ago
<input checked="" type="checkbox"/>	OWASP Dependency-Check 5.4.3 Security DevOps Build Tools Build Reports This plug-in can independently execute a Dependency-Check analysis and visualize results. Dependency-Check is a utility that identifies project dependencies and checks if there are any known, publicly disclosed, vulnerabilities.	1 mo 16 days ago
<input checked="" type="checkbox"/>	Docker 1.5 Cloud Providers Cluster Management docker This plugin integrates Jenkins with Docker	1 mo 21 days ago

Configure in Global Tool Configuration

Goto Manage Jenkins → Tools → Install JDK(17) and NodeJs(16)→ Click on Apply and Save

JDK

Name

jdk7

☒ Install automatically ?

Install from adoptium.net ?

Version ?

jdk-17.0.8.1+1

Add Installer

Add JDK

Add NodeJS

NodeJS

Name

node16

☒ Install automatically ?

Install from nodejs.org

Version

NodeJS 16.2.0

For the underlying architecture, if available, force the installation of the 32bit package. Otherwise the build will fail

☐ Force 32bit architecture

Global npm packages to install

Specify list of packages to install globally -- see npm install -g. Note that you can fix the packages version by using the syntax 'packageName@version'

Global npm packages refresh hours

For Sonarqube use the latest version

Add SonarQube Scanner

SonarQube Scanner

Name

sonar-scanner

☒ Install automatically ?

Install from Maven Central

Version

SonarQube Scanner 5.0.1.3006

Add Installer ▾

Add SonarQube Scanner

For Owasp use the 6.5.1 version

Dependency-Check installations

Add Dependency-Check

Dependency-Check

Name

DP-Check

☒ Install automatically ?

Install from github.com

Version

dependency-check 6.5.1

Add Installer ▾

Use the latest version of Docker

Dashboard > Manage Jenkins > Tools

Add Docker

≡ Docker

Name

docker

☒ Install automatically ?

≡ Download from docker.com

Docker version ?

latest

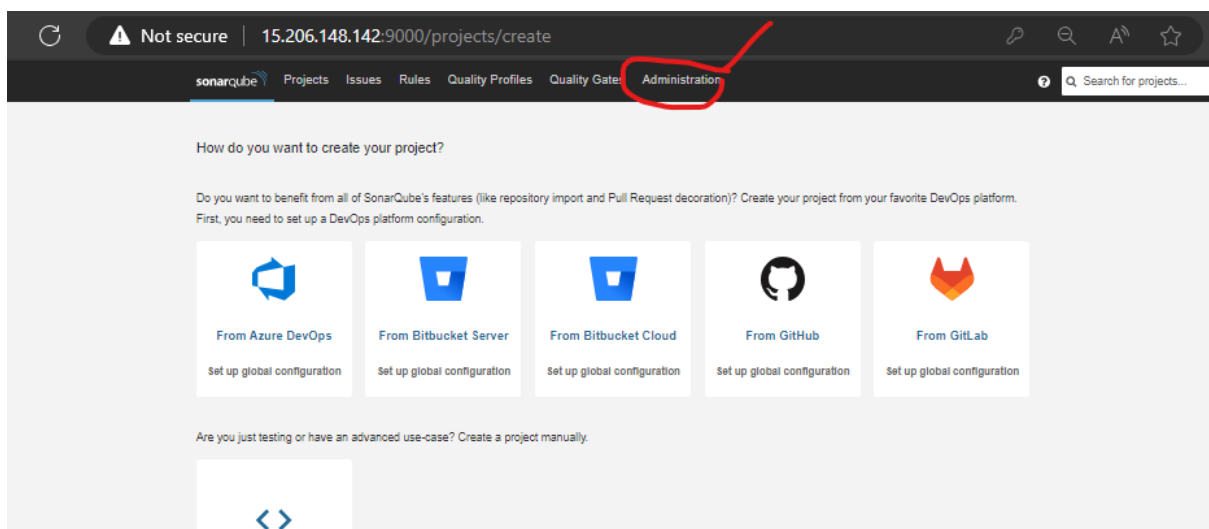
Add Installer ▾

Add Docker

Click apply and save.

Configure Sonar Server in Manage Jenkins

Grab the Public IP Address of your EC2 Instance, Sonarqube works on Port 9000, so <Public IP>:9000. Goto your Sonarqube Server. Click on Administration → Security → Users → Click on Tokens and Update Token → Give it a name → and click on Generate Token



The screenshot shows the SonarQube Administration interface. The top navigation bar includes 'sonarqube', 'Projects', 'Issues', 'Rules', 'Quality Profiles', 'Quality Gates', and 'Administration' (highlighted with a red box). Below the navigation bar, the 'Administration' section is active, with a dropdown menu showing 'Users' (highlighted with a red box), 'Groups', 'Global Permissions', and 'Permission Templates'. The 'Users' section is selected, showing a list of users: 'Administrator admin' and 'sonar-administrators'. The 'Tokens' tab is selected, showing a table of tokens for the 'Administrator admin' user. The table has columns: Name, Type, Project, Last use, Created, and Expiration. A new token 'Jenkins' has been created, with a 'Revoke' button next to it. The 'Generate Tokens' section is also visible, showing a form to generate a new token.

Administration

Configuration Security Projects System Marketplace

General Users Groups Global Permissions Permission Templates

SCM Accounts Last connection Groups Tokens

A Administrator admin < 1 hour ago sonar-administrators sonar-users Update Tokens

Tokens of Administrator

Generate Tokens

Name Expires in

Enter Token Name 30 days Generate

New token "Jenkins" has been created. Make sure you copy it now, you won't be able to see it again!

Copy squ_21d162904c1c72cf8b39665f96480185c99dc2f9

Name	Type	Project	Last use	Created	Expiration	
Jenkins	User		Never	September 8, 2023	October 8, 2023	Revoke

copy Token

Goto Jenkins Dashboard → Manage Jenkins → Credentials → Add Secret Text. It should look like this

Dashboard > Manage Jenkins > Credentials > System > Global credentials (unrestricted) >

New credentials

Kind
Secret text

Scope ?
Global (Jenkins, nodes, items, all child items, etc)


Secret
POST THE TOKEN HERE

ID ?
Sonar-token

Description ?
Sonar-token

Create

Credentials that should be available irrespective of domain specification to requirements matching.

ID	Name	Kind	Description
 Sonar-token	sonar	Secret text	sonar

Now, go to Dashboard → Manage Jenkins → System and Add like the below image.

Dashboard > Manage Jenkins > System >

SonarQube servers

If checked, job administrators will be able to inject a SonarQube server configuration as environment variables in the build.

☐ Environment variables Enable injection of SonarQube server configuration as build environment variables

SonarQube installations

List of SonarQube installations

Name

sonar-server

Server URL

Default is http://localhost:9000

http://13.232.17.191:9000

Server authentication token

SonarQube authentication token. Mandatory when anonymous access is disabled.

Sonar-token

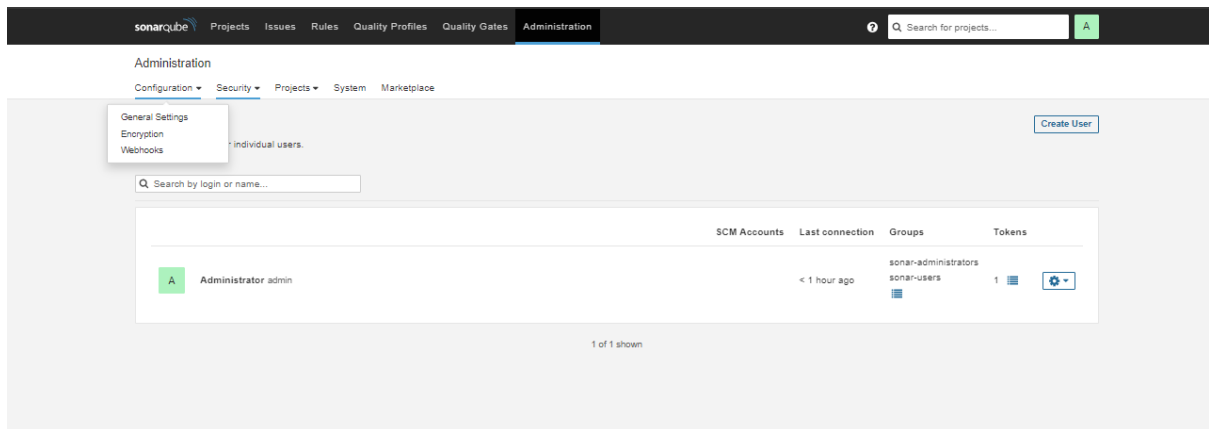
Add

Save Apply

Click on Apply and Save

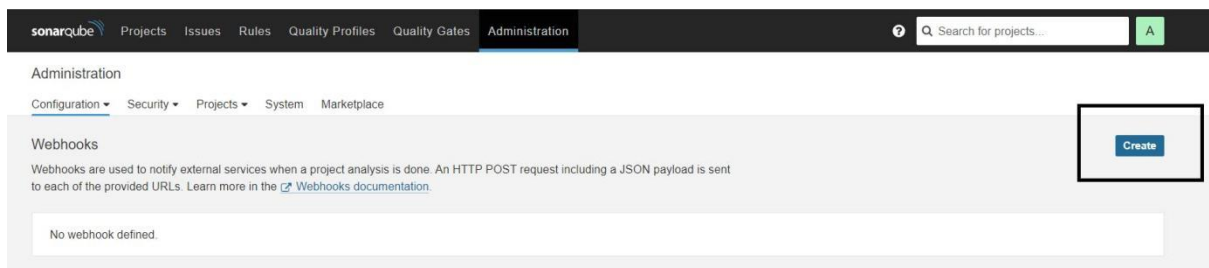
In the Sonarqube Dashboard add a quality gate also

Administration-> Configuration->Webhooks



The screenshot shows the SonarQube Administration interface. The top navigation bar includes 'sonarqube' and links to 'Projects', 'Issues', 'Rules', 'Quality Profiles', 'Quality Gates', and 'Administration'. A search bar is on the right. The 'Administration' section is expanded, showing 'Configuration', 'Security', 'Projects', 'System', and 'Marketplace'. The 'Security' section is further expanded, showing 'General Settings', 'Encryption', and 'Webhooks'. The 'Webhooks' page has a 'Create User' button and a search bar. Below the search bar is a table with columns: 'SCM Accounts', 'Last connection', 'Groups', and 'Tokens'. The table contains one entry: 'Administrator admin' with a last connection of '< 1 hour ago' and two groups: 'sonar-administrators' and 'sonar-users'. The 'Tokens' column shows '1' token. A '1 of 1 shown' message is at the bottom.

SCM Accounts	Last connection	Groups	Tokens
A Administrator admin	< 1 hour ago	sonar-administrators sonar-users	1



The screenshot shows the SonarQube Administration interface. The top navigation bar includes 'sonarqube' and links to 'Projects', 'Issues', 'Rules', 'Quality Profiles', 'Quality Gates', and 'Administration'. A search bar is on the right. The 'Administration' section is expanded, showing 'Configuration', 'Security', 'Projects', 'System', and 'Marketplace'. The 'Configuration' section is further expanded, showing 'Webhooks'. The 'Webhooks' page 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 the description is a text box with the message 'No webhook defined.'

Add details

#in url section of quality gate

<http://jenkins-public-ip:8080>/sonarqube-webhook/

sonarqube Projects Issues Rules Quality Profiles Quality Gates Administration Search for projects... A

Administration

Configuration Security Projects System Marketplace

Webhooks Create

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](#).

Name	URL	Has secret?	Last delivery	Actions
Jenkins	http://15.206.148.142:8080/sonarqube-webhook/	No	Never	⚙️

Now add Docker credentials to the Jenkins to log in and push the image

Manage Jenkins -> Credentials -> global -> add credential

Add DockerHub Username and Password under Global Credentials

Jenkins Search (CTRL+K) 🔒 1 AIworkLabs

Dashboard > Manage Jenkins > Credentials > System > Global credentials (unrestricted) >

Global credentials (unrestricted) + Add Credentials

Credentials that should be available irrespective of domain specification to requirements matching.

ID	Name	Kind	Description
Sonar-token	Sonar-token	Secret text	Sonar-token 🔗
Dockerhub	aiworklabs/***** (Dockerhub)	Username with password	Dockerhub 🔗

Icon: S M L

Now let's create a new job for our pipeline create new one with name "docker"

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 this script

```
pipeline{
  agent any
  tools{
    jdk 'jdk17'
    nodejs 'node16'
  }
  environment {
    SCANNER_HOME=tool 'sonar-scanner'
  }
  stages {
    stage('clean workspace'){
      steps{
        cleanWs()
      }
    }
  }
}
```

```

    }
    stage('Checkout from Git'){
        steps{
            git branch: 'main', url:
'https://github.com/RajinikanthVadla/Tetris-V1.git'
        }
    }
    stage("Sonarqube Analysis "){
        steps{
            withSonarQubeEnv('sonar-server') {
                sh "' $SCANNER_HOME/bin/sonar-scanner -
Dsonar.projectName=TetrisVersion1.0 \
-Dsonar.projectKey=TetrisVersion1.0 '"
            }
        }
    }
    stage("quality gate"){
        steps {
            script {
                waitForQualityGate abortPipeline: false, credentialsId: 'Sonar-
token'
            }
        }
    }
    stage('Install Dependencies') {
        steps {

```

```

        sh "npm install"
    }
}
stage('OWASP FS SCAN') {
    steps {
        dependencyCheck additionalArguments: '--scan ./ --
disableYarnAudit --disableNodeAudit', odcInstallation: 'DP-Check'
        dependencyCheckPublisher pattern: '**/dependency-check-
report.xml'
    }
}
stage('TRIVY FS SCAN') {
    steps {
        sh "trivy fs . > trivyfs.txt"
    }
}
stage("Docker Build & Push"){
    steps{
        script{
            withDockerRegistry(credentialsId: 'docker', toolName: 'docker'){
                sh "docker build -t tetrisv1 ."
                sh "docker tag tetrisv1 aiworklabs/workshop:latest "
                sh "docker push aiworklabs/workshop:latest "
            }
        }
    }
}

```

```

    }

    stage("TRIVY"){

        steps{

            sh "trivy image sevenajay/tetrisv1:latest > trivyimage.txt"

        }

    }

}

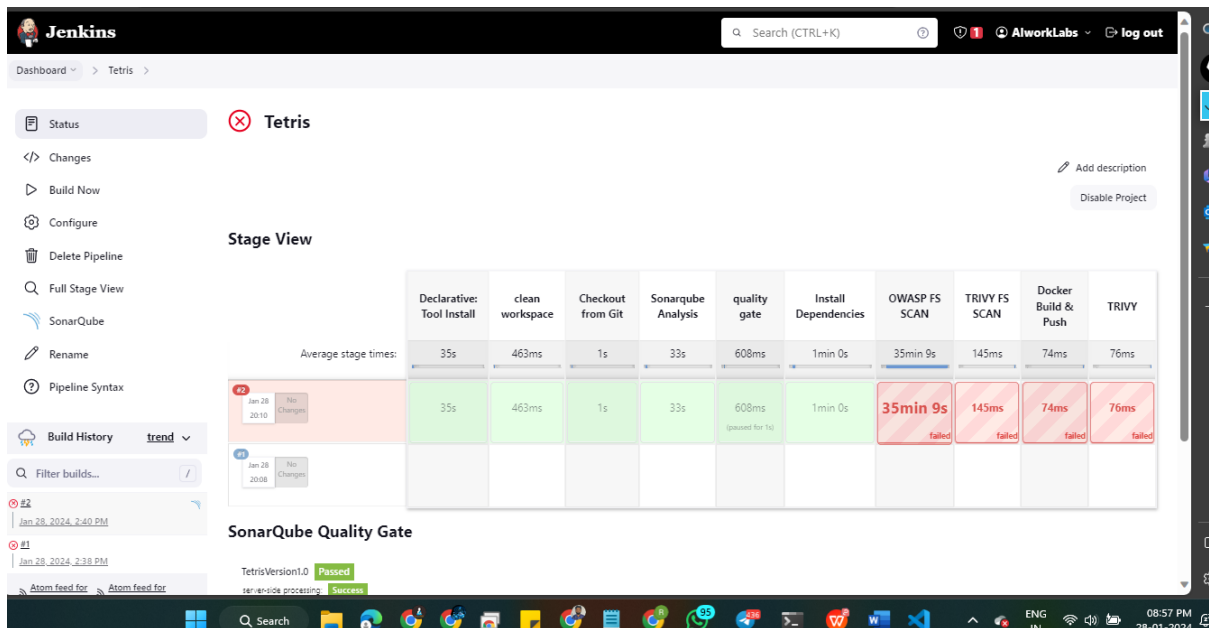
```

The screenshot shows the Jenkins web interface for a pipeline named 'Tetris'. The top navigation bar includes the Jenkins logo, a search bar, and user information. The left sidebar contains navigation links: Status, Changes, Build Now, Configure, Delete Pipeline, Full Stage View, Rename, and Pipeline Syntax. The main content area displays the 'Stage View' of the pipeline. A table shows the stages and their durations for two builds. Build #2 (Jan 28, 2024, 2:40 PM) is highlighted in blue, indicating it is the current build. Build #1 (Jan 28, 2024, 2:08 PM) is shown below it. The stages and their durations are as follows:

Stage	Build #2 (Jan 28, 2024, 2:40 PM)	Build #1 (Jan 28, 2024, 2:08 PM)
Declarative: Tool Install	35s	35s
clean workspace	463ms	463ms
Checkout from Git	1s	1s
Sonarqube Analysis	33s	33s
quality gate	608ms (paused for 1s)	608ms
Install Dependencies	1min 0s	1min 0s
OWASP FS SCAN	1min 12s	1min 12s

The 'Average stage times' are also displayed above the table. The 'Build History' section on the left shows the list of builds, with build #2 being the most recent and highlighted.

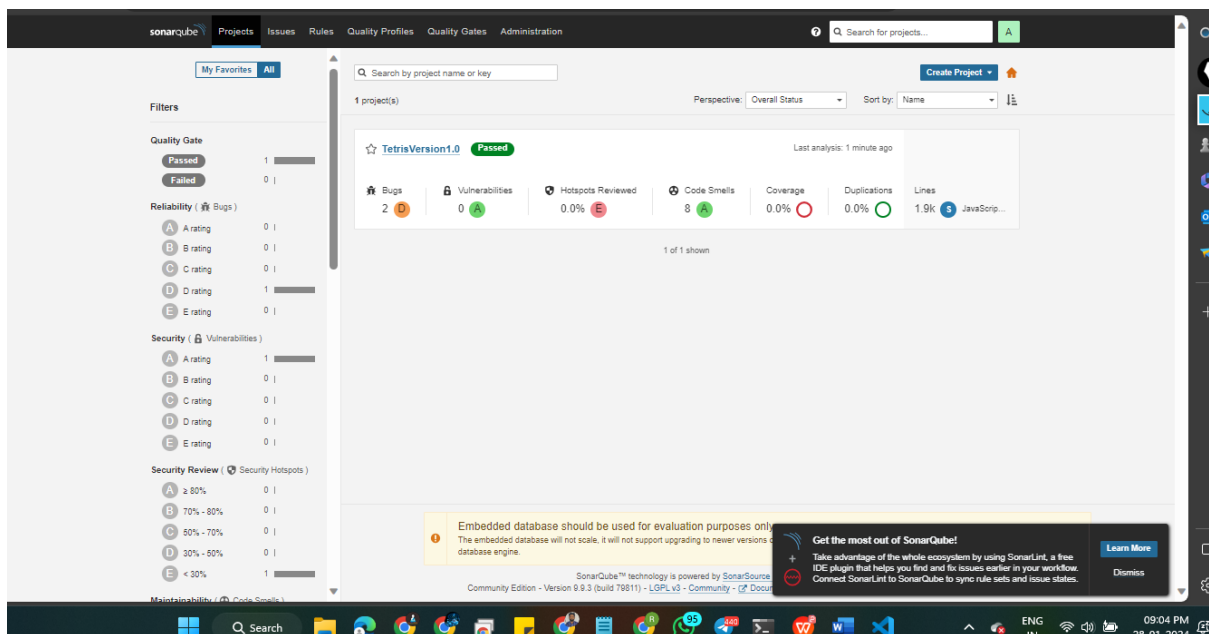
Let this pipeline complete
but I have a error

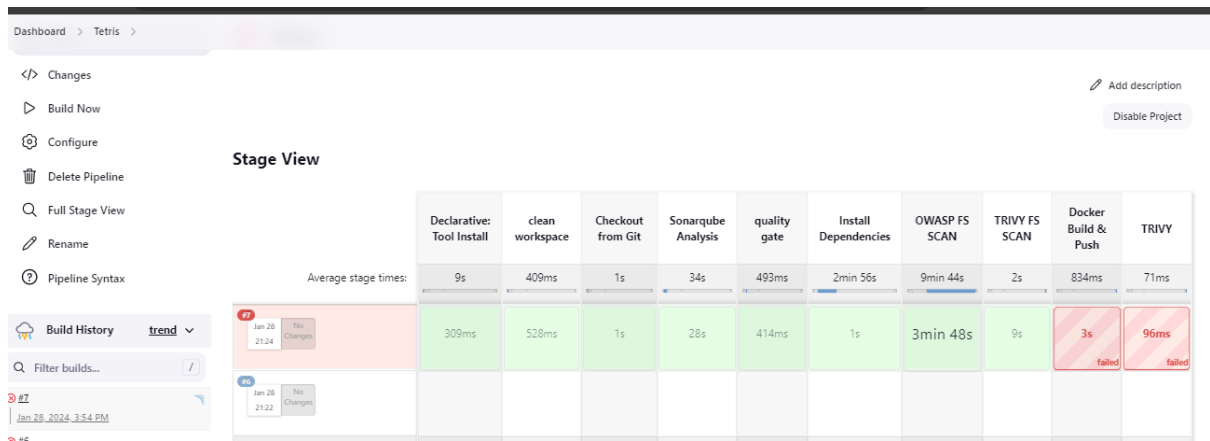


lets re-run this pipeline

squ_388c7c2a0a9e5141c0662e926667e43728c2cccf

Then after the you can see sonarqube dashboard





you have to get failure if not you doing copy paste or simply not doing anything ---- request try yourself to solve errors

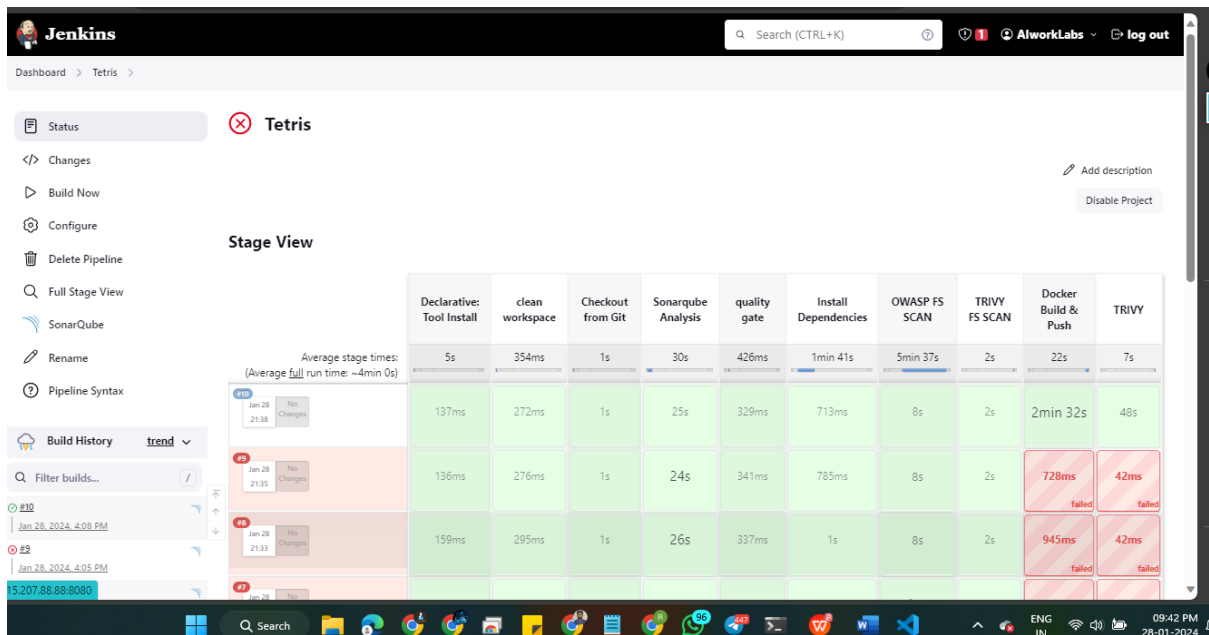
```
sudo usermod -aG docker ubuntu
```

```
newgrp docker
```

```
sudo chmod 777 /var/run/docker.sock
```

run above above commands if you are getting error

-



Let's add the Image Updater stage to the Pipeline

Go to Terminal of your Jenkins instance SSH and enter the below command

```
aws eks update-kubeconfig --name <CLUSTER NAME> --region <CLUSTER REGION>
```

```
aws eks update-kubeconfig--name EKS_CLOUD--region ap-south-1
```

then see `kubectl get nodes` in server Jenkins-Argo server

`kubectl get nodes`

ARGO CD SETUP

Let's install ArgoCD

ARGOCD INSTALLATION LINK

You will be redirected to this page



Then run these commands

```
kubectl create namespace argocd
```

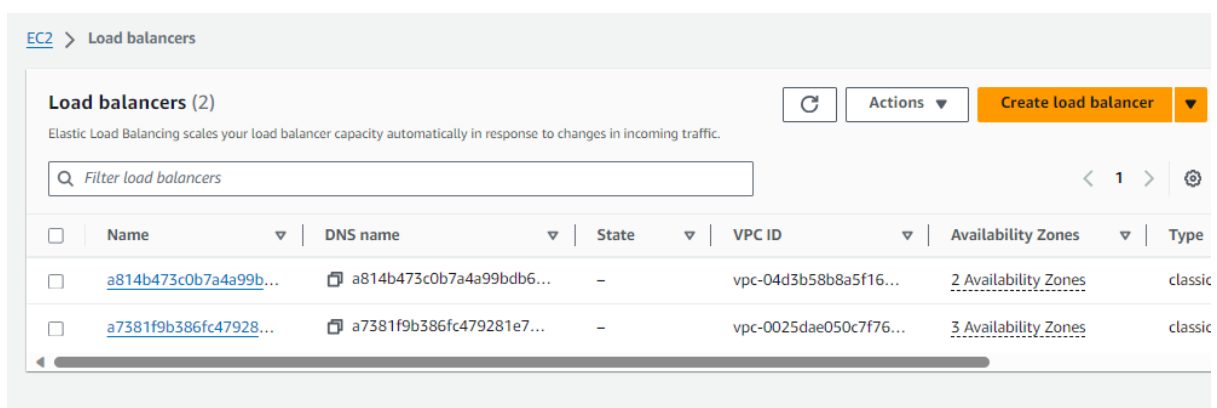
```
kubectl apply -n argocd -f https://raw.githubusercontent.com/argoproj/argo-cd/v2.4.7/manifests/install.yaml
```

COMMANDS ARGOCD

By default, argocd-server is not publicly exposed. For this project, we will use a Load Balancer to make it usable:

Run This command mandatory

```
kubectl patch svc argocd-server -n argocd -p '{"spec": {"type": "LoadBalancer"}}'
```



load balancer will create automatically

&& patched with above command

Wait about 2 minutes for the LoadBalancer creation

```
sudo apt install jq -y
```

```
export ARGOCD_SERVER=`kubectl get svc argocd-server -n argocd -o json | jq --raw-output '.status.loadBalancer.ingress[0].hostname'`
```

when you run this command, it will export the hostname of the ArgoCD server's load balancer and store it in the ARGOCD_SERVER environment variable, which you can then use in other commands or scripts to interact with the ArgoCD server. This can be useful when you need to access the ArgoCD web UI or interact with the server programmatically.

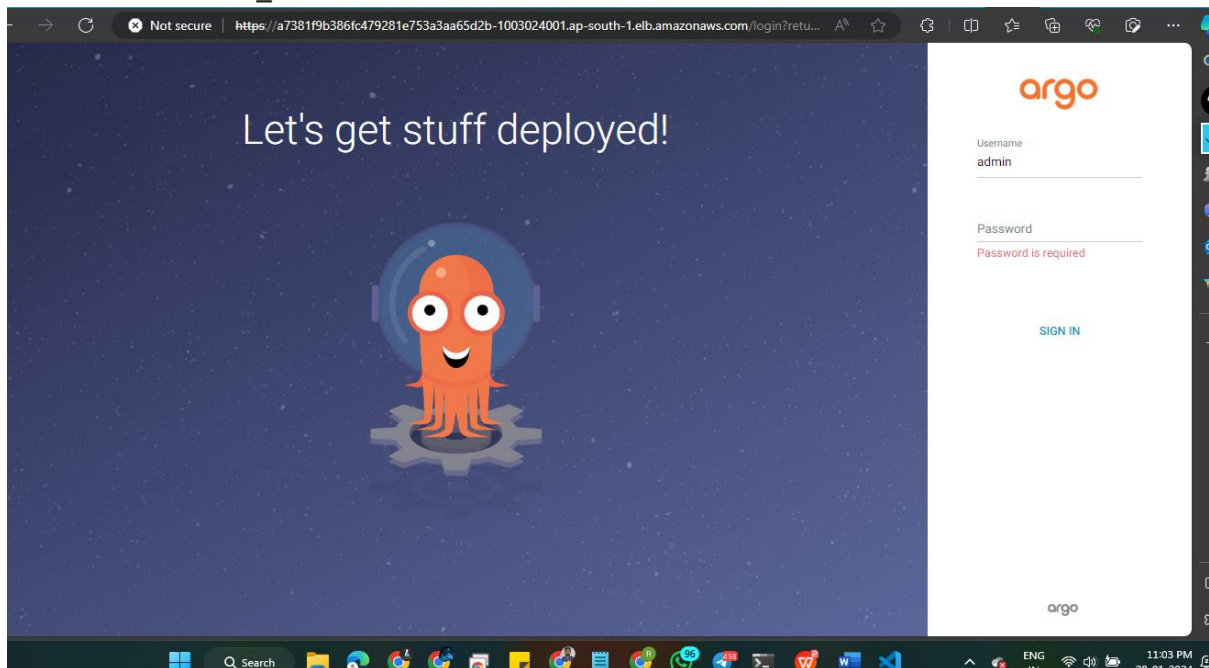
If run this command you will get the load balancer external IP

If you want to see your password provide the below command

```
echo $ARGO_PWD
```

Now copy the load balancer IP and paste it into the browser

```
echo $ARGOCD_SERVER
```



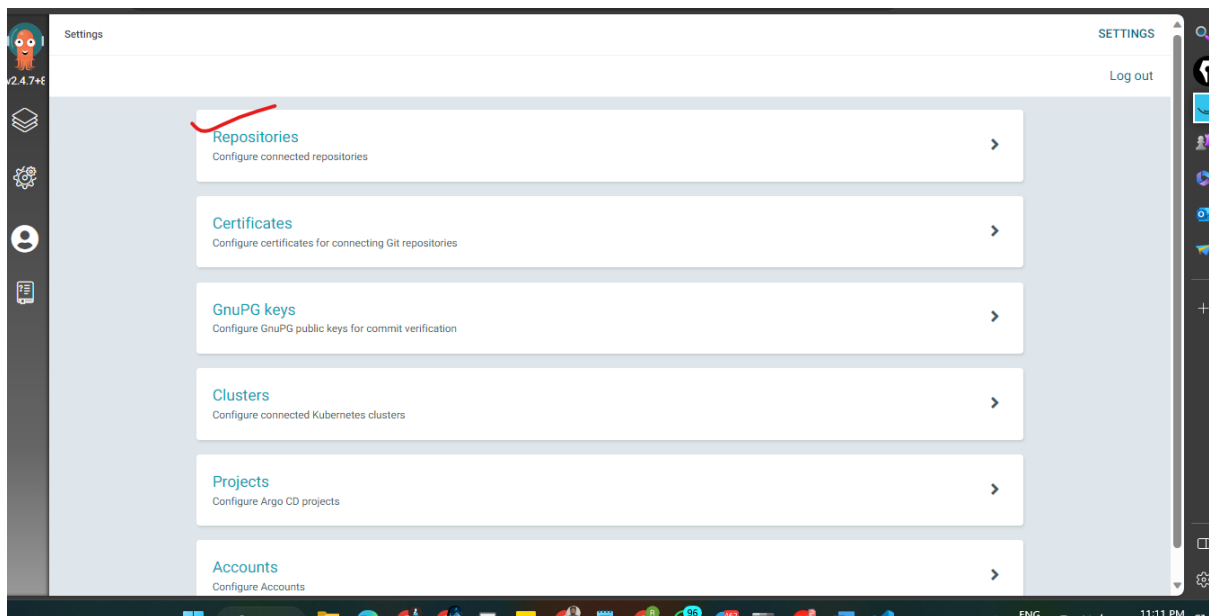
Username is admin

For the password, you have to provide the below command and copy it

```
export ARGO_PWD=`kubectl -n argocd get secret argocd-initial-admin-secret -o jsonpath="{.data.password}" | base64 -d`
```

```
echo $ARGO_PWD
```

```
root@ip-172-31-0-210:/home/ubuntu# echo "ARGO_PWD is: $ARGO_PWD"
ARGO_PWD is:
root@ip-172-31-0-210:/home/ubuntu# export ARGO_PWD=`kubectl -n argocd get secret argocd-initial-admin-secret -o jsonpath="{.data.password}" | base64 -d`
root@ip-172-31-0-210:/home/ubuntu# echo "$ARGO_PWD"
fftsv8FgDijJu8IZ3
```



after configuring the git repo things

Applications

CREATECANCEL

NEW APPSYNC APPS

GENERAL

EDIT AS YAML

Application Name

testris

Project Name

default

SYNC POLICY

Automatic

☐ PRUNE RESOURCES

☐ SELF HEAL

☐ SET DELETION FINALIZER

SYNC OPTIONS

☐ SKIP SCHEMA VALIDATION

☐ PRUNE LAST

☐ RESPECT IGNORE DIFFERENCES


☐ AUTO-CREATE NAMESPACE

☐ APPLY OUT OF SYNC ONLY

Applications

NEW APPSYNC APPSREFRESH APPS

Search applications...



No applications yet

Create new application to start managing resources in your cluster

CREATE APPLICATION

Applications

CREATE

CANCEL

Repository URL

https://github.com/RajinikanthVadla/Tetris-V1.git

GIT ✓

Revision

HEAD

Branches ▾

Path

./

DESTINATION

Cluster URL

https://kubernetes.default.svc

URL ▾

Namespace

default

Then its will connect the app github repo

Applications

APPLICATIONS TITLE

+ NEW APP

SYNC APPS

REFRESH APPS

Search applications...

Filters

★ Favorites Only

SYNC STATUS

Unknown 0

Synced 1

OutOfSync 0

HEALTH STATUS

Unknown 0

Progressing 0

Suspended 0

Healthy 1

Degraded 0

Missing 0

LABELS

LABELS

tetris

Project: default

Labels:

Status: Healthy Synced

Reposi... https://github.com/RajinikanthVadla/...

Target ... HEAD

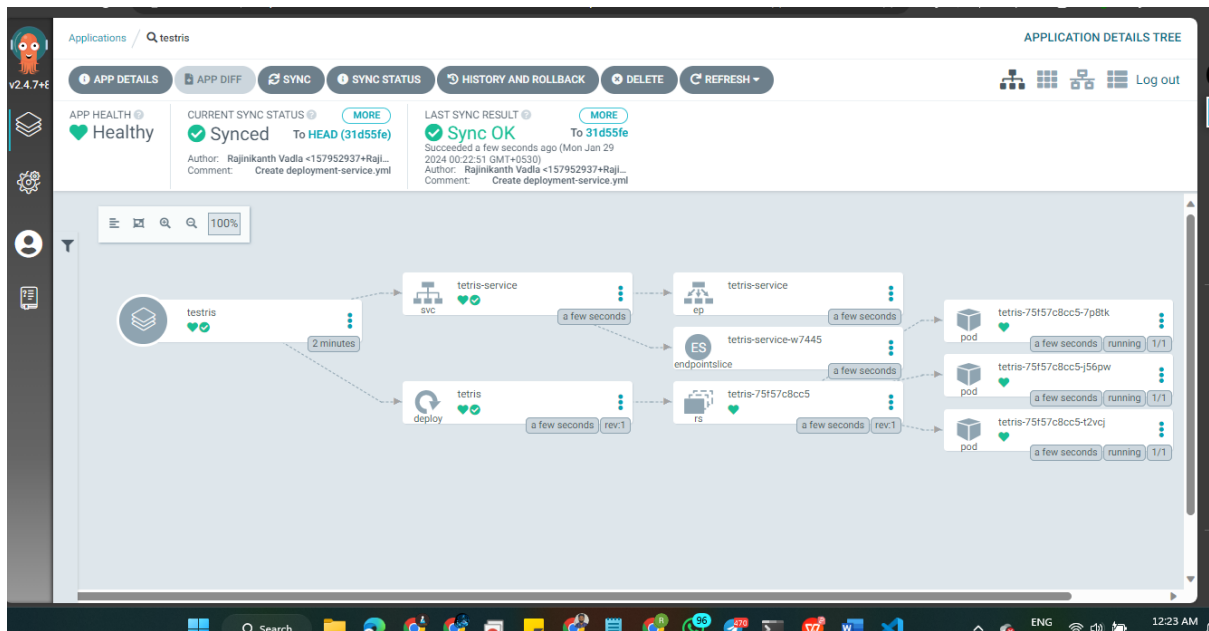
Path: src

Destin... in-cluster

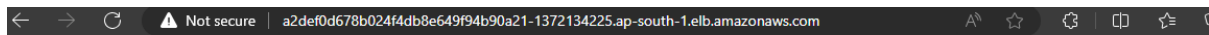
Names... default

SYNC

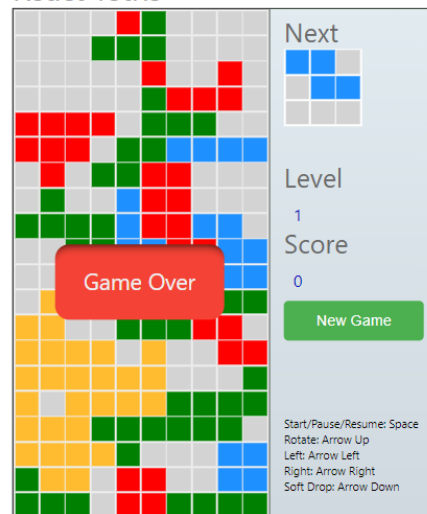
Windows Taskbar



take LoadBalancer dns and map with dns



React Tetris



helm repo add grafana <https://grafana.github.io/helm-charts>

helm repo update


```
helm install grafana grafana/grafana \
--namespace monitoring --create-namespace
```

```
kubectl get secret --namespace monitoring grafana -o jsonpath="{.data.admin-password}" | base64 -
-decode ; echo
```

```
kubectl expose deployment grafana --namespace monitoring --type=NodePort --name=grafana-
service
```

```
helm repo add prometheus-community https://prometheus-community.github.io/helm-charts
```

```
helm install prometheus prometheus-community/kube-prometheus-stack --namespace monitoring
```

Add Prometheus Data Source to Grafana

Log in to Grafana.

Navigate to Configuration > Data Sources.

Add a new Prometheus data source:

URL: `http://prometheus-server:80`

8. Import Kubernetes Monitoring Dashboard

Grafana has pre-built dashboards to monitor Kubernetes nodes and pods.

Go to Create > Import.

Use the following dashboard IDs (examples):

6417 (Kubernetes Monitoring)

12740 (Kubernetes All-In-One)

```
helm install grafana grafana/grafana \
--namespace monitoring \
--set adminPassword='EKS!sAWSome' \
--values grafana.yaml \
```

```
--set service.type=LoadBalancer
```

```
kubectll get all -n Grafana
```

-----or-----

```
helm repo add prometheus-community https://prometheus-community.github.io/helm-charts
```

```
helm repo add grafana https://grafana.github.io/helm-charts
```

```
helm repo update
```

```
helm install prometheus prometheus-community/kube-prometheus-stack --namespace devops
```

```
helm install grafana grafana/grafana \
```

```
--namespace devops \
```

```
--set adminPassword='YourAdminPassword' \
```

```
--set service.type=LoadBalancer
```

```
kubectll get svc --namespace devops grafana -o jsonpath='{.status.loadBalancer.ingress[0].ip}'
```

```
kubectll get svc --namespace devops
```

```
kubectll edit svc prometheus1-kube-prometheu-prometheus -n monitoring
```

```
#to edit Clusert Ip to Load balancer
```