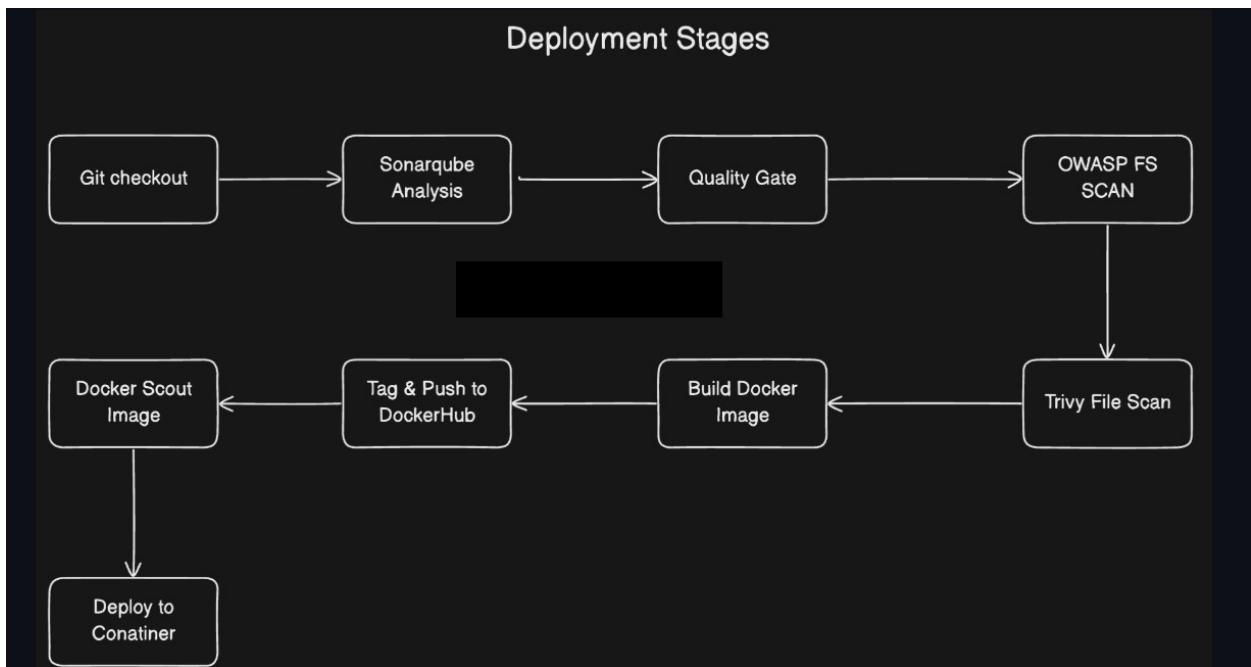
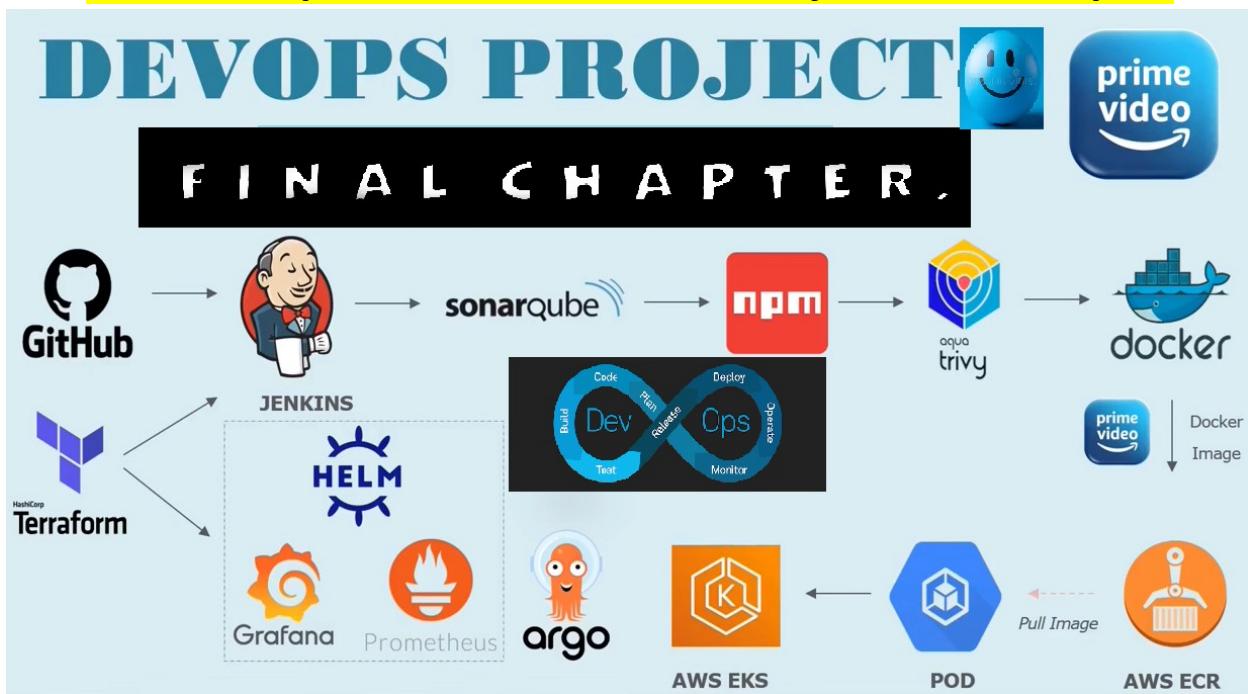


## AWS DevOps- CI/CD Automation Project- Final Chapter



This is a fully automated DevOps CI/CD project. In this project, we automate a complete DevOps pipeline using Jenkins, SonarQube, Docker, ECR (Amazon Elastic Container Registry), and AWS services. Set up a Jenkins pipeline to build and deploy an application into AWS, with quality checks using SonarQube and security scans using Trivy. The pipeline will also push Docker images to AWS ECR and handle Docker image cleanup. I will walk you through step-by-step, from Jenkins setup to deployment. I have added --how 2 create sub-domain from the load balancer endpoint using Route53.

## Overview:

This project focuses on implementing a Continuous Integration (CI) and Continuous Deployment (CD) pipeline using Jenkins. The goal is to deploy a clone of the Amazon Prime Video application to Amazon Elastic Container Registry (ECR) while incorporating security vulnerability scanning using [Trivy](#). The pipeline also integrates SonarQube for static code analysis and quality gate checks.

**This project uses: Jenkins for Continuous Integration (CI) SonarQube for Code Quality Analysis Docker for Containerization AWS ECR for storing Docker images Trivy for vulnerability scanning Terraform for infrastructure as code (IAC)**

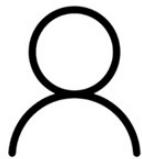
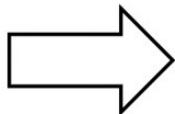
## Introduction

- Pre-requisites
- Configuration
- Creating Jenkins server using Terraform
- Access Jenkins Server
- Configure SonarQube
- Configure Jenkins
- Build Pipeline Overview
- Writing Build pipeline code from scratch
- Running Build pipeline
- Creating EKS using Terraform
- Deployment pipeline overview
- Running Deployment pipeline
- Access Prometheus, Grafana & ArgoCD
- Deploy Application using ArgoCD
- Access Amazon prime Application
- Creating sub-domain using Route53
- Access Grafana & create dashboard
- Running Cleanup pipeline
- Cleanup Jenkins Server & AWS EKS using Terraform

# PRE- REQUISITES



# CONFIGURATION



## 💡 Tools & Technologies:

- **Jenkins**: An automation server used to manage the CI/CD pipeline.
- **SonarQube**: A platform for continuous inspection of code quality and security vulnerabilities.
- **Trivy**: A security vulnerability scanner specifically for Docker images.
- **Docker**: A containerization platform used to package the application into lightweight containers.
- **AWS ECR (Elastic Container Registry)**: A fully managed Docker container registry for storing Docker images.
- **AWS CLI**: Command-line tool to interact with AWS services.

# Go to AWS IAM .. create user – give Admin Access Permission

The screenshot shows the AWS IAM User Details page for a user named 'devops-project-user'. The user was created on October 20, 2024, at 07:16 UTC+05:30. It has an ARN of arn:aws:iam::309395755719:user/devops-project-user and its console access is disabled. There is one policy attached: 'Permissions policies (1)'. The 'Security credentials' tab is selected, showing one access key available for creation. The left sidebar shows the IAM navigation menu with 'Users' selected.

# now Generate Access Key

## Access key best practices & alternatives Info

Avoid using long-term credentials like access keys to improve your security. Consider the following use cases and alternatives.

### Use case

#### Command Line Interface (CLI)

You plan to use this access key to enable the AWS CLI to access your AWS account.

#### Local code

You plan to use this access key to enable application code in a local development environment to access your AWS account.

### > Create access key

## Set description tag - *optional* Info

The description for this access key will be attached to this user as a tag and shown alongside the access key.

### Description tag value

Describe the purpose of this access key and where it will be used. A good description will help you rotate this access key confidently later.

aws-access-keys-secret-key

Maximum 256 characters. Allowed characters are letters, numbers, spaces representable in UTF-8, and: \_ . : / = + - @

Cancel

Previous

Create access key

## Retrieve access keys Info

### Access key

If you lose or forget your secret access key, you cannot retrieve it. Instead, create a new access key and make the old key inactive.

Access key ►

Secret access key

 AKIAUQCLMSLD5NUMX5FZ

 \*\*\*\*\* [Show](#)

### Access key best practices

```
# Access key:  
AKIAUQCLMSLD5NUMX5FZ  
  
# Secret key: ➔  
No2KBJb8bKUKLzottBzlcaG4Huh+rVHy+MOg8aMU
```

## # Jenkin Plugins

```
# JENKINS PLUGINS  
SonarQube scanner  
NodeJS  
Pipeline: Stage View  
Docker, Docker Commons, Docker pipeline, Docker API, docker-build-step  
Eclipse Temurin Installer  
Prometheus metrics
```

## ✚ Pipeline Stages:

### 1. Git Checkout:

This stage pulls the source code from a GitHub repository, which contains the Amazon Prime Video clone project.

### 2. SonarQube Analysis:

Performs static code analysis using SonarQube to check for code smells, bugs, and security vulnerabilities. It uses the pre-configured SonarQube scanner.

### 3. Quality Gate:

Ensures that the code meets the defined quality standards using SonarQube's Quality Gate. If the quality gate fails, the pipeline stops.

### 4. Install npm Dependencies:

This stage installs the required npm packages for the application using the command `npm install`.

### 5. Trivy Security Scan:

Scans the application's directory for known vulnerabilities using Trivy. The scan results are stored in a file named `trivy.txt`.

## 6. Build Docker Image:

This stage builds the Docker image for the application using the Dockerfile found in the repository. The image is tagged based on the user-defined ECR repository name.

## 7. Create ECR Repository (if it doesn't exist):

The pipeline checks whether the specified ECR repository exists in the AWS account. If not, it creates the repository automatically. This step ensures that the Docker image has a valid destination for storage.

## 8. Tag & Push Docker Image to AWS ECR:

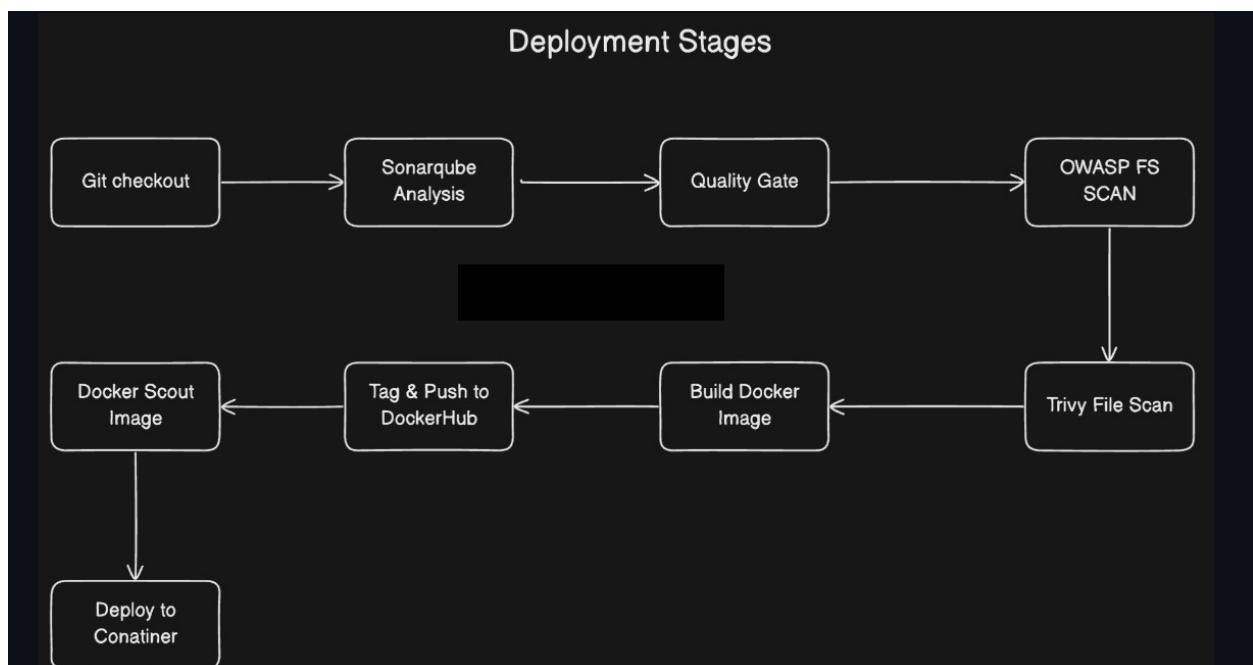
The Docker image is tagged with both a unique build number and the latest tag. Afterward, the image is pushed to the specified ECR repository in the user's AWS account.

## 9. Delete Docker Images from Jenkins Server

Once the images are pushed to ECR, the unused (dangling) docker images will be deleted to avoid storage issues.

## 10. Create ArgoCD, Grafana & Prometheus

The pipeline will create ArgoCD, Grafana & Prometheus. ArgoCD will be used to deploy Docker image from ECR. Grafana & Prometheus will be used for monitoring the application.



#Key pair for SSH Generation

**Resources**

You are using the following Amazon EC2 resources in the US East (N. Virginia) Region:

Instances (running)	Auto Scaling Groups
Capacity Reservations	Dedicated Hosts
Elastic IPs	Instances
Key pairs	Load balancers
Placement groups	Security groups
Snapshots	Volumes

**Launch instance**  
To get started, launch an Amazon EC2 instance, which is a virtual server in the cloud.

**Service health**

[AWS Health Dashboard](#)

**Zones**

[EC2](#) > [Key pairs](#) > Create key pair

## Create key pair Info

### Key pair

A key pair, consisting of a private key and a public key, is a set of security credentials that you use to prove your identity when connecting to an instance.

#### Name

The name can include up to 255 ASCII characters. It can't include leading or trailing spaces.

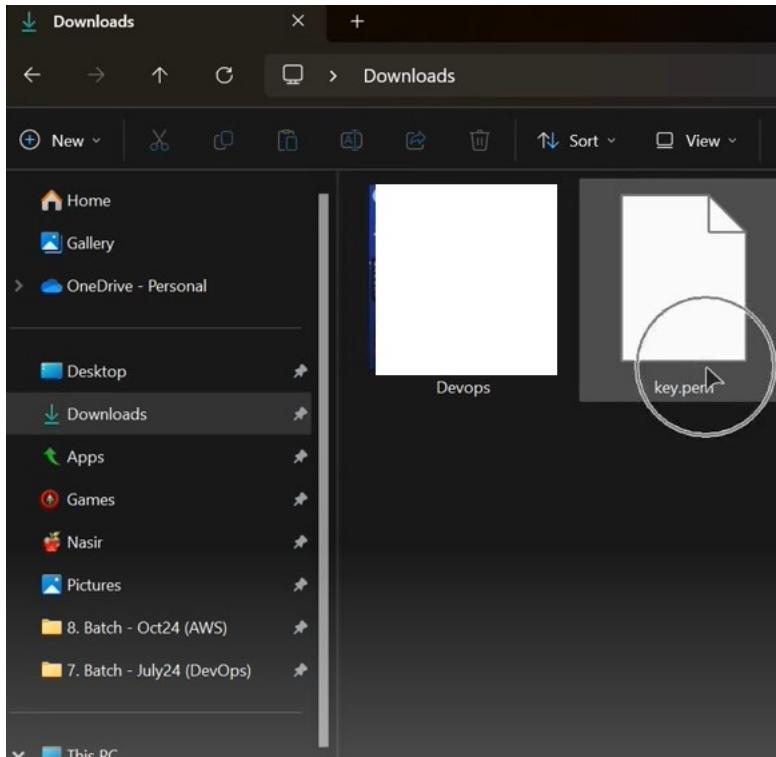
#### Key pair type Info

 RSA ED25519

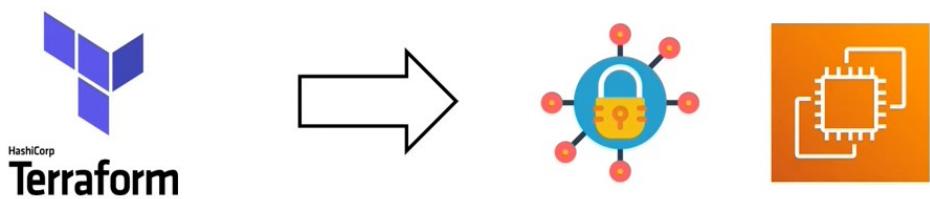
#### Private key file format

.pem  
For use with OpenSSH

.ppk  
For use with PuTTY



# CREATE INFRA USING TERRAFORM



## **INFRASTRUCTURE CONFIGURATION:**

### **EC2 Instance:**

- **Operating System:** Ubuntu 24.04

### **Security Group:**

- **SSH:** Port 22
- **HTTP:** Port 80
- **HTTPS:** Port 443
- **NodeJS:** Port 3000
- **Jenkins:** Port 8080
- **SonarQube:** Port 9000

### **Jenkins Plugins:**

#### 1. **SonarQube Scanner:**

Used to integrate Jenkins with SonarQube to check code quality and perform quality gate checks.

#### 2. **NodeJS:**

Manages and allows the use of different versions of Node.js within Jenkins.

#### 3. **Pipeline: Stage View:**

Provides a visual interface to monitor the stages of the Jenkins pipeline.

#### 4. **Docker:**

Allows Jenkins to manage Docker containers directly.

#### 5. **Docker Commons:**

Facilitates the sharing of common Docker configurations across Jenkins jobs.

#### 6. **Docker Pipeline:**

Supports building and running Docker containers within Jenkins pipelines.

#### 7. **Docker API:**

Provides an API to interact with Docker within Jenkins.

#### 8. **docker-build-step:**

Adds Docker-related build steps, like building images or running containers in freestyle jobs.

#### 9. **SSH Agent:**

Manages SSH credentials for remote server access within Jenkins pipelines.

#### 10. **Eclipse Temurin Installer:**

Used to install OpenJDK within Jenkins.

#### 11. **Prometheus Metrics:**

Enables Jenkins to expose metrics for monitoring and alerting.

## **SONARQUBE CONFIGURATION:**

### **SonarQube Webhook:**

The SonarQube webhook is used to send analysis results back to Jenkins.

- **Configuration:**
  - Name: sonarqube-webhook
  - URL: http://<jenkins-url>:8080/sonarqube-webhook/
  - Secret: None
- ❖ **SonarQube Token:**
  - Generated token to authenticate Jenkins with SonarQube for analysis and quality gate checks.
- ❖ **Credentials:**
  - **SonarQube Token:** Used to authenticate and interact Jenkins with SonarQube.
  - **AWS credentials (Access Key and Secret Key):** These credentials are used to authenticate and interact Jenkins with AWS, to push the Docker Image to AWS ECR.
- ❖ **Jenkins SonarQube System Configuration:**
  - Helps to store SonarQube server details globally in Jenkins instead of specifying them in each pipeline script.
  - **Configuration:**
    - Name: sonar-server
    - URL: http://<sonar-server-url>:9000

## **JENKINS GLOBAL TOOL CONFIGURATION:**

### **⊕ JDK Installation:**

- **Name:** jdk17
- **Install From:** adoptium.net
- **Version:** jdk17.0.8.1+1

### **⊕ SonarQube Scanner Installation:**

- **Name:** sonar-scanner
- **Install From:** Maven Central
- **Version:** 6.2.0.4584

### **⊕ NodeJS Installation:**

- **Name:** node16
- **Install From:** nodejs.org
- **Version:** 16.20.0

### **⊕ Docker Installation:**

- **Name:** docker
- **Install From:** docker.com
- **Version:** Latest

### **⊕ Flow of Execution:**

1. **Source Code Fetching:** The pipeline begins by pulling the code from GitHub.
2. **Static Analysis:** SonarQube performs code quality and security checks on the pulled code.
3. **Build & Testing:** The application's dependencies are installed, and the application is packaged into a Docker image.
4. **Security Scanning:** Trivy scans the Docker image for vulnerabilities and produces a report.
5. **Deployment:** The final Docker image is pushed to AWS ECR, making it ready for deployment to AWS ECS or EKS. The application is deployed using ArgoCD. The pods are monitored using Grafana & Prometheus.

## Conclusion:

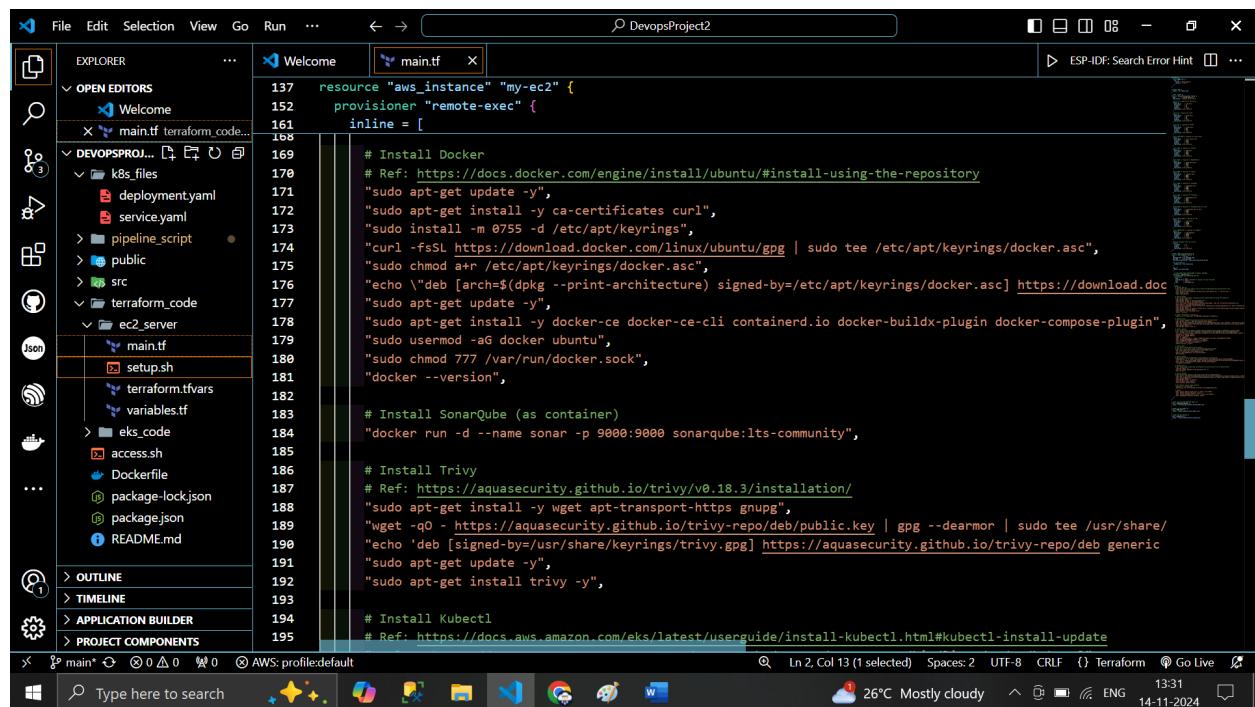
This project provides a comprehensive CI/CD pipeline from source code management to deployment with integrated code quality checks and security scans. Using AWS ECR ensures that the Dockerized application is securely stored and readily available for deployment in a scalable AWS environment. The application is deployed using ArgoCD. The pods are monitored using Grafana & Prometheus.

|

## # Practicals

### # Create Terraform main.tf that initiates all necessary microservices

Docker /ec2/argocd/sonarQube/grafana/Prometheus/Trivy/k8s/java/helm/java17



The screenshot shows a Windows terminal window titled "DevopsProject2". The left pane displays a file tree with several folders like "k8s\_files", "public", "src", and "terraform\_code". The right pane shows the content of the "main.tf" file. The file contains Terraform code for provisioning an AWS instance, installing Docker, SonarQube, Trivy, and Kubernetes components, and setting up a Java application. The code includes URLs for Docker, SonarQube, and Trivy installations, as well as specific commands for each component's setup. The terminal also shows system status at the bottom, including weather (26°C Mostly cloudy), battery level (13:31), and network connectivity (ENG 14-11-2024).

```
resource "aws_instance" "my-ec2" {
  provisioner "remote-exec" {
    inline = [
      "# Install Docker
      # Ref: https://docs.docker.com/engine/install/ubuntu/#install-using-the-repository
      "sudo apt-get update -y",
      "sudo apt-get install -y ca-certificates curl",
      "sudo install -m 0755 -d /etc/apt/keyrings",
      "curl -fsSL https://download.docker.com/linux/ubuntu/gpg | sudo tee /etc/apt/keyrings/docker.asc",
      "sudo chmod a=r /etc/apt/keyrings/docker.asc",
      "echo \"deb [arch=$(dpkg --print-architecture) signed-by=/etc/apt/keyrings/docker.asc] https://download.docker.com/linux/ubuntu/stable\" | sudo tee /etc/apt/sources.list.d/docker.list",
      "sudo apt-get update -y",
      "sudo apt-get install -y docker-ce docker-ce-cli containerd.io docker-buildx-plugin docker-compose-plugin",
      "sudo usermod -aG docker ubuntu",
      "sudo chmod 777 /var/run/docker.sock",
      "docker --version",

      # Install SonarQube (as container)
      "docker run -d --name sonar -p 9000:9000 sonarqube:lts-community",

      # Install Trivy
      # Ref: https://aquasecurity.github.io/trivy/v0.18.3/installation/
      "sudo apt-get install -y wget apt-transport-https gnupg",
      "wget -qO - https://aquasecurity.github.io/trivy-repo/deb/public.key | gpg --dearmor | sudo tee /usr/share/debian/trivy.gpg",
      "echo 'deb [signed-by=/usr/share/keyrings/trivy.gpg] https://aquasecurity.github.io/trivy-repo/deb generic' | sudo tee /etc/apt/sources.list.d/trivy.list",
      "sudo apt-get update -y",
      "sudo apt-get install trivy -y",

      # Install Kubectl
      # Ref: https://docs.aws.amazon.com/eks/latest/userguide/install-kubectl.html#kubectl-install-update
    ]
  }
}
```

#note key.pem file must be placed in same folder of terraform main.tf

```

k8s_files
> pipeline_script
> public
> src
> terraform_code
  > ec2_server
    & key.pem
  main.tf
    $ setup.sh
    & terraform.tfvars
    & variables.tf
  > eks_code
$ access.sh
  Dockerfile
{} package-lock.json
{} package.json
  Project_WriteUp.docx
  README.md

```

```

resource "aws_instance" "my-ec2" {
  provisioner "remote-exec" {
    inline = [
      "echo \"deb [signed-by=/usr/share/keyrings/jenkins-keyring.asc] https://pk",
      "sudo apt-get update -y",
      "sudo apt-get install -y jenkins",
      "sudo systemctl start jenkins",
      "sudo systemctl enable jenkins",
      "# Get Jenkins initial login password",
      "ip=$(curl -s ifconfig.me)",
      "pass=$(sudo cat /var/lib/jenkins/secrets/initialAdminPassword)",
      "# Output",
      "echo 'Access Jenkins Server here --> http://'$ip':8080'",
      "echo 'Jenkins Initial Password: '$pass'",
      "echo 'Access SonarQube Server here --> http://'$ip':9000'",
      "echo 'SonarQube Username & Password: admin'",
      "echo 'Access Grafana Server here --> http://'$ip':3000'",
      "echo 'Access Prometheus Server here --> http://'$ip':9090'"
    ]
  }
}

```

# Terminal

## 2. Initialize and Apply Terraform:

- Run the below commands to reduce the path displayed in VS Code terminal (Optional)

```

code $PROFILE
function prompt {"$PWD > "}
function prompt ${$(Get-Location -Leaf) + " > "}

```

Try the new cross-platform PowerShell <https://aka.ms/pscore6>

```

PS C:\Users\Lenovo\Desktop\AWS-DevOps- Project\DevopsProject2> code $PROFILE
PS C:\Users\Lenovo\Desktop\AWS-DevOps- Project\DevopsProject2> function prompt {"$PWD > "}
C:\Users\Lenovo\Desktop\AWS-DevOps- Project\DevopsProject2> function prompt ${$(Get-Location -Leaf) + " > "}
PS>

```

# AWS Configure .. Connect to AWS Cloud + initiate TF

```

PS>aws configure
AWS Access Key ID [*****KA7K]: AKIAUQCLMSLD5NUMX5FZ
AWS Secret Access Key [*****1Pp]: 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
Default region name [us-east-1]: us-east-1
Default output format [None]:
PS>terraform init
Initializing the backend...
Initializing provider plugins...
- Finding hashicorp/aws versions matching "5.67.0"...
- Installing hashicorp/aws v5.67.0...

```

.terraform Folder is created

Default output format [None]:  
PS>**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 has been successfully initialized!**

You may now begin working with Terraform. Try running "terraform plan" to see any changes that are required for your infrastructure. All Terraform commands should now work.

If you ever set or change modules or backend configuration for Terraform, rerun this command to reinitialize your working directory. If you forget, other commands will detect it and remind you to do so if necessary.

C: > Users > Admin > Documents > WindowsPowerShell > Microsoft.PowerShell\_profile.ps1

```
1 function prompt {"$PWD > "}
```

PROBLEMS OUTPUT DEBUG CONSOLE TERMINAL AZURE PORTS

PS>**terraform apply --auto-approve**

```

    + to_port          = 2380
    },
]
+ name           = "JENKINS-SERVER-SG"
+ name_prefix    = (known after apply)
+ owner_id       = (known after apply)
+ revoke_rules_on_delete = false
+ tags_all       = (known after apply)
+ vpc_id         = (known after apply)
}

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

Changes to Outputs:
+ PRIVATE-IP      = (known after apply)
+ PUBLIC-IP       = (known after apply)
+ SERVER-SSH-ACCESS = (known after apply)
aws_security_group.my-sg: Creating...
aws_security_group.my-sg: Creation complete after 6s [id=sg-0376c4a89f26497fd]
aws_instance.my-ec2: Creating...
aws_instance.my-ec2: Still creating... [10s elapsed]

aws_instance.my-ec2 (remote-exec): No VM guests are running outdated
aws_instance.my-ec2 (remote-exec): hypervisor (qemu) binaries on this
aws_instance.my-ec2 (remote-exec): host.
aws_instance.my-ec2 (remote-exec): Synchronizing state of jenkins.service with SysV service script with /usr/lib
/usr/lib/systemd/systemd-sysv-install.
aws_instance.my-ec2 (remote-exec): Executing: /usr/lib/systemd/systemd-sysv-install enable jenkins
aws_instance.my-ec2 (remote-exec): Access Jenkins Server here --> http://100.24.54.83:8080
aws_instance.my-ec2 (remote-exec): Jenkins Initial Password: 315c46806a9f4eca82f27a2239647548
aws_instance.my-ec2 (remote-exec): Access SonarQube Server here --> http://100.24.54.83:9000
aws_instance.my-ec2 (remote-exec): SonarQube Username & Password: admin
aws_instance.my-ec2 (remote-exec): Access Grafana Server here --> http://100.24.54.83:3000
aws_instance.my-ec2 (remote-exec): Access Prometheus Server here --> http://100.24.54.83:9090
aws_instance.my-ec2: Creation complete after 2m32s [id=i-0858872b9a500cf8e]

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

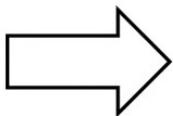
Outputs:

PRIVATE-IP = "172.31.88.75"
PUBLIC-IP = "100.24.54.83"
SERVER-SSH-ACCESS = "ubuntu@100.24.54.83"
PS>[

```

# CONFIGURE SONARQUBE

**sonarqube**



```
# cp Jenkins URL
```

```
/usr/lib/systemd/systemd-sysv-install
aws_instance.my-ec2 (remote-exec): Executing: /usr/lib/systemd/systemd-sysv-install enable jenkins
aws_instance.my-ec2 (remote-exec): Access Jenkins Server here --> http://100.24.54.83:8080
aws_instance.my-ec2 (remote-exec): Jenkins Initial Password: 315c46806a9f4eca82f27a2239647548
aws_instance.my-ec2 (remote-exec): Access SonarQube Server here --> http://100.24.54.83:9000
aws_instance.my-ec2 (remote-exec): SonarQube Username & Password: admin
aws_instance.my-ec2 (remote-exec): Access Grafana Server here --> http://100.24.54.83:3000
aws_instance.my-ec2 (remote-exec): Access Prometheus Server here --> http://100.24.54.83:9090
aws_instance.my-ec2: Creation complete after 2m32s [id=i-0858872b9a500cf8e]
```

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

```
Outputs:
```

```
PRIVATE-IP = "172.31.88.75"
PUBLIC-IP = "100.24.54.83"
SERVER-SSH-ACCESS = "ubuntu@100.24.54.83"
PS>
```



```
# password
```

```
aws_instance.my-ec2 (remote-exec): Executing: /usr/lib/systemd/systemd-sysv-install enable jenkins
aws_instance.my-ec2 (remote-exec): Access Jenkins Server here --> http://100.24.54.83:8080
aws_instance.my-ec2 (remote-exec): Jenkins Initial Password: 315c46806a9f4eca82f27a2239647548
aws_instance.my-ec2 (remote-exec): Access SonarQube Server here --> http://100.24.54.83:9000
aws_instance.my-ec2 (remote-exec): SonarQube Username & Password: admin
aws_instance.my-ec2 (remote-exec): Access Grafana Server here --> http://100.24.54.83:3000
aws_instance.my-ec2 (remote-exec): Access Prometheus Server here --> http://100.24.54.83:9090
aws_instance.my-ec2: Creation complete after 2m32s [id=i-0858872b9a500cf8e]

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

Outputs:

PRIVATE-IP = "172.31.88.75"
PUBLIC-IP = "100.24.54.83"
SERVER-SSH-ACCESS = "ubuntu@100.24.54.83"
PS>
```

## Getting Started

# Unlock Jenkins

To ensure Jenkins is securely set up by the administrator, a password has been written to the log ([not sure where to find it?](#)) and this file on the server:

`/var/lib/jenkins/secrets/initialAdminPassword`

Please copy the password from either location and paste it below.

Administrator password

.....

I

## Getting Started

# Getting Started

✓ Folders	✓ OWASP Markup Formatter	✓ Build Timeout	⌚ Credentials Binding
⌚ Timestamper	⌚ Workspace Cleanup	⌚ Ant	⌚ Gradle
⌚ Pipeline	⌚ GitHub Branch Source	⌚ Pipeline: GitHub Groovy Libraries	⌚ Pipeline Graph View
⌚ Git	⌚ SSH Build Agents	⌚ Matrix Authorization Strategy	⌚ PAM Authentication
⌚ LDAP	⌚ Email Extension	⌚ Mailer	⌚ Dark Theme

\*\* Ionicons API  
Folders  
OWASP Markup Formatter  
\*\* ASM API  
\*\* JSON Path API  
\*\* Structs  
\*\* Pipeline: Step API  
\*\* Token Macro  
Build Timeout  
\*\* bouncycastle API

## Create First Admin User

Username



Password

.....

Confirm password

.....

Full name

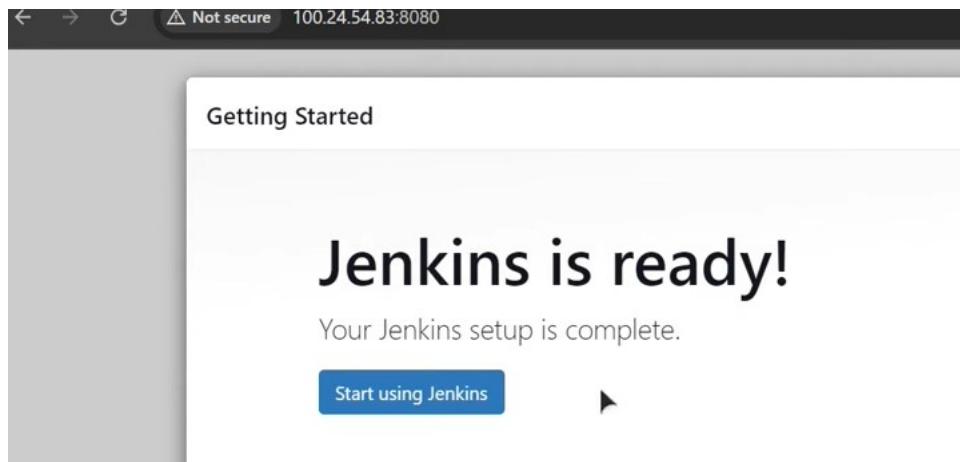
## Getting Started

# Instance Configuration

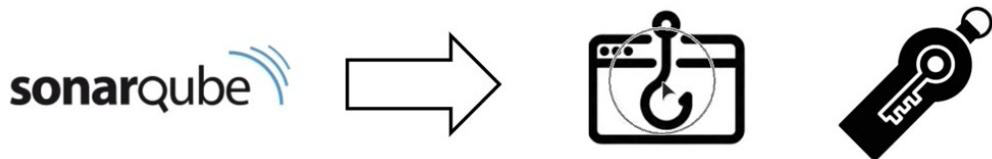
Jenkins URL:

The Jenkins URL is used to provide the root URL for absolute links to various Jenkins resources. That means this value is required for proper operation of many Jenkins features including email notifications, PR status updates, and the BUILD\_URL environment variable provided to build steps.

The proposed default value shown is **not saved yet** and is generated from the current request, if possible. The best practice is to set this value to the URL that users are expected to use. This will avoid confusion when sharing or viewing links.



# CONFIGURE SONARQUBE



# setting up Webhook - generating the Token

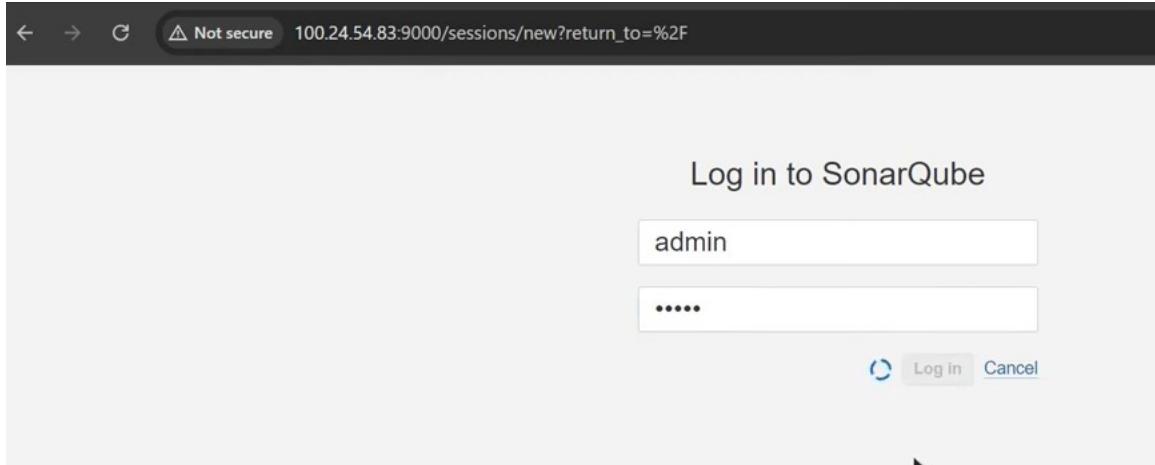
# SonarCube URL

```
aws_instance.my-ec2 (remote-exec): Executing: /usr/lib/systemd/systemd-sysv-install enable jenkins
aws_instance.my-ec2 (remote-exec): Access Jenkins Server here --> http://100.24.54.83:8080
aws_instance.my-ec2 (remote-exec): Jenkins Initial Password: 315c46806a9f4eca82f27a2239647548
aws_instance.my-ec2 (remote-exec): Access SonarQube Server here --> http://100.24.54.83:9000
aws_instance.my-ec2 (remote-exec): SonarQube Username & Password: admin
aws_instance.my-ec2 (remote-exec): Access Grafana Server here --> http://100.24.54.83:3000
aws_instance.my-ec2 (remote-exec): Access Prometheus Server here --> http://100.24.54.83:9090
aws_instance.my-ec2: Creation complete after 2m32s [id=i-0858872b9a500cf8e]

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

**Outputs:**

```
PRIVATE-IP = "172.31.88.75"
PUBLIC-IP = "100.24.54.83"
SERVER-SSH-ACCESS = "ubuntu@100.24.54.83"
PS>
```



# change the pswd :

## 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**

← → C ⚠ Not secure 100.24.54.83:9000/projects/create

sonarqube Projects Issues Rules Quality Profiles Quality Gates Administration ? Q Search for pr

How do you want to create your project?

Do you want to benefit from all of SonarQube's features (like repository import and Pull Request decoration)? Create your project from your favorite DevOps platform. First, you need to set up a DevOps platform configuration.

 From Azure DevOps <a href="#">Set up global configuration</a>	 From Bitbucket Server <a href="#">Set up global configuration</a>	 From Bitbucket Cloud <a href="#">Set up global configuration</a>	 From GitHub <a href="#">Set up global configuration</a>	 From GitLab <a href="#">Set up global configuration</a>
---	---	--	--	---

Are you just testing or have an advanced use-case? Create a project manually.

# WEBHOOK FOR SONARQUBE



Send Notifications to Jenkins – WebHook is used

# go to Administration

← → ⌂ △ Not secure 100.24.54.83:9000/admin/settings

sonarqube Projects Issues Rules Quality Profiles Quality Gates Administration

Administration

Configuration ▾ Security ▾ Projects ▾ System Marketplace

General Settings ►

Edit global settings for this SonarQube instance.

Find in Settings

Analysis Scope Duplications

Authentication

DevOps Platform Integrations

External Analyzers

General

Housekeeping

Cross project duplication detection

DEPRECATED - By default, SonarQube detects duplications at project level. This means that a block duplicated on two different projects won't be reported. Setting this parameter to "true" allows to detect duplicates across projects. Note that activating this property will significantly increase each SonarQube analysis time, and therefore badly impact the performances of report processing as more and more projects are getting involved in this cross project duplication mechanism.

(default)

Get it

This screenshot shows the SonarQube Administration interface. The top navigation bar includes links for sonarqube, Projects, Issues, Rules, Quality Profiles, Quality Gates, and Administration. The Administration tab is selected. Below the navigation is a breadcrumb trail: Configuration ▾ > General Settings. The main content area is titled 'General Settings' with a right-pointing arrow. A sub-section titled 'Cross project duplication detection' is shown, containing a note about deprecated behavior and a toggle switch that is currently off. A '(default)' label is next to the switch. At the bottom right of this section is a 'Get it' button with a SonarQube icon. On the left, there's a sidebar with links for Analysis Scope, Duplications, Authentication, DevOps Platform Integrations, External Analyzers, General, and Housekeeping. A search bar labeled 'Find in Settings' is located at the bottom of the sidebar.

Click on

← → ⌂ △ Not secure 100.24.54.83:9000/admin/settings

Configuration ▾ Security ▾ Projects ▾ System

General Settings

Encryption

Webhooks

Find in Settings

This screenshot shows the SonarQube Administration interface with the Configuration tab selected. The top navigation bar includes links for Configuration ▾ (which is active), Security ▾, Projects ▾, and System. Below the navigation is a breadcrumb trail: Configuration ▾ > General Settings. The main content area shows a dropdown menu for 'General Settings' which is currently open. The menu items are 'General Settings' (highlighted with a blue background), 'Encryption', and 'Webhooks'. At the bottom of the dropdown is a search bar labeled 'Find in Settings'. To the right of the dropdown, there is a note: 'Edit global settings for this SonarQube instance.' followed by a SonarQube icon.

Administration

Configuration ▾ Security ▾ Projects ▾ System Marketplace

### Webhooks

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](#).

No webhook defined.

## # Create

### Create Webhook

All fields marked with \* are required

**Name \***  

**URL \***  

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.

# add url of Jenkins ..

The screenshot shows the Jenkins dashboard at [100.24.54.83:8080](http://100.24.54.83:8080). The main content area displays the message "Welcome to Jenkins!" and "Start building your software project". On the left, there are links for "New Item", "Build History", "Manage Jenkins", and "My Views".

+ New Item

Build History

Manage Jenkins

My Views

## Welcome to Jenkins!

This page is where your Jenkins jobs will be displayed. To get started, you can : builds or start building a software project.

Start building your software project

### # Webhook Created

The screenshot shows the SonarQube administration interface at [100.24.54.83:9000/admin/webhooks](http://100.24.54.83:9000/admin/webhooks). The "Webhooks" section is displayed, showing a table with one entry: "sonarqube-webhook" with URL "http://100.24.54.83:8080/sonarqube-webhook". A "Create" button is visible in the top right corner.

### # Token

The screenshot shows the SonarQube administration interface at [100.24.54.83:9000/admin/webhooks](http://100.24.54.83:9000/admin/webhooks). The "Webhooks" section is displayed, showing a table with one entry: "sonarqube-webhook" with URL "http://100.24.54.83:8080/sonarqube-webhook". A tooltip is shown over the "Webhooks" link in the sidebar, explaining that 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. A link to the "Webhooks documentation" is provided.

Administration

Configuration ▾ Security ▾ Projects ▾ System Marketplace

Users Create and administer individual users.

Search by login or name...

	SCM Accounts	Last connection	Groups	Tokens
A Administrator admin		< 1 hour ago	sonar-administrators sonar-users	0

1 of 1 shown

Not secure 100.24.54.83:9000/admin/users

Accounts	Last connection	Groups	Tokens
	< 1 hour ago	sonar-administrators sonar-users	0

**Update Tokens**

# Token Generated - that will be saved in Jenkins - Credential Section (two way Communication Sonar <-> Jenkins)

Tokens of Administrator

Generate Tokens

Name	Expires in
Enter Token Name	30 days

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

squ\_1T81a5ee618c498c4fbe4d8394238d4af00d6a0

Name	Type	Project	Last use	Created	Expiration
sonar-token	User		Never	October 20, 2024	November 19, 2024

# cp - sonar token .. Go to Jenkins -manage jenkins – credentials

The screenshot shows the Jenkins Manage Jenkins interface at the URL [100.24.54.83:8080/manage/](http://100.24.54.83:8080/manage/). The left sidebar includes links for New Item, Build History, Manage Jenkins (which is selected), and My Views. The main content area is titled "Manage Jenkins" and contains a message about building on the built-in node. It features a "System Configuration" section with six items: System, Tools, Plugins, Nodes, Clouds, and Appearance. Each item has a brief description and a corresponding icon. A search bar at the top right says "Search (CTRL+K)".

## # Credentials

The screenshot shows the Jenkins Credentials page at the URL [100.24.54.83:8080/manage/credentials](http://100.24.54.83:8080/manage/credentials). The left sidebar shows the user is in the "Manage Jenkins > Credentials" section. The main content area is titled "Credentials" and displays a table with columns: T, P, Store ↓, Domain, ID, and Name. Below the table, a section titled "Stores scoped to Jenkins" lists "System" and "(global)" under the "Domains" column. A toolbar at the bottom allows filtering by icon size: S, M, and L.

The screenshot shows the Jenkins Global credentials (unrestricted) page. The URL is 100.24.54.83:8080/manage/credentials/store/system/domain/\_/. The page title is "Global credentials (unrestricted)". The content area displays a message: "This credential domain is empty. How about [adding some credentials?](#)". Below this message, there is a table header with columns: ID, Name, Kind, and Description. There are also icons for S, M, and L.

## Global credentials (unrestricted)

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

ID	Name	Kind	Description
This credential domain is empty. How about <a href="#">adding some credentials?</a>			

Icon: S M L

### # Add Credentials

The screenshot shows the Jenkins # Add Credentials form for a Secret text credential. The Kind is set to "Secret text". The Scope is "Global (Jenkins, nodes, items, all child items, etc)". The Secret field contains a redacted value. The ID is "sonar-token" and the Description is "sonar-token". A "Create" button is at the bottom.

### # I need to Add both AWS Access key + secrete Key to run AWS ECR + EKS Cmds from Jenkins

```
# ACCESS key:  
AKIAUQCLMSLD5NUMX5FZI  
  
# Secret key:  
No2KBJb8bKUKLzottBzlcaG4Huh+rVHy+MOg8aMU
```

Not secure 100.24.54.83:8080/manage/credentials/store/system/domain/\_/

**Kind**

Secret text

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

**Secret**  
.....

**ID** ?  
access-key

**Description** ?  
access-key

**Create**

## 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	
access-key	access-key	Secret text	access-key	

Icon: S M L

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	
access-key	access-key	Secret text	access-key	
secret-key	secret-key	Secret text	secret-key	

Icon: S M L

# Configure Jenkins

# CONFIGURE JENKINS



# Credentials part is done .. heading to other 3 .. sections

A screenshot of the Jenkins 'Manage Jenkins' page. The top navigation bar shows 'Jenkins' and 'Dashboard &gt; Manage Jenkins'. The main content area is titled 'Manage Jenkins' and features a 'System Configuration' sidebar. The 'System' option in the sidebar is highlighted with a mouse cursor. Other options include 'Tools', 'Plugins', 'Nodes', 'Clouds', and 'Appearance'. The 'Security' section is also visible at the bottom.

# Click on Plugins

Dashboard > Manage Jenkins > Plugins

## Plugins

- Updates
- Available plugins**
- Installed plugins
- Advanced settings
- Download progress

Search: eclipse temu

Install	Name	Released
<input checked="" type="checkbox"/>	SonarQube Scanner 2.17.2 External Site/Tool Integrations Build Reports	8 mo 3 days ago
<input checked="" type="checkbox"/>	NodeJS 1.6.2 npm	2 mo 7 days ago
<input checked="" type="checkbox"/>	Pipeline: Stage View 2.34 User Interface	11 mo ago
<input checked="" type="checkbox"/>	Eclipse Temurin installer 1.5 Provides an installer for the JDK tool that downloads the JDK from https://adoptium.net	2 yr 0 mo ago

Install

## # last Docker

Dashboard > Manage Jenkins > Plugins

## Plugins

- Updates
- Available plugins**
- Installed plugins
- Advanced settings
- Download progress

Search: docker

Install	Name	Released
<input type="checkbox"/>	User Interface Pipeline Stage View Plugin.	11 mo ago
<input checked="" type="checkbox"/>	Eclipse Temurin installer 1.5 Provides an installer for the JDK tool that downloads the JDK from https://adoptium.net	2 yr 0 mo ago
<input checked="" type="checkbox"/>	Docker 1.7.0 Cloud Providers Cluster Management docker	5 days 3 hr ago
<input checked="" type="checkbox"/>	Docker Commons 443.v921729d5611d Library plugins (for use by other plugins) docker	2 mo 14 days ago
<input checked="" type="checkbox"/>	Docker Pipeline 580.vc0c340686b_54 pipeline DevOps Deployment docker	5 mo 1 day ago

Install

Search: docker

Install	Name	Released
<input type="checkbox"/>	pipeline DevOps Deployment docker Build and use Docker containers from pipelines.	5 mo 1 day ago
<input checked="" type="checkbox"/>	Docker API 3.4.0-94.v65ced49b_a_7d5 Library plugins (for use by other plugins) docker	25 days ago

This plugin is up for adoption! We are looking for new maintainers. Visit our [Adopt a Plugin](#) initiative for more information.

## #Prometheus-metric

Dashboard > Manage Jenkins > Plugins

**Plugins**

- Updates
- Available plugins**
- Installed plugins
- Advanced settings
- Download progress

prometheus

Build Tools docker

This plugin allows to add various docker commands to your job as build steps.

Warning: This plugin version may not be safe to use. Please review the following security notices:  
• [CSRF vulnerability and missing permission check](#)

4 mo 25 days ago

Prometheus metrics 787.v52e8f47488fc

monitoring Miscellaneous

Jenkins Prometheus Plugin expose an endpoint (default /prometheus) with metrics where a Prometheus Server can scrape.

27 days ago

Loading plugin extensions



→ [Go back to the top page](#)

(you can start using the installed plugins right away)

→ [Restart Jenkins when installation is complete and no jobs are running](#)

← → ⌛ ⚠ Not secure 100.24.54.83:8080/manage/pluginManager/updates/



Please wait while Jenkins is restarting ...

Your browser will reload automatically when Jenkins is ready.

Safe Restart

Builds on agents can usually continue.

# 3<sup>rd</sup> Part- System

# CONFIGURE JENKINS



# in System .. setting up SonarQube in jenkins Server..

# Manage Jenkins - Click System – to setup sonarQube server

A screenshot of a web browser displaying the Jenkins 'Manage Jenkins' page at the URL 100.24.54.83:8080/manage/. The page has a header bar with back, forward, and search icons. The main content area shows a sidebar with 'Build History', 'Manage Jenkins' (which is currently selected), and 'My Views'. Below the sidebar is a 'Build Queue' section stating 'No builds in the queue.' On the right side, there's a 'System Configuration' section with several options: 'System' (Configure global settings and paths), 'Tools' (Configure tools, their locations and automatic installers), 'Plugins' (Add, remove, disable or enable plugins that can extend the functionality of Jenkins), 'Nodes' (Add, remove, control and monitor the various nodes that Jenkins runs jobs on), 'Clouds' (Add, remove, and configure cloud instances to provision agents on-demand), and 'Appearance' (Configure the look and feel of Jenkins). A red callout box highlights the 'System' option.

# integrate Jenkins with SonarQube



# Jenkins

Dashboard > Manage Jenkins > System >

Search (CTRL+K)

## System

### Home directory ?

By default, Jenkins stores all of its data in this directory on the file system

/var/lib/jenkins (edit)

### System Message

This message will be displayed at the top of the Jenkins Main page. This can be useful for posting notifications to your users

Plain text [Preview](#)

### Maven Project Configuration

#### Global MAVEN\_OPTS ?

→ C Not secure 100.24.54.83:8080/manage/configure

## SonarQube servers

If checked, job administrators will be able to inject a SonarQul



Environment variables

### SonarQube installations ▶

List of SonarQube installations

Add SonarQube

## SonarQube installations

List of SonarQube installations

Name

Server URL

Default is http://localhost:9000

# choose Sonar Token

### Server authentication token

SonarQube authentication token. Mandatory when anonymous access is disabled.

- none -

+ Add

- none -  
access-key  
secret-key  
sonar-token

Advanced ▾

# Save

# 4<sup>th</sup> Part Configure the tools

# CONFIGURE JENKINS



JDK --> 17.0.8

SONAR-SCANNER --> latest

NodeJS --> 16.20.0

Docker --> latest

## # Click on tools in manage Jenkins

The screenshot shows the Jenkins 'Manage Jenkins' interface. In the top navigation bar, 'Manage Jenkins' is highlighted. Below it, there's a warning message about a docker-build-step plugin vulnerability. The main area is titled 'System Configuration' and contains several sections: 'System' (Configure global settings and paths), 'Tools' (Configure tools, their locations and automatic installers), 'Nodes' (Add, remove, control and monitor the various nodes that Jenkins runs jobs on), 'Clouds' (Add, remove, and configure cloud instances to provision agents on-demand), 'Managed files' (e.g. settings.xml for maven, central managed scripts, custom files, ...), and 'Plugins' (Add, remove, disable or enable plugins that can extend the functionality of Jenkins). On the left, there are dropdown menus for 'Build History', 'My Views', 'Build Queue' (which shows 'No builds in the queue'), and 'Build Executor Status' (which shows '1 Idle' and '2 Idle').

The screenshot shows the Jenkins 'Tools' configuration page. At the top, the URL is 100.24.54.83:8080/manage/configureTools/. The page title is 'Tools'. It includes sections for 'Maven Configuration' (Default settings provider: 'Use default maven settings') and 'JDK installations' (button to 'Add JDK').

Add JDK

≡ **JDK**

Name

! Required

JDK

**Install automatically** ?

Add Installer ^

Filter

Extract \*.zip/\*.tar.gz

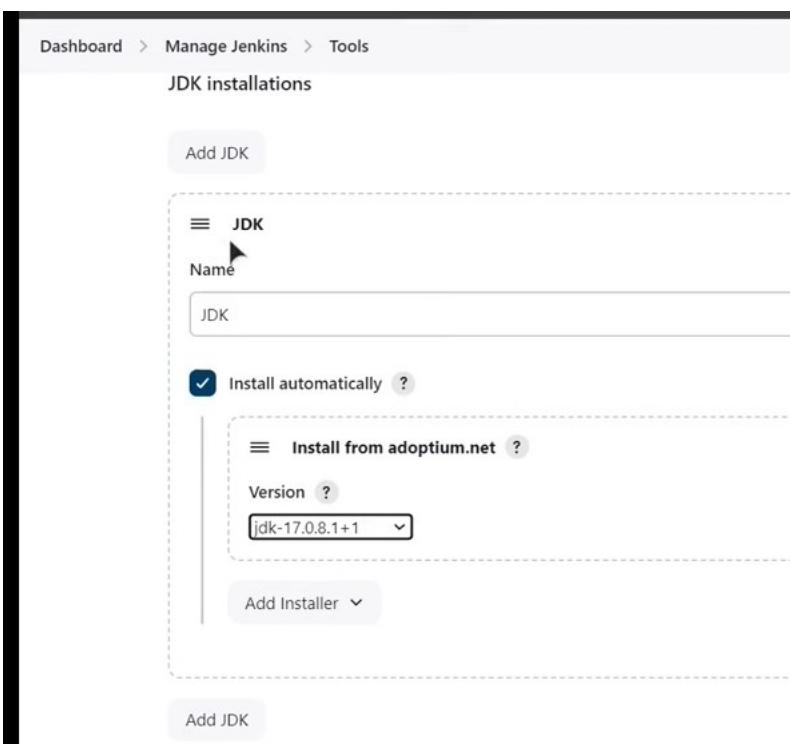
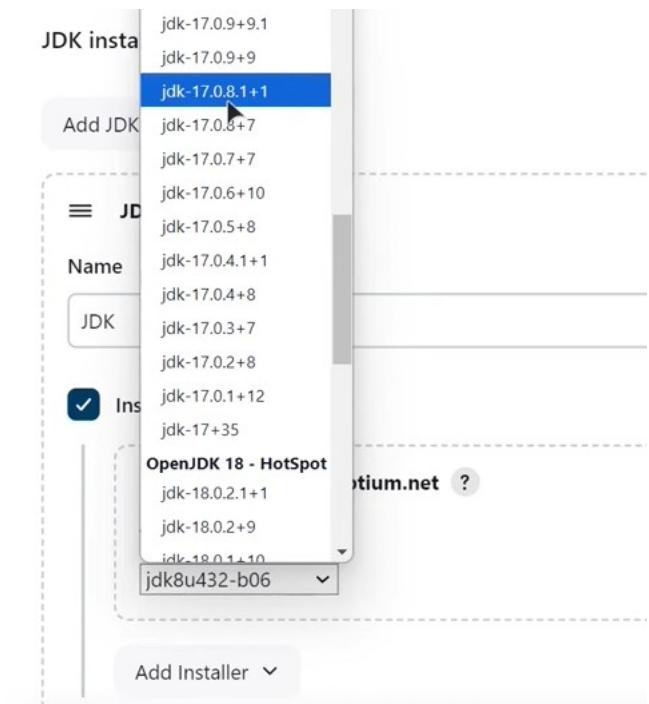
Add J

Install from [adoptium.net](#)

Run Batch Command

Run Shell Command

...



## Add SonarQube Scanner

### SonarQube Scanner

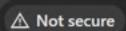
Name

 Required

Install automatically 

#### Install from Maven Central

Version

 Not secure 100.24.54.83:8080/manage/configureTools/

### NodeJS

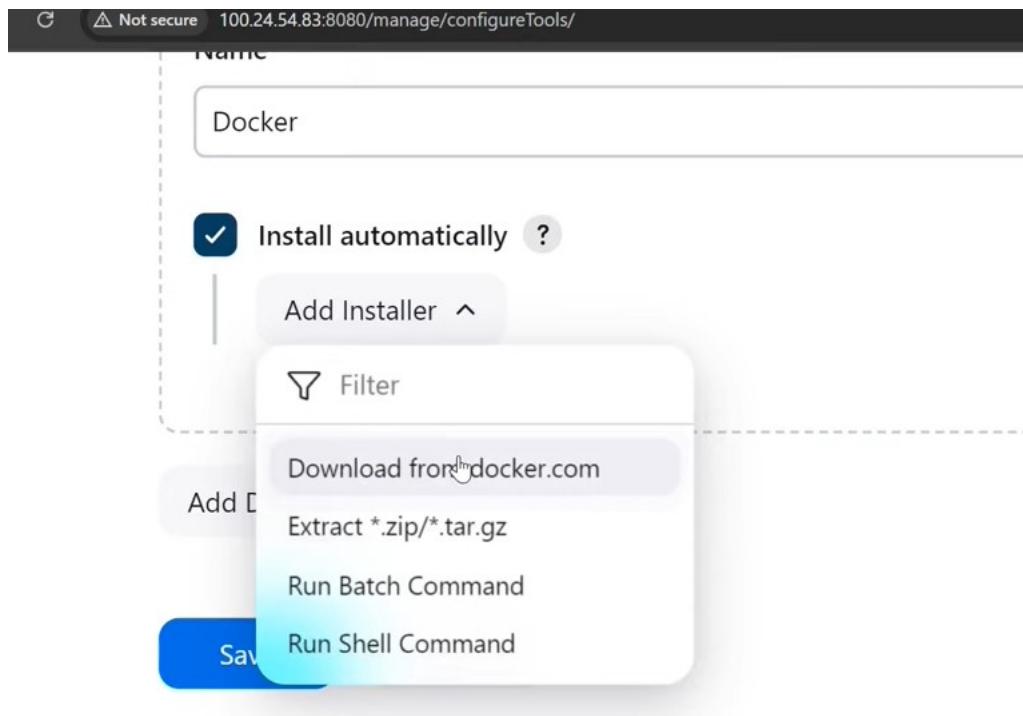
Name

Install automatically 

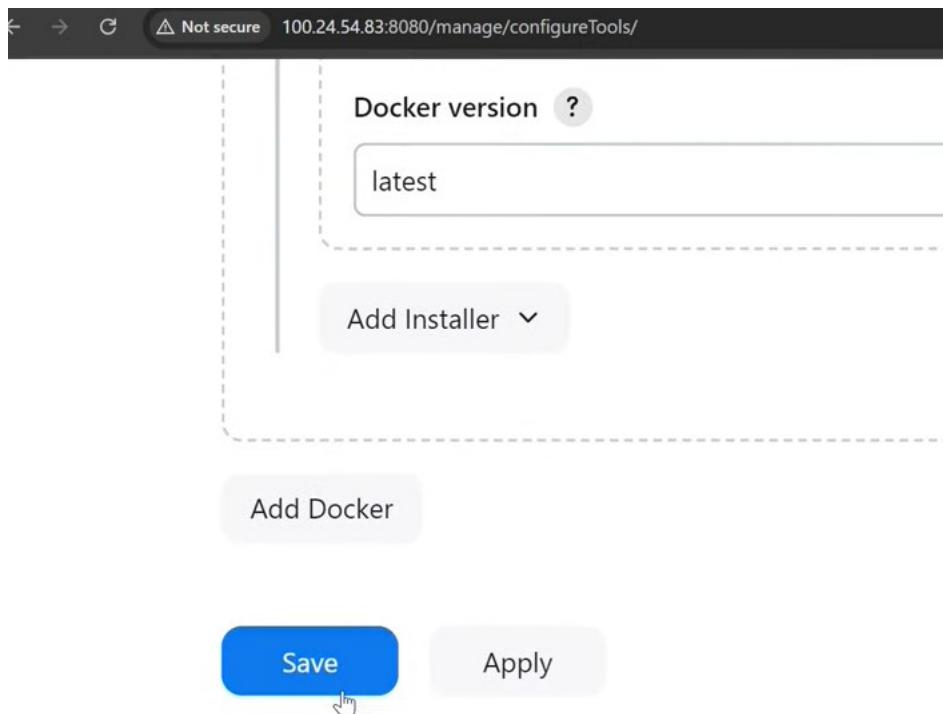
#### Install from nodejs.org

Version

For the underlying architecture, if available, force the installation



# save



# Build Pipeline..

# BUILD PIPELINE OVERVIEW



## Pipeline Overview

### Pipeline Stages

1. **Git Checkout:** Clones the source code from GitHub.
2. **SonarQube Analysis:** Performs static code analysis.
3. **Quality Gate:** Ensures code quality standards.
4. **Install NPM Dependencies:** Installs NodeJS packages.
5. **Trivy Security Scan:** Scans the project for vulnerabilities.
6. **Docker Build:** Builds a Docker image for the project.
7. **Push to AWS ECR:** Tags and pushes the Docker image to ECR.
8. **Image Cleanup:** Deletes images from the Jenkins server to save space.

```
# Create – pipeline – Test it sample one
```

Dashboard > All > New Item

## New Item

Enter an item name

Select an item type

**build-pipeline**

 **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,

### # Generate the script- sample

← → C △ Not secure 100.24.54.83:8080/job/build-pipeline/configure

Dashboard > build-pipeline > Configuration

## Configure

General

Advanced Project Options **Pipeline**

### Pipeline

Definition

Pipeline script

Script ?

```
1 pipeline {  
2     agent any  
3     stages {  
4         stage('Hello') {  
5             steps {  
6                 echo 'Hello World'  
7             }  
8         }  
9     }  
10 }  
11 }  
12 }
```

Hello World

Use Groovy Sandbox ?

 Jenkins

Dashboard > build-pipeline >

[Status](#) **build-pipeline**

</> Changes

▷ Build Now

⚙ Configure

trash Delete Pipeline

🔍 Full Stage View

⠇ Stages

✍ Rename

ⓘ Pipeline Syntax

 Build History trend



 Jenkins

Dashboard > build-pipeline >

[Status](#) **build-pipeline**

</> Changes Build scheduled

▷ Build Now

⚙ Configure

trash Delete Pipeline

🔍 Full Stage View

⠇ Stages

✍ Rename

ⓘ Pipeline Syntax

### Stage View

No data available. This Pipeline has not yet run.

### Permalinks

▶

Jenkins

Dashboard > build-pipeline >

Status

</> Changes

▷ Build Now

⚙ Configure

trash Delete Pipeline

🔍 Full Stage View

⠇ Stages

✍ Rename

❓ Pipeline Syntax

## build-pipeline

### Stage View



### Permalinks

Build History

trend ▾

# Logs – o/p

← → ⌂ Not secure 100.24.54.83:8080/job/build-pipeline/

### Stage Logs (Hello)

Print Message -- Hello World (self time 17ms)

```
Hello World
```

</> Changes

▷ Build Now

⚙ Configure

trash Delete Pipeline

🔍 Full Stage View

⠇ Stages

✍ Rename

❓ Pipeline Syntax

### Stage View

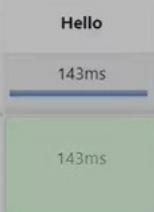
Average stage times:  
(Average full run time: ~2s)

#1 Oct 20 07:58 No Changes

Hello

143ms

143ms



# Go back to Configure

Jenkins

Dashboard > build-pipeline > Configuration

Configure General

Description

Plain text [Preview](#)

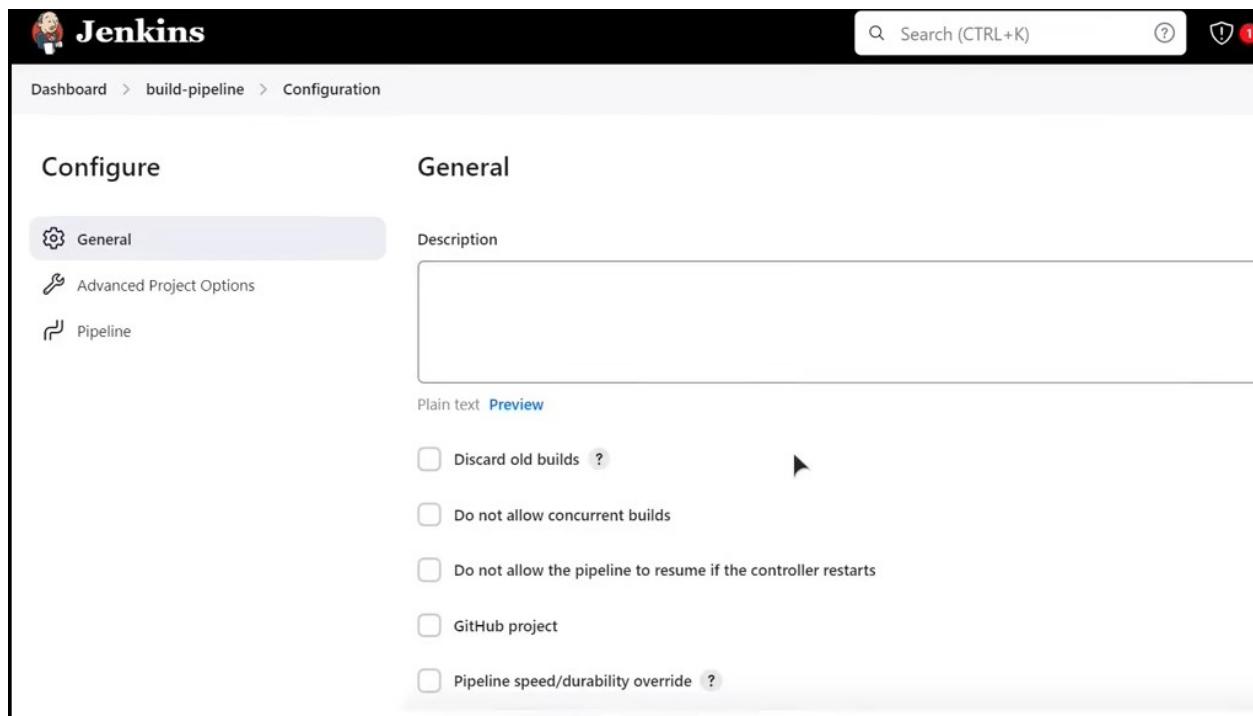
Discard old builds [?](#)

Do not allow concurrent builds

Do not allow the pipeline to resume if the controller restarts

GitHub project

Pipeline speed/durability override [?](#)



This screenshot shows the Jenkins Pipeline configuration screen. It has two main sections: 'Configure' on the left and 'General' on the right. Under 'General', there's a 'Description' field which is currently empty. Below it are several checkboxes for pipeline settings: 'Discard old builds', 'Do not allow concurrent builds', 'Do not allow the pipeline to resume if the controller restarts', 'GitHub project', and 'Pipeline speed/durability override'. The 'GitHub project' checkbox is checked.

# 1<sup>st</sup> stage .. Git checkout



# Click on Pipeline Syntax

← → G Not secure 100.24.54.83:8080/job/build-pipeline/configure

Dashboard > build-pipeline > Configuration

## Configure

Definition

Pipeline script

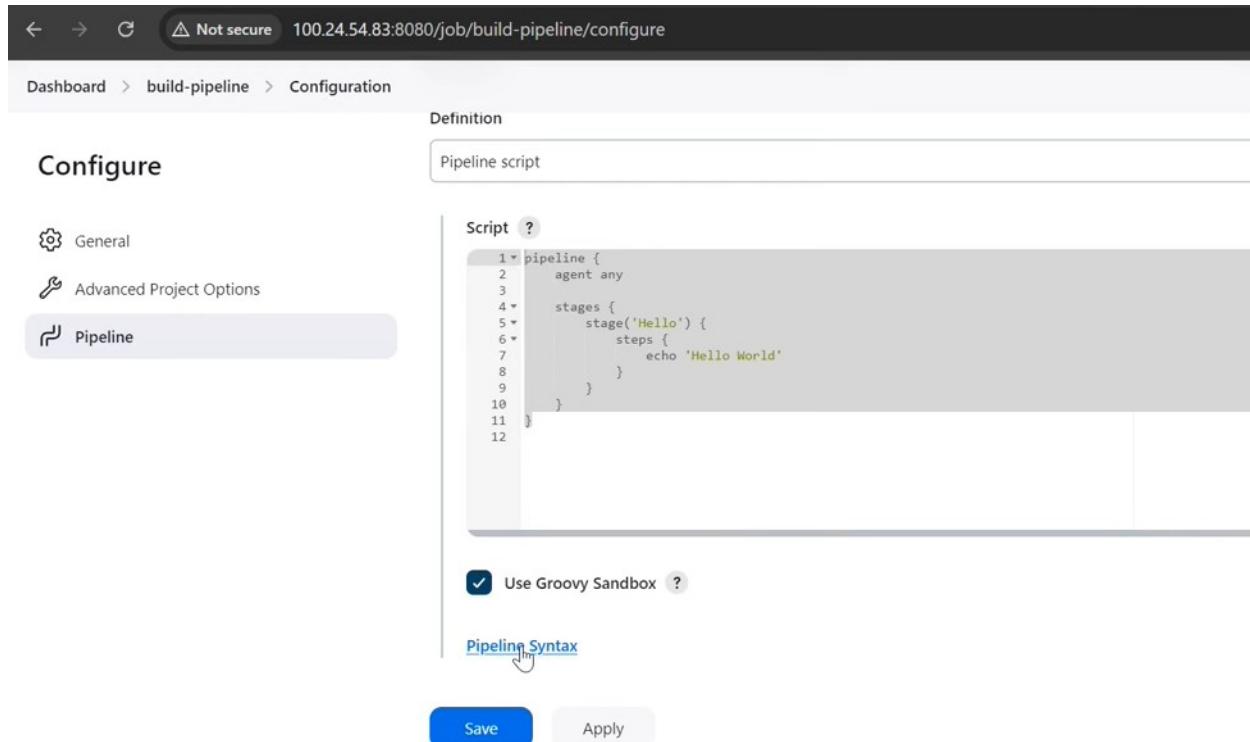
Script ?

```
1 pipeline {  
2     agent any  
3  
4     stages {  
5         stage('Hello') {  
6             steps {  
7                 echo 'Hello World'  
8             }  
9         }  
10    }  
11}  
12
```

Use Groovy Sandbox ?

[Pipeline Syntax](#)

Save Apply



← → G Not secure 100.24.54.83:8080/job/build-pipeline/pipeline-syntax/

Dashboard > build-pipeline > Pipeline Syntax

(?) IntelliJ IDEA GDSL

git ?

Repository URL ?

```
https://github.com/Siddhartha082/AWS-DevOps-Final-Chapter-Project
```

Branch ?

```
main
```

Credentials ?

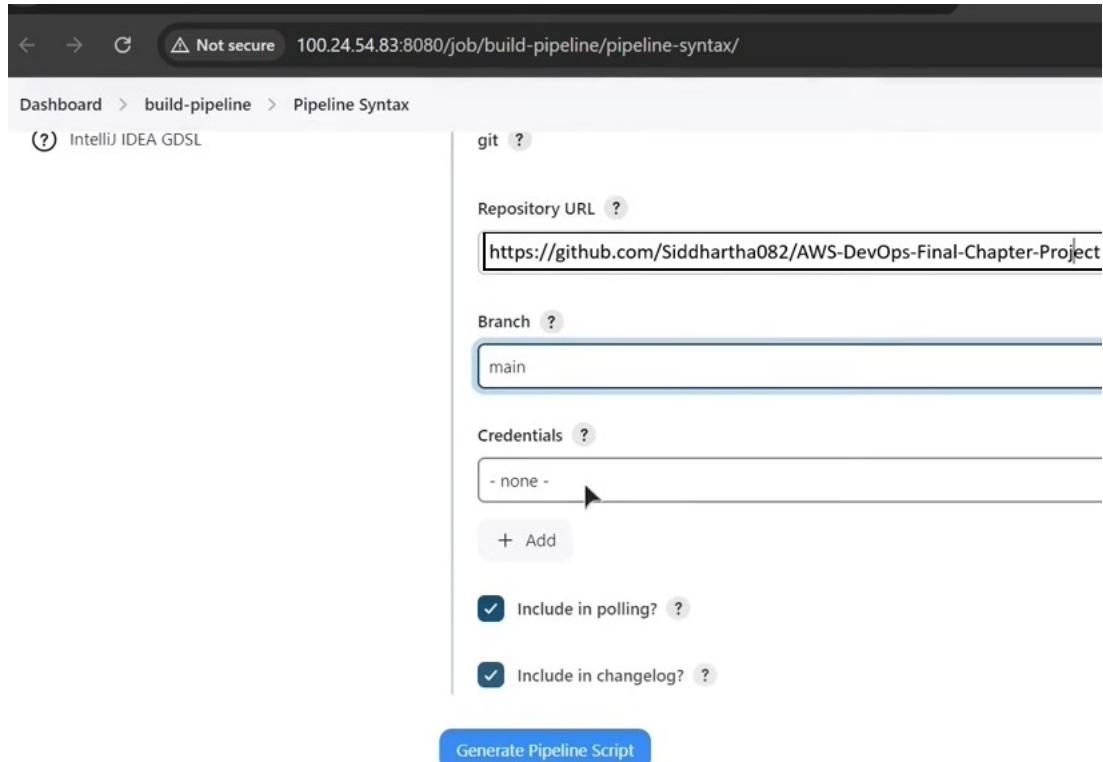
```
- none -
```

+ Add

Include in polling? ?

Include in changelog? ?

Generate Pipeline Script



# note no Credentials added as my repo is in Public domain

Include in polling? ?

Include in changelog? ?

Generate Pipeline Script

```
git branch: 'main', url: https://github.com/Siddhartha082/AWS-DevOps-Final-Chapter-Project
```

```
environment {
    SCANNER_HOME = tool 'SonarQube Scanner'
}

stages {
    stage('1. Git Checkout') {
        steps [
            git branch: 'main', url: 'https://github.com/Siddhartha082/AWS-DevOps-Final-Chapter-Project.git'
        ]
    }
}
```

# 2<sup>nd</sup> Stage- Static code analysis Test- SCAT – SonarQube



```
stage('2. SonarQube Analysis') {
    steps [
        withSonarQubeEnv ('sonar-server') {
            sh """
                $SCANNER_HOME/bin/sonar-scanner \
                -Dsonar.projectName=amazon-prime \
                -Dsonar.projectKey=amazon-prime
            """
        }
    ]
}
```

← → ⌂ Not secure 100.24.54.83:8080/job/build-pipeline/pipeline-syntax/

**Steps**

Roles Reference

mentionation

ference

GDSL

**Sample Step**

```
withSonarQubeEnv: Prepare SonarQube Scanner environment
```

**withSonarQubeEnv ?**

**Server authentication token**

SonarQube authentication token. Mandatory when anonymous access is disabled. Will default to the one defined in the SonarQube configuration.

- none -

+ Add

**Generate Pipeline Script**

```
withSonarQubeEnv
  // some block
```

## # go to Jenkins – tools

Dashboard > Manage Jenkins

**Manage Jenkins**

Warnings have been published for the following currently installed components: [Go to plugin manager](#) [Configure which of these warnings are shown](#)

**docker-build-step 2.12:** [CSRF vulnerability and missing permission check](#) (no fix available)  
No fixes for these issues are available. It is recommended that you review the security advisory and apply mitigations if possible, or uninstall this plugin.

**Build Queue**  
No builds in the queue.

**Build Executor Status**  
1 Idle  
2 Idle

**System Configuration**

- System** Configure global settings and paths.
- Nodes** Add, remove, control and monitor the various nodes that Jenkins runs
- Tools** Configure tools, their locations and automatic installers.
- Clouds** Add, remove, and configure cloud instances to provision agents on-
- Plugins** Add, remove, disable or enable plugins that can extend the functionality of Jenkins.
- Appearance** Configure the look and feel of Jenkins

## # env - var

## Add SonarQube Scanner

### SonarQube Scanner

#### Name

SonarQube Scanner



Install automatically ?

### Install from Maven Central

#### Version

## # Env Variable Scanner

```
environment {
    SCANNER_HOME = tool 'SonarQube Scanner'
}

stages {
    stage('1. Git Checkout') {
        steps {
            git branch: 'main', url: 'https://github.com/Siddhartha082/AWS-DevOps-Final-Chapter-Project.git'
        }
    }

    stage('2. SonarQube Analysis') {
        steps {
            withSonarQubeEnv ('sonar-server') {
                sh """
                    $SCANNER_HOME/bin/sonar-scanner \
                    -Dsonar.projectName=amazon-prime \
                    -Dsonar.projectKey=amazon-prime
                """
            }
        }
    }
}
```

# Sonar helps in checking duplication/ Bugs in code / check security -vulnerabilities - Sonar Scanner

#3<sup>rd</sup> Stage Quality Gate - Definition & example

```
stage('3. Quality Gate') {
    steps {
        waitForQualityGate abortPipeline: false,
        credentialsId: 'sonar-token'
    }
}
```

The screenshot shows a web-based interface for defining a Jenkins pipeline. At the top, a URL bar indicates the address is 100.24.54.83:8080/job/build-pipeline/pipeline-syntax/. The main area is titled "Steps" and contains a "Sample Step" section. Inside the "Sample Step" section, there is a dropdown menu set to "waitForQualityGate: Wait for SonarQube analysis to be completed and return quality gate status". Below this, under "waitForQualityGate", there is a "Server authentication token" field containing "sonar-token". A warning message "⚠ Cannot find any credentials with id sonar-token" is displayed next to the field. A "Generate Pipeline Script" button is located below the step configuration. The generated Jenkinsfile code is shown in a large text area at the bottom:

```
withSonarQubeEnv (
    // some block
)
```

A second warning message "⚠ Cannot find any credentials with id sonar-token" is also present in this area. A "Generate Pipeline Script" button is located below the Jenkinsfile code.

```
waitForQualityGate abortPipeline: false, credentialsId: 'sonar-token'
```

# The Quality Gate

The Quality Gate lets you set your own code quality and security conditions by selecting a metric and then setting the pass/fail threshold. If any of the conditions in the QG fail, the overall QG fails and you know not to merge your code until you remedy the situation. The QG is dynamically updated so you'll know immediately if a fix gives you the 'GREEN' light! Just like with the QP, you can use the built-in Quality Gate called Sonar way or customize your own based on your team's clean, secure coding definition that we talked about earlier. An example will demonstrate how it all comes together. The graphic below shows you how the Reliability (bugs) Rating metric is calculated.

Sonar way BUILT-IN

**Conditions** ⓘ

**Conditions on New Code**

Conditions on New Code apply to all branches and to Pull Requests.

Metric	Operator	Value
Coverage	is less than	80.0%
Duplicated Lines (%)	is greater than	3.0%
Maintainability Rating	is worse than	A
Reliability Rating	is worse than	A
Security Hotspots Reviewed	is less than	100%
Security Rating	is worse than	A

**Reliability Rating (reliability\_rating)**

A = 0 Bugs

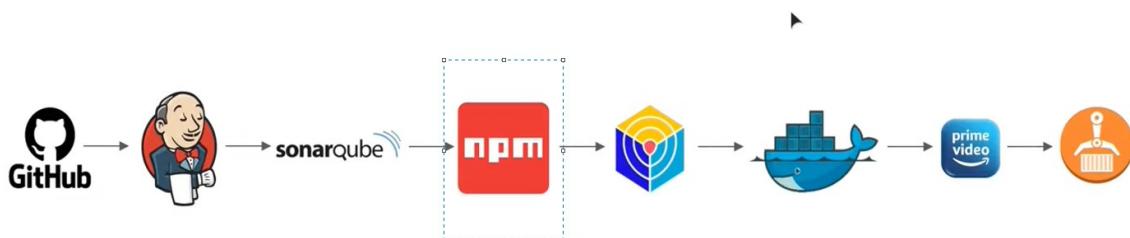
B = at least 1 Minor Bug

C = at least 1 Major Bug

D = at least 1 Critical Bug

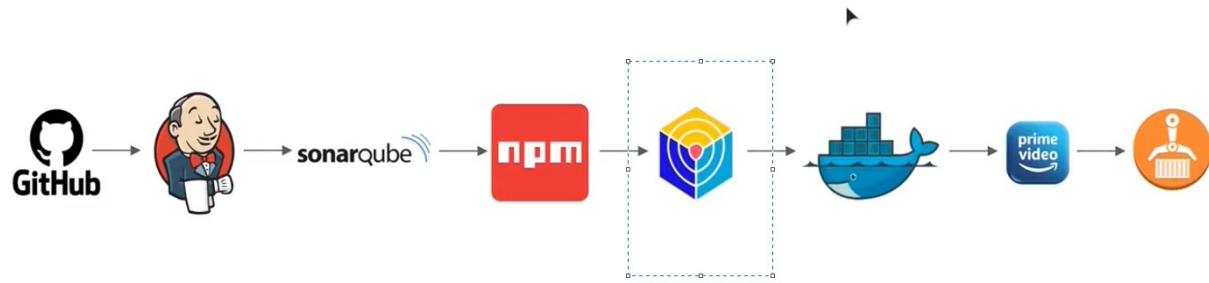
E = at least 1 Blocker Bug

## #4<sup>th</sup> Stage npm installation



```
stage('4. Install npm') {
  steps {
    | sh "npm install"
  }
}
```

## #5<sup>th</sup> Stage Trivy - Scans the project for vulnerabilities.



```
stage('5. Trivy Scan') {  
    steps {  
        | sh "trivy fs . > trivy.txt"  
    }  
}
```

## # 6<sup>th</sup> Stage Docker file



```
1 # Use Node.js Alpine base image  
2 FROM node:alpine  
3  
4 # Create and set the working directory inside the container  
5 WORKDIR /app  
6  
7 # Copy package.json and package-lock.json to the working directory  
8 COPY package.json package-lock.json /app/  
9  
10 # Install dependencies  
11 RUN npm install  
12  
13 # Copy the entire codebase to the working directory  
14 COPY . /app/  
15  
16 # Expose the port your container app  
17 EXPOSE 3000  
18  
19 # Define the command to start your application (replace "start" with the actual command to start your app)  
20 CMD ["npm", "start"]  
21
```

```

stage('6. Build Docker Image') {
    steps {
        sh "docker build -t ${params.ECR_REPO_NAME} ."
    }
}

parameters {
    string(name: 'ECR_REPO_NAME', defaultValue: 'amazon-prime', description: 'Enter repository name')
    string(name: 'AWS_ACCOUNT_ID', defaultValue: '851725192411', description: 'Enter AWS Account ID') // Added missing quote
}

```

### # 7<sup>th</sup> stage – ECR Creation



# Ref - <https://docs.aws.amazon.com/cli/latest/reference/ecr/create-repository.html>

AWS CLI Command Reference Home User Guide Forum GitHub Star 15,567

Creates a repository. For more information, see [Amazon ECR repositories in the Amazon Elastic Container Registry User Guide](#).

See also: [AWS API Documentation](#)

**Synopsis**

```

create-repository
[--registry-id <value>]
--repository-name <value>
[--tags <value>]
[--image-tag-mutability <value>]
[--image-scanning-configuration <value>]
[--encryption-configuration <value>]
[--cli-input-json <value>]
[--generate-cli-skeleton <value>]
[--debug]
[--endpoint-url <value>]
[--no-verify-ssl]
[--no-paginate]
[--output <value>]
[--query <value>]
[--profile <value>]
[--region <value>]
[--version <value>]
[--color <value>]
[--no-sign-request]
[--ca-bundle <value>]
[--cli-read-timeout <value>]
[--cli-connect-timeout <value>]

```

**Options**

Type here to search 26°C Mostly cloudy 14:53 15-11-2024

JENKINS SERVER → AWS ACCOUNT

aws configure  
access secret  
secret key

# go to Pipeline script --set up a Connection ... using credentials as key to connections

The screenshot shows the Jenkins Pipeline Syntax configuration page. The URL is 100.24.54.83:8080/job/build-pipeline/pipeline-syntax/. The page displays a sample step:

```
Sample Step
withCredentials: Bind credentials to variables
```

Below this, there is a 'Bindings' section with an 'Add' button. A dropdown menu is open, showing various credential types. The 'Docker client certificate' option is highlighted with a cursor icon.

Secret values are masked on a best-effort basis to prevent accidental disclosure. Multiline secrets, such as the contents of a SSH private key file, are not masked. See the inline help for details and usage guidelines.

Bindings

Add ^

Filter

- Certificate
- Docker client certificate
- Git Username and Password
- SSH User Private Key
- Secret ZIP file
- Secret file
- Secret text
- Username and password (conjoined)

**Secret text** ?

Variable ?  
AWS\_ACCESS\_KEY

Credentials ?  
access-key

+ Add

Add ▾

Generate Pipeline Script

```
waitForQualityGate abortPipeline: false, credentialsId: 'sonar-token'
```

**Secret text** ? ×

Variable ?  
AWS\_SECRET\_KEY

Credentials ? ▶  
secret-key

+ Add

Add ▾

Add ▾

Generate Pipeline Script

```
withCredentials([string(credentialsId: 'access-key', variable: 'AWS_ACCESS_KEY'), string(credentialsId: 'secret-key', variable: 'AWS_SECRET_KEY')) {  
    // some block  
}
```

Global Variables

# AWS Configure -

<https://docs.aws.amazon.com/cli/latest/reference/configure/set.html>

The screenshot shows a browser window with four tabs open: 'create-repository — AWS CLI 1', 'Siddhartha082/AWS-DevOps-F', 'AWS DevOps CI-CD Amazon', and 'set — AWS CLI 1.36.3 Command'. The current tab is 'set — AWS CLI 1.36.3 Command'. The URL in the address bar is 'docs.aws.amazon.com/cli/latest/reference/configure/set.html'. The page content includes a 'Examples' section with code snippets and a resulting configuration file.

Given an empty config file, the following commands:

```
$ aws configure set aws_access_key_id default_access_key
$ aws configure set aws_secret_access_key default_secret_key
$ aws configure set default.region us-west-2
$ aws configure set default.ca_bundle /path/to/ca-bundle.pem
$ aws configure set region us-west-1 --profile testing
$ aws configure set profile.testing.region eu-west-1
$ aws configure set preview.cloudsearch true
```

will produce the following config file:

```
[default]
region = us-west-2
ca_bundle = /path/to/ca-bundle.pem

[profile testing]
region = us-west-1
```

A screenshot of a Windows taskbar. The taskbar includes icons for File Explorer, Task View, Start, Taskbar settings, and a search bar. A PowerShell window is open, displaying the same configuration command as the previous screenshot.

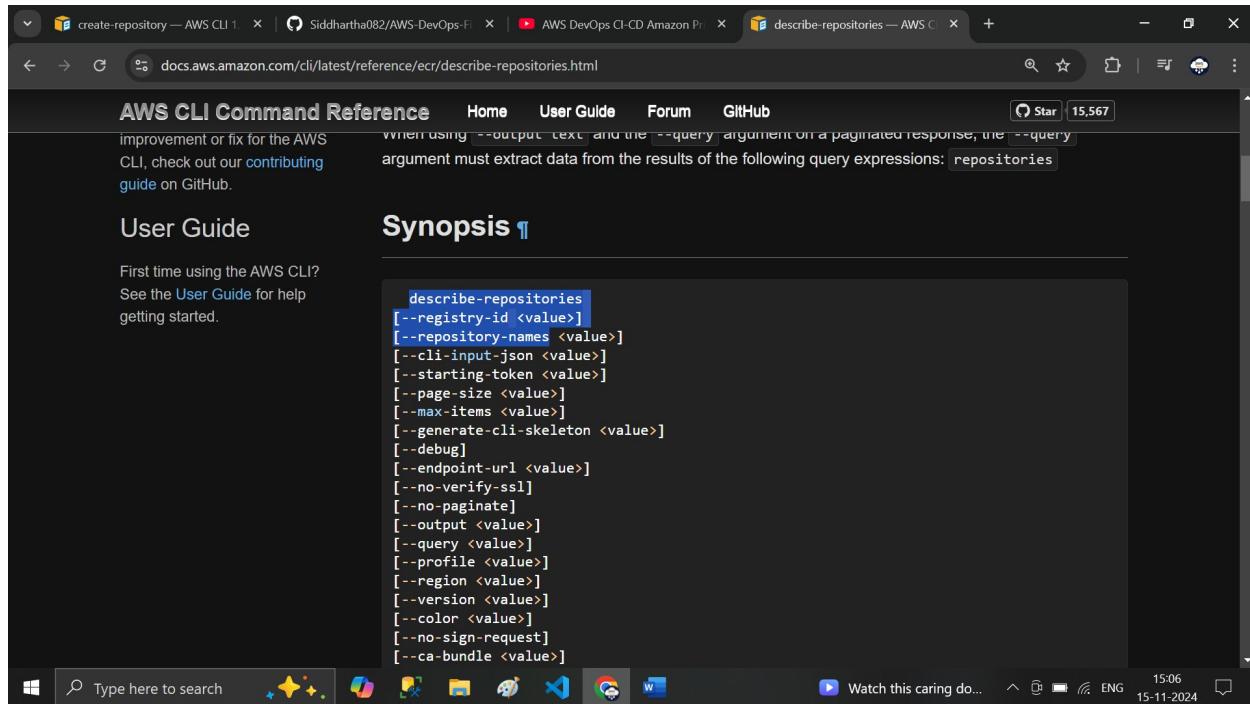
```
stage('7. Create ECR repo') {
    steps {
        withCredentials([string(credentialsId: 'access-key', variable: 'AWS_ACCESS_KEY'),
                        |           |           | string(credentialsId: 'secret-key', variable: 'AWS_SECRET_KEY')]) {
            sh """
                aws configure set aws_access_key_id $AWS_ACCESS_KEY
                aws configure set aws_secret_access_key $AWS_SECRET_KEY
                aws ecr describe-repositories --repository-names ${params.ER_REPO_NAME} --region us-east-1 || \
                aws ecr create-repository --repository-name ${params.ER_REPO_NAME} --region us-east-1
            """
        }
    }
}
```

# AWS ECR describe repo # if the ECR repo already existed then new repo will not be created

Else

4<sup>th</sup> Line of the code .. new ECR repo will be created ..... @ us-east-1 region

<https://docs.aws.amazon.com/cli/latest/reference/ecr/describe-repositories.html>



## # 8<sup>th</sup> Stage – Login into ECR & Tagging the image(Docker)

### ■ Go to AWS Console- ECR -create - ECR—

The screenshot shows the AWS CloudWatch Metrics console. The search bar at the top has the query 'AWS Lambda'. The results table lists two metrics:

Metric	Last updated
Lambda Metrics	1 minute ago
Lambda Metrics (CloudWatch Metrics Metrics Insights)	1 minute ago

The screenshot shows the AWS ECR console. The sidebar on the left shows a 'Private registry' section with 'Repositories' and 'Features & Settings' options. The main area shows a repository named 'test'. The 'Images' tab is selected, displaying the following table:

Image tag	Artifact type	Pushed at	Size (MB)	Image URI	Digest
No images No images to display					

# Mac/Linux

Use the following steps to authenticate and push an image to your repository. For additional registry authentication methods, including the Amazon ECR credential helper, see [Registry authentication](#).

1. Retrieve an authentication token and authenticate your Docker client to your registry. Use the AWS CLI:

```
 aws ecr get-login-password --region us-east-1 | docker login --username AWS --password-stdin  
851725192411.dkr.ecr.us-east-1.amazonaws.com
```

Note: if you receive an error using the AWS CLI, make sure that you have the latest version of the AWS CLI and Docker installed.

2. Build your Docker image using the following command. For information on building a Docker file from scratch, see the instructions [here](#). You can skip this step if your image has already been built:

```
 docker build -t test .
```

3. After the build is completed, tag your image so you can push the image to this repository:

```
 docker tag test:latest 851725192411.dkr.ecr.us-east-1.amazonaws.com/test:latest
```

4. Run the following command to push this image to your newly created AWS repository:

```
 docker push 851725192411.dkr.ecr.us-east-1.amazonaws.com/test:latest
```

## # windows

1. Retrieve an authentication token and authenticate your Docker client to your registry. Use the AWS TOOLS for PowerShell:

```
 (Get-ECRLoginCommand).Password | docker login --username AWS --password-stdin 851725192411.dkr.ecr.us-east-1.amazonaws.com
```

Note: if you receive an error using the AWS TOOLS for PowerShell, make sure that you have the latest version of the AWS TOOLS for PowerShell and Docker installed.

2. Build your Docker image using the following command. For information on building a Docker file from scratch, see the instructions [here](#). You can skip this step if your image has already been built:

```
 docker build -t test .
```

3. After the build is completed, tag your image so you can push the image to this repository:

```
 docker tag test:latest 851725192411.dkr.ecr.us-east-1.amazonaws.com/test:latest
```

4. Run the following command to push this image to your newly created AWS repository:

```
 docker push 851725192411.dkr.ecr.us-east-1.amazonaws.com/test:latest
```

```
stage('8. Login to ECR & tag image') {  
    steps {  
        withCredentials([string(credentialsId: 'access-key', variable: 'AWS_ACCESS_KEY'),  
                      string(credentialsId: 'secret-key', variable: 'AWS_SECRET_KEY')]) {  
            sh """  
                aws ecr get-login-password --region us-east-1 | docker login --username AWS --password-stdin ${params.AWS_ACCOUNT_ID}.dkr.ecr.us-east-1.amazonaws.com  
                docker tag ${params.ECR_REPO_NAME} ${params.AWS_ACCOUNT_ID}.dkr.ecr.us-east-1.amazonaws.com/${params.ECR_REPO_NAME}:${BUILD_NUMBER}  
                docker tag ${params.ECR_REPO_NAME} ${params.AWS_ACCOUNT_ID}.dkr.ecr.us-east-1.amazonaws.com/${params.ECR_REPO_NAME}:latest  
            """  
        }  
    }  
}
```

# Build\_Number plays a vital role in jenkins that Signifies the Versions of images stored

<https://wiki.jenkins.io/display/JENKINS/Building+a+software+project>

The screenshot shows the Jenkins Pipeline configuration page. The pipeline script is defined as follows:

```
1 pipeline {  
2     agent any  
3  
4     stages {  
5         stage('Hello') {  
6             steps {  
7                 echo 'Hello World'  
8             }  
9         }  
10    }  
11}  
12
```

Below the script, there is a checked checkbox for "Use Groovy Sandbox". At the bottom, there are "Save" and "Apply" buttons.

The screenshot shows the Jenkins Pipeline status page for the "build-pipeline" job. The pipeline has run successfully, indicated by a green checkmark icon. The status bar shows "Status" and "build-pipeline". The sidebar includes links for "Changes", "Build Now", "Configure", "Delete Pipeline", "Full Stage View", "Stages", "Rename", and "Pipeline Syntax". The "Permalinks" section lists the last four builds: "Last build (#1, 42 min ago)", "Last stable build (#1, 42 min ago)", "Last successful build (#1, 42 min ago)", and "Last completed build (#1, 42 min ago)". Below this is a "Build History" section with a "trend" dropdown and a "Filter..." input field. The first build is highlighted with a green checkmark and the number "#1".

This is a zoomed-in view of the Jenkins Build History page for build #1. It shows the build was started on "Oct 20, 2024, 2:28 AM". The "trend" dropdown is set to "trend". A "Filter..." input field is present. The build number "#1" is highlighted with a green checkmark.

Environment Variable	Description
BUILD_NUMBER	The current build number, such as "153"
BUILD_ID	The current build id, such as "2005-08-22_23-59-59" (YYYY-MM-DD_hh-mm-ss, defunct since version 1.597)
BUILD_URL	The URL where the results of this build can be found (e.g. http://buildserver/jenkins/job/MyJobName/666/)
NODE_NAME	The name of the node the current build is running on. Equals 'master' for master node.
JOB_NAME	Name of the project of this build. This is the name you gave your job when you first set it up. It's the third column of the Jenkins Dashboard main page.
BUILD_TAG	String of jenkins-\$(JOB_NAME)-\$(BUILD_NUMBER). Convenient to put into a resource file, a jar file, etc for easier identification.

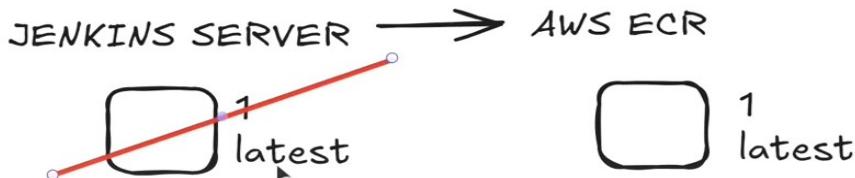
#9<sup>th</sup> Stage : Push to AWS ECR: Tags and pushes the Docker image to ECR.

4. Run the following command to push this image to your newly created AWS repository:

```
docker push 851725192411.dkr.ecr.us-east-1.amazonaws.com/test:latest
```

```
stage('9. Push image to ECR') {
    steps {
        withCredentials([string(credentialsId: 'access-key', variable: 'AWS_ACCESS_KEY'),
                       string(credentialsId: 'secret-key', variable: 'AWS_SECRET_KEY')]) {
            sh """
                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
            """
        }
    }
}
```

# Note once image is Pushed in AWS ECR , We need to Clean up the images stored in Jenkins to remove unnecessary memory space



# 10<sup>th</sup> Final Stage Clean up images in Jenkins

```
stage('10. Cleanup Images') {
    steps [
        sh """
            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
        docker images
        """
    ]
}
```

## # tools

The screenshot shows the Jenkins 'Tools' configuration page at the URL `100.24.54.83:8080/manage/configureTools/`. The page title is 'Manage Jenkins > Tools'. A sub-header says 'Use default maven global settings'. Below this, under 'JDK installations', there is a section titled 'JDK installations ^' with a status of 'Edited'. A button labeled 'Add JDK' is visible. A new entry for 'JDK' is listed, which includes:

- Name:** `JDK`
- Install automatically:**
- Install from adoptium.net:**
- Version:** `jdk-17.0.8.1+1`

← → ⌂ Not secure 100.24.54.83:8080/manage/configureTools/

Dashboard > Manage Jenkins > Tools

NodeJS installations ^ Edited

Add NodeJS

**NodeJS**

Name: NodeJS

Install automatically ?

**Install from nodejs.org**

Version: NodeJS 16.20.0

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 semicolons.

```
pipeline {
    agent any

    parameters {
        string(name: 'ECR_REPO_NAME', defaultValue: 'amazon-prime', description: 'Enter repository name')
        string(name: 'AWS_ACCOUNT_ID', defaultValue: '851725192411', description: 'Enter AWS Account ID') // Added missing quote
    }

    tools {
        jdk 'JDK'
        nodejs 'NodeJS'
    }
}
```

# Combined all the stages .. @ once .. Paste it on Jenkins Pipeline Script

Not secure 100.24.54.83:8080/job/build-pipeline/configure

Dashboard > build-pipeline > Configuration

### Configure

General

Advanced Project Options

Pipeline

Definition

Pipeline script

```

94
95
96
97
98
99
100
101
102
103
104
105
106
107
108
109
110
111
    }
}

stage('10, Cleanup Images') {
    steps {
        sh """
            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
            docker images """
    }
}

```

Use Groovy Sandbox ?

[Pipeline Syntax](#)

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

# save it

# PART 1 – CI -PIPELINE ....

Not secure 100.24.54.83:8080/job/build-pipeline/

Status build-pipeline

</> Changes

▷ Build with Parameters

⚙ Configure

Delete Pipeline

🔍 Full Stage View

Average stage times:

Declarative: Tool Install

4s

#2 Oct 20 08:51 No Changes

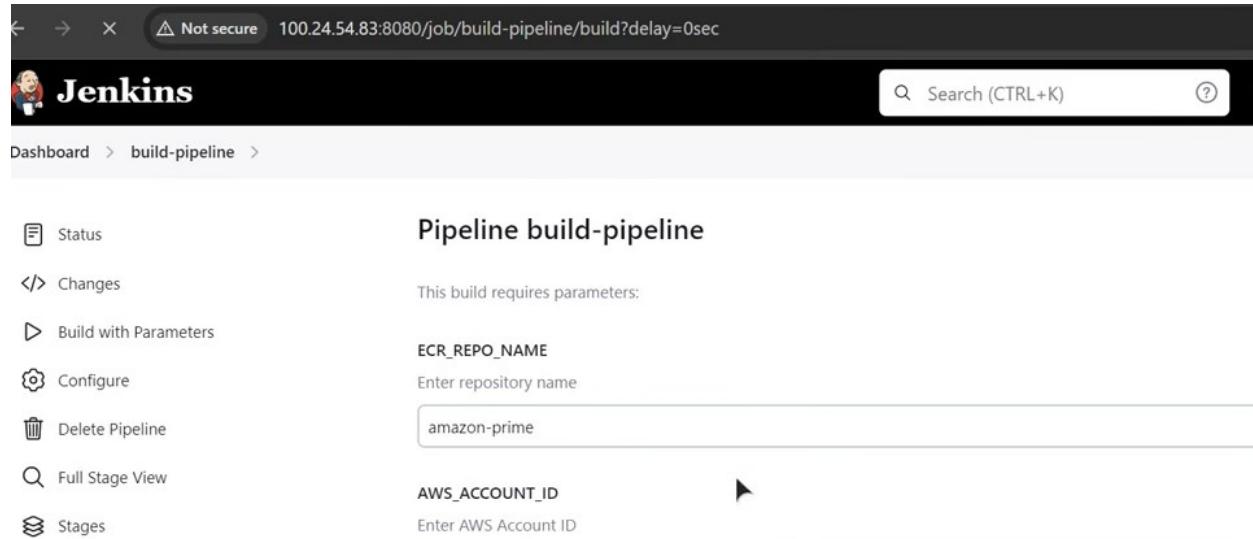
4s aborted

Stages

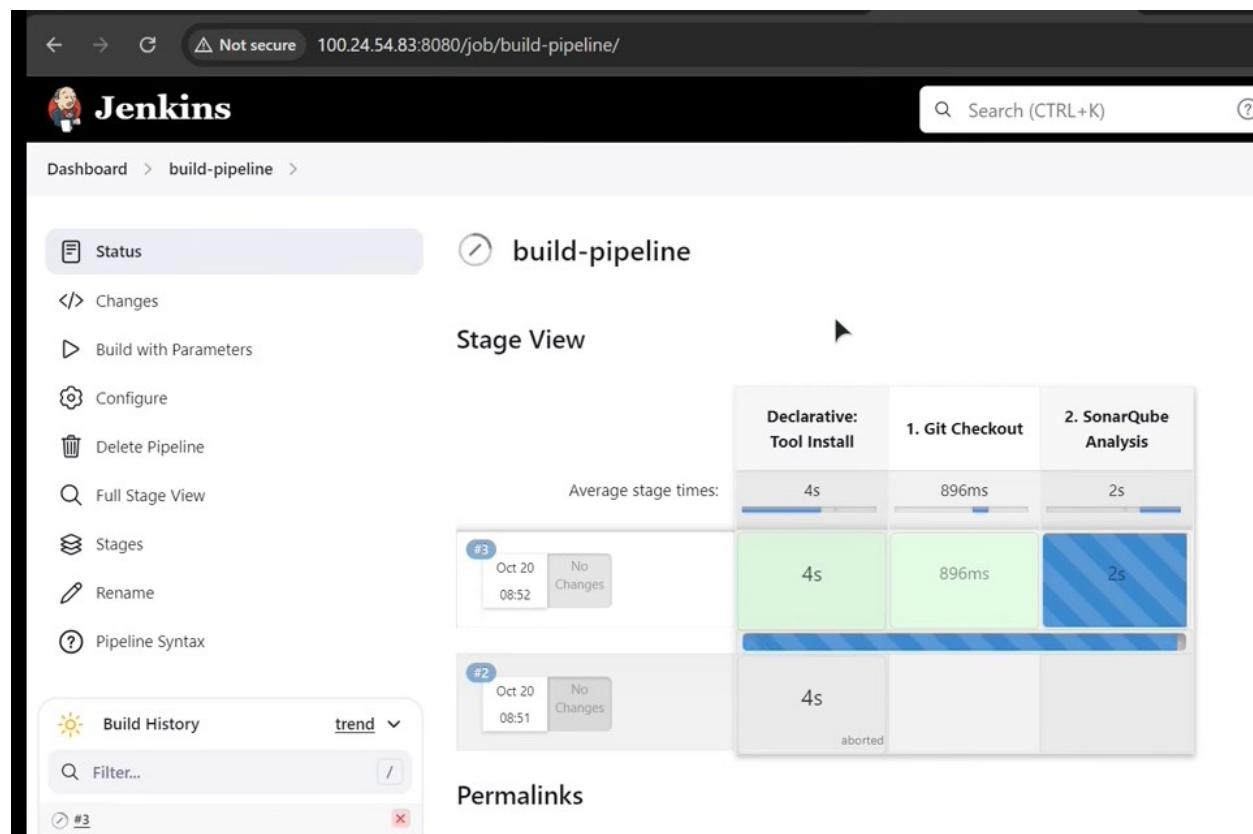
Rename

...

# Click Build with parameter



This screenshot shows the Jenkins Pipeline configuration page for a job named "build-pipeline". The left sidebar contains links for Status, Changes, Build with Parameters, Configure, Delete Pipeline, Full Stage View, and Stages. The main area is titled "Pipeline build-pipeline" and displays a "Build with Parameters" section. It requires parameters: ECR\_REPO\_NAME (repository name set to "amazon-prime") and AWS\_ACCOUNT\_ID (AWS Account ID). A large green arrow points to the "Build with Parameters" link.



This screenshot shows the Jenkins Pipeline stage view for the "build-pipeline" job. The left sidebar includes Status, Changes, Build with Parameters, Configure, Delete Pipeline, Full Stage View, Stages, Rename, and Pipeline Syntax. The main area is titled "Stage View" and shows a table of stages. The first stage, "Declarative: Tool Install", took 4s. The second stage, "1. Git Checkout", took 896ms. The third stage, "2. SonarQube Analysis", took 2s. Below the table, two builds are listed: #3 (Oct 20 08:52, No Changes) and #2 (Oct 20 08:51, No Changes). Build #3 is shown as completed with a green bar, while build #2 is shown as aborted. A large green arrow points to the "Stage View" link.

# in SonarQube .. Projects

Not secure 100.24.54.83:9000/projects

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

My Favorites All

Filters

Quality Gate

- Passed 1
- Failed 0

Reliability ( Bugs )

- A rating 0
- B rating 0
- C rating 0
- D rating 1
- E rating 0

Security ( Vulnerabilities )

- A rating 0
- B rating 1

Search by project name or key

1 project(s)

Perspective: Overall Status Sort by: Name

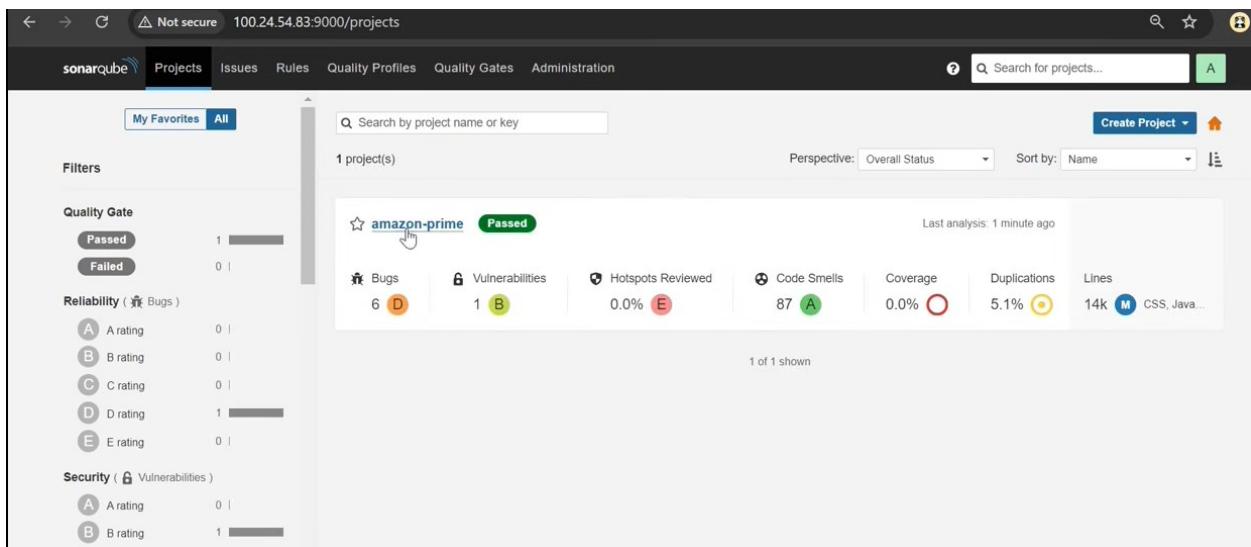
Last analysis: 1 minute ago

amazon-prime Passed

Bugs: 6 (D) Vulnerabilities: 1 (B) Hotspots Reviewed: 0.0% (E)

Code Smells: 87 (A) Coverage: 0.0% Duplications: 5.1% Lines: 14k (M) CSS, Java...

1 of 1 shown



Not secure 100.24.54.83:8080/job/build-pipeline/

Stage Logs (1. Git Checkout)

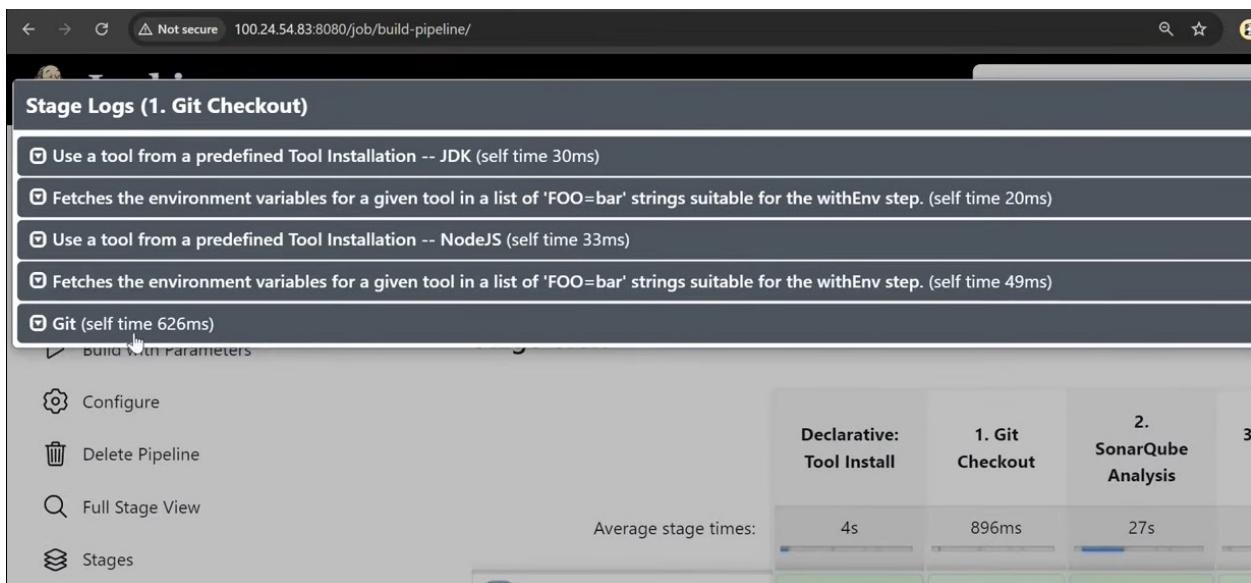
- Use a tool from a predefined Tool Installation -- JDK (self time 30ms)
- Fetches the environment variables for a given tool in a list of 'FOO=bar' strings suitable for the withEnv step. (self time 20ms)
- Use a tool from a predefined Tool Installation -- NodeJS (self time 33ms)
- Fetches the environment variables for a given tool in a list of 'FOO=bar' strings suitable for the withEnv step. (self time 49ms)
- Git (self time 626ms)

Build With Parameters

Configure Delete Pipeline Full Stage View Stages

Declarative: Tool Install 1. Git Checkout 2. SonarQube Analysis 3

Average stage times: 4s 896ms 27s



## # SonarQube Analysis

```
← → ⚡ Not secure 100.24.54.83:8080/job/build-pipeline/ ⚡

Stage Logs (2. SonarQube Analysis)

- Use a tool from a predefined Tool Installation -- JDK (self time 27ms)
- Fetches the environment variables for a given tool in a list of 'FOO=bar' strings suitable for the withEnv step. (self time 21ms)
- Use a tool from a predefined Tool Installation -- NodeJS (self time 30ms)
- Fetches the environment variables for a given tool in a list of 'FOO=bar' strings suitable for the withEnv step. (self time 37ms)
- Shell Script -- /var/lib/jenkins/tools/hudson.plugins.sonar.SonarRunnerInstallation/SonarQube_Scanner/bin/sonar-scanner -Dsonar.projectName=amazon-prime -Dsonar.projectKey=amazon-prime (self time 26s)



```
scanner/bin/sonar-scanner -Dsonar.projectName=amazon-prime -Dsonar.projectKey=amazon-prime
03:22:57.294 INFO Scanner configuration file: /var/lib/jenkins/tools/hudson.plugins.sonar.SonarRunnerInstallation/SonarQube_Scanner/conf/sonar-scanner.properties
03:22:57.296 INFO Project root configuration file: NONE
03:22:57.310 INFO SonarScanner CLI 6.2.1.4610
03:22:57.312 INFO Java 17.0.8.1 Eclipse Adoptium (64-bit)
03:22:57.312 INFO Linux 6.8.0-1012-aws amd64
03:22:57.342 INFO User cache: /var/lib/jenkins/.sonar/cache
03:22:58.485 INFO Communicating with SonarQube Server 9.9.7.96285
03:22:58.823 INFO Load global settings
03:22:58.885 INFO Load global settings (done) | time=63ms
03:22:58.886 INFO Server id: 1478411E-AZKnqcITH9GsdBzx6AZ
03:22:58.894 INFO User cache: /var/lib/jenkins/.sonar/cache
03:22:58.892 INFO Load/download plugins
03:22:58.892 INFO Load plugins index
03:22:58.943 INFO Load plugins index (done) | time=51ms
03:23:00.892 INFO Load/download plugins (done) | time=2000ms
03:23:01.614 INFO Process project properties
03:23:01.614 INFO Process project properties (done) | time=0ms
03:23:01.616 INFO Execute project builders
```


```

```
← → ⚡ Not secure 100.24.54.83:8080/job/build-pipeline/ ⚡

INFO Sensor Analysis Warnings import [csharp]
INFO Sensor Analysis Warnings import [csharp] (done) | time=1ms
INFO Sensor Zero Coverage Sensor
INFO Sensor Zero Coverage Sensor (done) | time=19ms
INFO SCM Publisher SCM provider for this project is: git
INFO SCM Publisher 29 source files to be analyzed
INFO SCM Publisher 29/29 source files have been analyzed (done) | time=618ms
INFO CPD Executor 1 file had no CPD blocks
INFO CPD Executor Calculating CPD for 17 files
INFO CPD Executor CPD calculation finished (done) | time=39ms
INFO Analysis report generated in 70ms, dir size=926.2 kB
INFO Analysis report compressed in 83ms, zip size=268.4 kB
INFO Analysis report uploaded in 374ms
INFO ANALYSIS SUCCESSFUL, you can find the results at: http://100.24.54.83:9000/dashboard?id=amazon-prime
INFO Note that you will be able to access the updated dashboard once the server has processed the submitted analysis
INFO More about the report processing at http://100.24.54.83:9000/api/ce/task?id=AZKn81sBH9GsdBzx6KTA
INFO Analysis total time: 21.233 s
INFO EXECUTION SUCCESS
TINFO Total time: 25.017s
```

# Go to SonarQube .. click on amazon-prime

Not secure 100.24.54.83:9000/dashboard?id=amazon-prime

sonarqube Projects Issues Rules Quality Profiles Quality Gates Administration

amazon-prime main Overview Issues Security Hotspots Measures Code Activity

October 20, 2024 at 8:53 AM Version not provided

Project Settings Project Information

**QUALITY GATE STATUS**

**Passed**  
All conditions passed

**MEASURES**

New Code Overall Code

6 Bugs Reliability D

1 Vulnerabilities Security B

3 Security Hotspots 0.0% Reviewed Security Review E

3h 57min Debt 87 Code Smells Maintainability A

## # quality-gate

Not secure 100.24.54.83:8080/job/build-pipeline/

Stage Logs (3. Quality Gate)

- Use a tool from a predefined Tool Installation -- JDK (self time 47ms)
- Fetches the environment variables for a given tool in a list of 'FOO=bar' strings suitable for the withEnv step. (self time 63ms)
- Use a tool from a predefined Tool Installation -- NodeJS (self time 44ms)
- Fetches the environment variables for a given tool in a list of 'FOO=bar' strings suitable for the withEnv step. (self time 57ms)
- Wait for SonarQube analysis to be completed and return quality gate status (self time 164ms)

Checking status of SonarQube task 'AZKn81sBH9GsdBzx6KTa' on server 'sonar-server'  
SonarQube task 'AZKn81sBH9GsdBzx6KTa' status is 'SUCCESS'  
SonarQube task 'AZKn81sBH9GsdBzx6KTa' completed. Quality gate is 'OK'

Full Stage View Average stage times: 4s 896ms 27s 570ms 20s 8s 1min 50s

Stages Oct 20 08:52 No Changes

Rename Pipeline Syntax

## # npm insalled

**Stage Logs (4. Install npm)**

- Use a tool from a predefined Tool Installation -- JDK (self time 42ms)
- Fetches the environment variables for a given tool in a list of 'FOO=bar' strings suitable for the withEnv step. (self time 32ms)
- Use a tool from a predefined Tool Installation -- NodeJS (self time 43ms)
- Fetches the environment variables for a given tool in a list of 'FOO=bar' strings suitable for the withEnv step. (self time 54ms)
- Shell Script -- npm install (self time 20s)

```
a.org/en-US/docs/Web/JavaScript/Reference/Global_Objects/Array/sort#browser_compatibility
npm WARN deprecated sourcemap-codec@1.4.8: Please use @jridgewell/sourcemap-codec instead
npm WARN deprecated workbox-cacheable-response@6.6.0: workbox-background-sync@6.6.0
npm WARN deprecated svgo@1.3.2: This SVGO version is no longer supported. Upgrade to v2.x.x.

added 1490 packages, and audited 1491 packages in 18s
```

# trivy scan

**Stage Logs (5. Trivy Scan)**

- Use a tool from a predefined Tool Installation -- JDK (self time 32ms)
- Fetches the environment variables for a given tool in a list of 'FOO=bar' strings suitable for the withEnv step. (self time 24ms)
- Use a tool from a predefined Tool Installation -- NodeJS (self time 31ms)
- Fetches the environment variables for a given tool in a list of 'FOO=bar' strings suitable for the withEnv step. (self time 38ms)
- Shell Script -- trivy fs . > trivy.txt (self time 8s)

```
+ trivy fs .
2024-10-20T03:23:46Z    INFO    [vulndb] Need to update DB
2024-10-20T03:23:46Z    INFO    [vulndb] Downloading vulnerability DB...
2024-10-20T03:23:46Z    INFO    [vulndb] Downloading artifact...      repo="ghcr.io/aquasecurity/trivy-db:2"
21.41 MiB / 54.36 MiB [----->_____] 39.39% ? p/s ?42.39 MiB / 54.36 MiB [----->_____]
77.98% ? p/s ?54.36 MiB / 54.36 MiB [----->_____] 100.00% ? p/s ?54.36 MiB / 54.36 MiB [----->_____
100.00% 54.86 MiB p/s ETA 054.36 MiB / 54.36 MiB [----->] 100.00% 54.86 MiB p/s ETA 054.36 MiB / 54.36 MiB [----->]
100.00% 54.86 MiB p/s ETA 054.36 MiB / 54.36 MiB [----->] 100.00% 51.32 MiB p/s ETA 054.36 MiB / 54.36 MiB [----->]
100.00% 51.32 MiB p/s ETA 054.36 MiB / 54.36 MiB [----->] 100.00% 51.32 MiB p/s ETA 054.36 MiB / 54.36 MiB [----->]
100.00% 48.01 MiB p/s ETA 054.36 MiB / 54.36 MiB [----->] 100.00% 48.01 MiB p/s ETA 054.36 MiB / 54.36 MiB [----->]
6 MiB [----->] 100.00% 48.01 MiB p/s ETA 054.36 MiB / 54.36 MiB [----->] 100.00% 44.91 MiB p/s ETA 054.36 MiB / 54.36 MiB [----->]
4.36 MiB [----->] 100.00% 44.91 MiB p/s ETA 054.36 MiB / 54.36 MiB [----->] 100.00% 44.91 MiB p/s ETA 054.36 MiB / 54.36 MiB [----->]
ib p/s ETA 054.36 MiB / 54.36 MiB [----->] 100.00% 42.01 MiB p/s ETA 054.36 MiB / 54.36 MiB [----->] 100.00% 42.01 MiB p/s ETA 054.36 MiB / 54.36 MiB [----->]
0.00% 16.72 MiB p/s 3.5s2024-10-20T03:23:49Z    INFO    [vulndb] Artifact successfully downloaded repo="ghcr.io/aquasecurity/trivy-db:2"
2024-10-20T03:23:49Z    INFO    [vuln] Vulnerability scanning is enabled
2024-10-20T03:23:49Z    INFO    [secret] Secret scanning is enabled
```

# trivy.txt .. exsistence Click on # ( Build\_Number)

← → ⌂ Not secure 100.24.54.83:8080/job/build-pipeline/

Dashboard > build-pipeline >

Build with Parameters

Configure

Delete Pipeline

Full Stage View

Stages

Rename

Pipeline Syntax

**Stage View**

Average stage times:

Declarative: Tool Install	1. Git Checkout	2. SonarQube Analysis	3. Quality Gate
4s	896ms	27s	570ms

#3 Oct 20 08:52 No Changes

#2 Oct 20 08:51 No Changes

4s aborted

**Build History**

trend ▾

Filter... In progress > Console Output

- #3 Oct 20, 2024, 3:22 AM
- #2 Oct 20, 2024, 3:21 AM
- #1 Oct 20, 2024, 2:28 AM

Atom feed for all Atom feed for failures

**Permalinks**

- Last build (#3), 0.3 sec ago
- Last stable build (#1), 54 min ago
- Last successful build (#1), 54 min ago
- Last unsuccessful build (#2), 1 min 1 sec ago
- Last completed build (#2), 1 min 1 sec ago

## # Workspaces

← → ⌂ Not secure 100.24.54.83:8080/job/build-pipeline/3/console

Git Build Data

Pipeline Overview

Pipeline Console

Thread Dump

Pause/resume

Replay

Pipeline Steps

Workspaces

Status

Changes

Console Output

Edit Build Information

Parameters

Timings

**Workspaces for build-pipeline #3**

- /var/lib/jenkins/workspace/build\_pipeline on built-in



# Jenkins

Search (CTRL+K)

Dashboard > build-pipeline > #3 > Allocate node : Start > Workspace > Workspace

Up

Status

Console Output

Workspace

## Workspace

/

git  
scannerwork  
k8s\_files  
node\_modules  
pipeline\_script  
public  
src  
terraform\_code

access.sh	Oct 20, 2024, 3:22:56 AM	1.32 KiB		
Dockerfile	Oct 20, 2024, 3:22:56 AM	542 B		
package.json	Oct 20, 2024, 3:22:56 AM	814 B		
package-lock.json	Oct 20, 2024, 3:23:42 AM	664.33 KiB		
Project_WriteUp.docx	Oct 20, 2024, 3:22:56 AM	24.02 KiB		
README.md	Oct 20, 2024, 3:22:56 AM	13.64 KiB		
trivy.txt	Oct 20, 2024, 3:23:52 AM	26.51 KiB		

(all files in zip)

## # Open – encrypted

```
# docker img
```

<input checked="" type="checkbox"/> Use a tool from a predefined Tool Installation -- JDK (self time 30ms)
<input checked="" type="checkbox"/> Fetches the environment variables for a given tool in a list of 'FOO=bar' strings suitable for the withEnv step. (self time 24ms)
<input checked="" type="checkbox"/> Use a tool from a predefined Tool Installation -- NodeJS (self time 38ms)
<input checked="" type="checkbox"/> Fetches the environment variables for a given tool in a list of 'FOO=bar' strings suitable for the withEnv step. (self time 41ms)
<input checked="" type="checkbox"/> Shell Script -- docker build -t amazon-prime . (self time 2min 11s)

```
+ docker build -t amazon-prime[.
#0 building with "default" instance using docker driver

#1 [internal] load build definition from Dockerfile
#1 transferring dockerfile: 581B done
#1 DONE 0.1s

#2 [internal] load metadata for docker.io/library/node:alpine
#2 DONE 0.4s

#3 [internal] load .dockerignore
#3 transferring context: 2B done
#3 DONE 0.0s
```

## # ECR created

Stage Logs (7. Create ECR repo)
<input checked="" type="checkbox"/> Use a tool from a predefined Tool Installation -- JDK (self time 24ms)
<input checked="" type="checkbox"/> Fetches the environment variables for a given tool in a list of 'FOO=bar' strings suitable for the withEnv step. (self time 44ms)
<input checked="" type="checkbox"/> Use a tool from a predefined Tool Installation -- NodeJS (self time 47ms)
<input checked="" type="checkbox"/> Fetches the environment variables for a given tool in a list of 'FOO=bar' strings suitable for the withEnv step. (self time 59ms)
<input checked="" type="checkbox"/> Shell Script -- aws configure set aws_access_key_id \$AWS_ACCESS_KEY aws configure set aws_secret_access_key \$AWS_SECRET_KEY aws ecr describe-repositories --repository-names amazon-prime --region us-east-1    aws ecr create-repository --repository-name amazon-prime --region us-east-1 (self time 3s)

Warning: A secret was passed to "sh" using Groovy String interpolation, which is insecure.  
Affected argument(s) used the following variable(s): [AWS\_SECRET\_KEY, AWS\_ACCESS\_KEY]  
See <https://jenkins.io/redirect/groovy-string-interpolation> for details.

```
+ aws configure set aws_access_key_id ****
+ aws configure set aws_secret_access_key ****
+ aws ecr describe-repositories --repository-names amazon-prime --region us-east-1

An error occurred (RepositoryNotFoundException) when calling the DescribeRepositories operation: The repository with name 'amazon-prime' does not exist in the registry with id '309395755719'.
+ aws ecr create-repository --repository-name amazon-prime --region us-east-1
{
```

## # AWS ECR

The screenshot shows the AWS Lambda console. At the top, there's a green banner indicating a successful deployment with the message 'Successfully deployed test'. Below this, the navigation bar includes links for EC2, S3, VPC, RDS, EFS, Simple Notification Service, CloudWatch, CloudTrail, Trusted Advisor, ElastiCache, Route 53, Support, CloudFormation, CloudFront, and ELB. The main content area is titled 'Private registry' and 'Repositories'. It shows a table with one repository entry:

Repository name	URI	Created at	Tag immutability	Encryption type
amazon-prime	1.amazonaws.com/amazon-prime	October 20, 2024, 08:56:09 (UTC+05:5)	Mutable	AES-256

# note Build # 3 .. here a new Repo will be created in AWS ECR with Build # 3 .. Test it Out

← → ⌂ Not secure 100.24.54.83:8080/job/build-pipeline/

Dashboard > build-pipeline >

Status build-pipeline

</> Changes

▷ Build with Parameters

⚙ Configure

Delete Pipeline

Full Stage View

Stages

Rename

Pipeline Syntax

Build History trend ▾

Filter... /

#3 Oct 20 08:52 No Changes

#2 Oct 20 08:51 No Changes

Average stage times:

Declarative: Tool Install	1. Git Checkout	2. SonarQube Analysis	3. Quality Gate	4. Install npm	5. Trivy Scan
4s	896ms	27s	570ms	20s	8s

Stage View

Declarative: Tool Install 1. Git Checkout 2. SonarQube Analysis 3. Quality Gate 4. Install npm 5. Trivy Scan

4s 896ms 27s 570ms 20s 8s

4s 896ms 27s 570ms 20s 8s

4s aborted

Permalinks

#

Images (1)

Search artifacts

<input type="checkbox"/>	Image tag	Artifact type	Pushed at	Size (MB)	Image L
<input type="checkbox"/>	3, latest	Image	October 20, 2024, 08:57:04 (UTC+05.5)	206.94	<input type="button" value="Copy URI"/>

# Login into ECR Tag Repo

← → ⌂ Not secure 100.24.54.83:8080/job/build-pipeline/

Stage Logs (8. Login to ECR & tag image)

Use a tool from a predefined Tool Installation -- JDK (self time 26ms)

Fetches the environment variables for a given tool in a list of 'FOO=bar' strings suitable for the withEnv step. (self time 24ms)

Use a tool from a predefined Tool Installation -- NodeJS (self time 34ms)

Fetches the environment variables for a given tool in a list of 'FOO=bar' strings suitable for the withEnv step. (self time 39ms)

# Push img to ECR

```

← → ⌂ Not secure 100.24.54.83:8080/job/build-pipeline/
Stage Logs (9. Push image to ECR)

 Use a tool from a predefined Tool Installation -- JDK (self time 26ms)
 Fetches the environment variables for a given tool in a list of 'FOO=bar' strings suitable for the withEnv step. (self time 25ms)
 Use a tool from a predefined Tool Installation -- NodeJS (self time 28ms)
 Fetches the environment variables for a given tool in a list of 'FOO=bar' strings suitable for the withEnv step. (self time 32ms)

```

The screenshot shows the AWS ECR console. The top navigation bar includes services like EC2, S3, VPC, RDS, EFS, Simple Notification Service, CloudWatch, CloudTrail, Trusted Advisor, ElastiCache, and Route 53. The main page displays the 'Amazon Elastic Container Registry' under 'Private registry'. A sidebar on the left lists options such as Repositories, Summary, Images, Permissions, Lifecycle Policy, Repository tags, and Features & Settings. The right panel is titled 'Image' and shows the details for the repository 'amazon-prime'. It lists the digest as sha256:99f262a2b22a00c4c64b83a9dd404dac09f9765779bc627cef2a296e47d3eb53 and the tag as 3, latest.

# final – imges clean up

```

← → ⌂ Not secure 100.24.54.83:8080/job/build-pipeline/
Stage Logs (10. Cleanup Images)

 Use a tool from a predefined Tool Installation -- JDK (self time 36ms)
 Fetches the environment variables for a given tool in a list of 'FOO=bar' strings suitable for the withEnv step. (self time 39ms)
 Use a tool from a predefined Tool Installation -- NodeJS (self time 35ms)
 Fetches the environment variables for a given tool in a list of 'FOO=bar' strings suitable for the withEnv step. (self time 60ms)

```

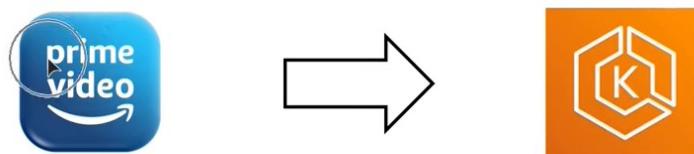
```

+ docker images
REPOSITORY      TAG          IMAGE ID       CREATED        SIZE
amazon-prime    latest        24de7cd09bb2   About a minute ago  710MB
grafana/grafana latest        2a1d608bb4df   2 weeks ago    461MB
sonarqube       lts-community e88d8cc07644   2 weeks ago    605MB
prom/prometheus latest        4022a502929b   7 weeks ago    275MB

```

# 2<sup>nd</sup> PART DEPLOYMENT ---- CD PART

# RUN JENKINS RELEASE PIPELINE

A screenshot of a code editor displaying a Jenkins pipeline configuration file named 'deployment\_eks.groovy'. The code defines a pipeline with parameters for AWS region, account ID, repository name, version, and cluster name. It includes stages for cloning the GitHub repository and logging into the EKS cluster using AWS credentials. The code editor interface shows syntax highlighting and various toolbars.

The screenshot shows a code editor window titled "DevopsProject2" displaying a Jenkinsfile named "main". The file contains Groovy script for a pipeline. The pipeline starts with a "Clone GitHub Repository" stage, followed by a "Login to EKS" stage which updates the kubeconfig. Then it moves to a "Select Image Version" stage where it defines the ECR image name and updates deployment files. Finally, it reaches a commented-out stage for deploying to EKS.

```
1 pipeline {
2     stages {
3         stage("Clone GitHub Repository") {
4             steps {
5                 ...
6             }
7         }
8         stage("Login to EKS") {
9             steps {
10                script {
11                    withCredentials([string(credentialsId: 'access-key', variable: 'AWS_ACCESS_KEY'),
12                                    string(credentialsId: 'secret-key', variable: 'AWS_SECRET_KEY')]) {
13                        sh "aws eks --region ${params.AWS_REGION} update-kubeconfig --name ${params.CLUSTER_NAME}"
14                    }
15                }
16            }
17        }
18        stage ("Select Image Version") {
19            steps {
20                script {
21                    def ECR_IMAGE_NAME = "${params.AWS_ACCOUNT_ID}.dkr.ecr.${params.AWS_REGION}.amazonaws.com/${params.ECR_REPO_NAME}:$"
22                    sh "sed -i 's|image: .*|image: ${ECR_IMAGE_NAME}|' k8s_files/deployment.yaml"
23                }
24            }
25        }
26        // -----
27        // USE THIS STAGE TO DEPLOY IMAGE AS PER THE VERSION NO
28    }
29}
```

This screenshot shows the same Jenkinsfile as the first one, but with additional stages added below the "Select Image Version" stage. A new "Deploy to EKS" stage is introduced, containing steps to apply deployment and service files using Kubernetes commands like "kubectl apply".

```
1 pipeline {
2     stages {
3         stage ("Select Image Version") {
4             steps {
5                 script {
6                     def ECR_IMAGE_NAME = "${params.AWS_ACCOUNT_ID}.dkr.ecr.${params.AWS_REGION}.amazonaws.com/${params.ECR_REPO_NAME}:$"
7                     sh "sed -i 's|image: .*|image: ${ECR_IMAGE_NAME}|' k8s_files/deployment.yaml"
8                 }
9             }
10        }
11        // -----
12        // USE THIS STAGE TO DEPLOY IMAGE AS PER THE VERSION NO
13        // -----
14        stage("Deploy to EKS") {
15            steps {
16                script {
17                    // Apply the deployment and service files
18                    sh "kubectl apply -f k8s_files/deployment.yaml"
19                    sh "kubectl apply -f k8s_files/service.yaml"
20                }
21            }
22        }
23    }
24}
```

# Go to eks.ft file in terraform folder ...

The screenshot shows the VS Code interface with the following details:

- EXPLORER**: Shows a project structure with folders like `k8s_files`, `pipeline_script`, `public`, `src`, and `terraform_code`. Within `terraform_code`, there are sub-folders `eks_code`, `.terraform`, and files `eks.tf`, `provider.tf`, `vpc.tf`, `access.sh`, and `Dockerfile`.
- TERMINAL**: Displays the command `PS>terraform init` being run. The output shows the initialization process: "Initializing the backend...", "Initializing modules...", and "Downloading registry.terraform.io/terraform-aws-modules/eks/aws 19.15.1 for eks...". It also lists dependencies for the `eks` module.
- STATUS BAR**: Shows the current file is `eks.tf` and the status `key.pem U`.

```
PS>terraform init
Initializing the backend...
Initializing modules...
Downloading registry.terraform.io/terraform-aws-modules/eks/aws 19.15.1 for eks...
- eks in .terraform\modules\eks
- eks.eks_managed_node_group in .terraform\modules\eks\modules\eks-managed-node-group
- eks.eks_managed_node_group.user_data in .terraform\modules\eks\modules\_user_data
- eks.fargate_profile in .terraform\modules\eks\modules\fargate-profile
Downloading registry.terraform.io/terraform-aws-modules/kms/aws 1.1.0 for eks.kms...
```

The screenshot shows the VS Code interface with the following details:

- EXPLORER**: Shows a project structure with a single file `provider.tf` currently selected.
- EDITOR**: Displays the `provider.tf` code. The code defines a `locals` block for a VPC and an `aws` provider block.
- STATUS BAR**: Shows the current file is `provider.tf` and the status `DevopsProject2`.

```
locals {
    region = "us-east-1"
    name   = "amazon-prime-cluster"
    vpc_cidr = "10.0.0.0/16"
    azs     = ["us-east-1a", "us-east-1b"]
    public_subnets = ["10.0.1.0/24", "10.0.2.0/24"]
    private_subnets = ["10.0.3.0/24", "10.0.4.0/24"]
    intra_subnets  = ["10.0.5.0/24", "10.0.6.0/24"]
    tags = {
        Example = local.name
    }
}

provider "aws" {
    region = "us-east-1"
}
```

# VPC

```

1 module "vpc" {
2   source  = "terraform-aws-modules/vpc/aws"
3   version = "5.13.0"
4
5   name = local.name
6   cidr = local.vpc_cidr
7
8   azs           = local.azs
9   private_subnets = local.private_subnets
10  public_subnets  = local.public_subnets
11  intra_subnets   = local.intra_subnets
12
13  enable_nat_gateway = true
14
15  public_subnet_tags = {
16    "kubernetes.io/role/elb" = 1
17  }
18
19  private_subnet_tags = {
20    "kubernetes.io/role/internal-elb" = 1

```

VS Code interface showing the file tree on the left and the code editor on the right. The status bar at the bottom shows AWS profile: default, 25°C Mostly cloudy, and the date/time.

```

> public
> src
└─ terraform_code
  └─ eks_code
    > .terraform
    & .terraform.lockfile U
    & .terraform.tfstate.l...
    & eks.tf

```

VS Code interface showing the file tree on the left and the code editor on the right. The status bar at the bottom shows PS>terraform apply --auto-approve.

## # Eks Cluster & nodegroup

```

└─ eks_code
  > .terraform
  & .terraform.lockfile U
  & .terraform.tfstate.l...
  & eks.tf
  & provider.tf
  {<} terraform.tfstate U
  & vpc.tf
  $ access.sh
  & Dockerfile
  {<} package-lock.json
  {<} package.json
  & Project_WriteUp.docx

```

VS Code interface showing the file tree on the left and the code editor on the right. The status bar at the bottom shows PS>terraform apply --auto-approve.

> .terraform

  └ eks.tf

    └ provider.tf

      └ vpc.tf

      \$ access.sh

      └ Dockerfile

      └ package-lock.json

      └ packagejson

      └ Project\_WriteUp.docx

      └ README.md

> OUTLINE

> TIMELINE

```
module.eks.aws_eks_addon.this["kube-proxy"]: Creation complete after 47s [id=amazon-prime-cluster:kube-proxy]
module.eks.aws_eks_addon.this["vpc-cni"]: Creation complete after 48s [id=amazon-prime-cluster:vpc-cni]

Warning: Argument is deprecated

  with module.eks.aws_iam_role.this[0],
  on .terraform\modules\eks\main.tf line 285, in resource "aws_iam_role" "this":
285: resource "aws_iam_role" "this" {

The inline_policy argument is deprecated. Use the aws_iam_role_policy resource instead. If Terraform
should exclusively manage all inline policy associations (the current behavior of this argument), use
the aws_iam_role_policies_exclusive resource as well.

  (and 7 more similar warnings elsewhere)
```

Apply complete! Resources: 62 added, 0 changed, 0 destroyed.

[EC2](#) [S3](#) [VPC](#) [RDS](#) [EFS](#) [Simple Notification Service](#) [CloudWatch](#) [CloudTrail](#) [Trusted Advisor](#) [ElastiCache](#) [Route 53](#) [Support](#) [CloudFormation](#) [CloudFront](#) [Elastic Beans](#)

**Amazon Elastic Kubernetes Service**

**Clusters**

▼ Amazon EKS Anywhere

Enterprise Subscriptions [New](#)

▼ Related services

Amazon ECR

AWS Batch

Console settings

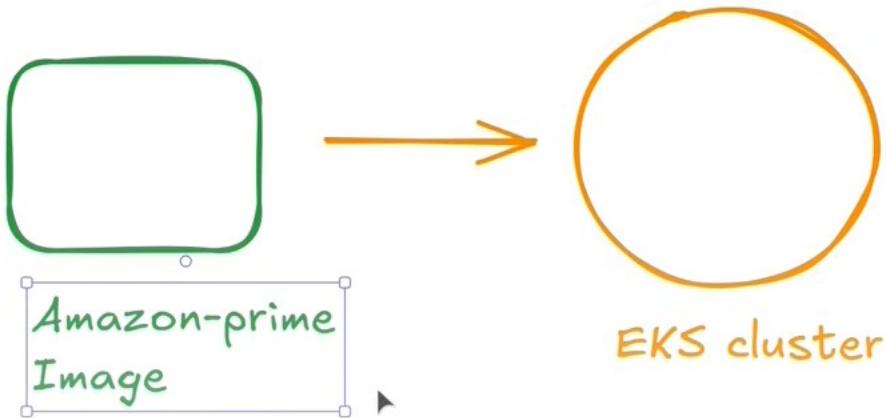
Documentation [?](#)

Submit feedback

EKS > Clusters

**Clusters (1) Info**

Cluster name	Status	Kubernetes version	Support period	Upgrade policy
amazon-prime-cluster	Active	1.31	Standard support until November 26, 2025	Extended



# Check Repo

To exit full screen, press Esc

Services N. Virginia

S3 VPC RDS EFS Simple Notification Service CloudWatch CloudTrail Trusted Advisor ElastiCache Route 53 Support CloudFormation CloudFront

Containers

# Amazon Elastic Container Registry

## Share and deploy container software, publicly or privately

Amazon Elastic Container Registry (ECR) is a fully managed container registry that makes it easy to store, share, and deploy container software, publicly or privately.

Create a repository

Create

Amazon Elastic Container Registry

Amazon ECR > Private registry > Repositories > amazon-prime

### amazon-prime

View push com

Private registry

Repositories

- Summary
- Images**
- Permissions
- Lifecycle Policy
- Repository tags

Features & Settings

Public registry

Repositories

Settings

#### Images (2)

<input type="checkbox"/>	Image tag	Artifact type	Pushed at	Size (MB)	Image URI	Digest
<input type="checkbox"/>	4, latest	Image	October 20, 2024, 09:07:26 (UTC+05.5)	206.94	<a href="#">Copy URI</a>	sha256:869cc6785cc...
<input type="checkbox"/>	3	Image	October 20, 2024, 08:57:04 (UTC+05.5)	206.94	<a href="#">Copy URI</a>	sha256:99f262a2b2...

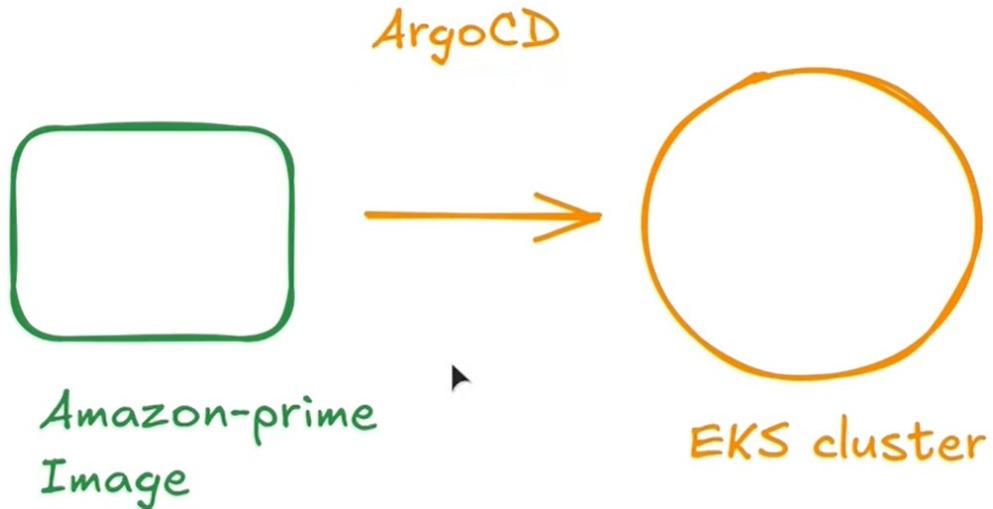
### amazon-prime

#### Images (2)

[Copy URI](#) [Details](#)

Search artifacts

<input type="checkbox"/>	Image tag	Artifact type	Pushed at	Size (MB)	Image URI
<input type="checkbox"/>	4, latest	Image	October 20, 2024, 09:07:26 (UTC+05.5)	206.94	<a href="#">Copy URI</a>
<input type="checkbox"/>	3	Image	October 20, 2024, 08:57:04 (UTC+05.5)	206.94	<a href="#">Copy URI</a>



#### # Deployment Pipeline

```

1 pipeline {
2   agent any
3
4   environment {
5     KUBECTL = '/usr/local/bin/kubectl'
6   }
7
8   parameters {
9     string(name: 'CLUSTER_NAME', defaultValue: 'amazon-prime-cluster', description: 'Enter your EKS cluster n
10 }
11
12 stages {
13   stage("Login to EKS") [
14     steps {
15       script {
16         withCredentials([string(credentialsId: 'access-key', variable: 'AWS_ACCESS_KEY'),
17                         |           |           |           | string(credentialsId: 'secret-key', variable: 'AWS_SECRET_KEY')]) {
18           sh "aws eks --region us-east-1 update-kubeconfig --name ${params.CLUSTER_NAME}"
19         }
20       }
21     }
22   }
23 }
```



#### # install helm chart for Prometheus + Grafana

```
stage("Configure Prometheus & Grafana") {
  steps {
    script {
      sh """
        helm repo add stable https://charts.helm.sh/stable || true
        helm repo add prometheus-community https://prometheus-community.github.io/helm-charts || true
        # Check if namespace 'prometheus' exists
        if kubectl get namespace prometheus > /dev/null 2>&1; then
          # If namespace exists, upgrade the Helm release
          helm upgrade stable prometheus-community/kube-prometheus-stack -n prometheus
        else
          # If namespace does not exist, create it and install Helm release
          kubectl create namespace prometheus
          helm install stable prometheus-community/kube-prometheus-stack -n prometheus
        fi
        kubectl patch svc stable-kube-prometheus-sta-prometheus -n prometheus -p '{"spec": {"type": "LoadBalancer"}}'
        kubectl patch svc stable-grafana -n prometheus -p '{"spec": {"type": "LoadBalancer"}}'
      """
    }
  }
}
```

# ACCESS GRAFANA, PROMETHEUS, ARGOCD



# Apply the ArgoCd manifest... Change the type from Nodeport to LoadBalancer

```
stage("Configure ArgoCD") {
    steps {
        script {
            sh """
                # Install ArgoCD
                kubectl create namespace argocd || true
                kubectl apply -n argocd -f https://raw.githubusercontent.com/argoproj/argo-cd/stable/manifests/k8s
                kubectl patch svc argocd-server -n argocd -p '{"spec": {"type": "LoadBalancer"}}'
            """
        }
    }
}
```

# load balancer to Prometheus + Grafana

```
fi
kubectl patch svc stable-kube-prometheus -n prometheus -p '{"spec": {"type": "LoadBalancer"}}'
kubectl patch svc stable-grafana -n prometheus -p '{"spec": {"type": "LoadBalancer"}}'
}
}
```

# Go Back to Jenkins – Cp the deployment.yml file paste it on jenkins -----select Pipeline

The screenshot shows the Jenkins 'New Item' creation interface. At the top, there's a breadcrumb navigation: Dashboard > All > New Item. Below that is a 'New Item' header. A text input field is labeled 'Enter an item name' and contains the value 'deployment-pipeline'. Below the input field is a section titled 'Select an item type' with three options: 'Freestyle project', 'Maven project', and 'Pipeline'. The 'Pipeline' option is highlighted with a light gray background and has a detailed description below it: '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.' At the bottom of the form, there are two buttons: 'Create' and 'Cancel'.

← → ⌂ Not secure 100.24.54.83:8080/job/deployment-pipeline/configure

General

Advanced Project Options

Pipeline

Script ?

```
43
44
45
46 v
47 v
48 v
49
50
51
52
53
54
55
56
57
58
59
60 }
```

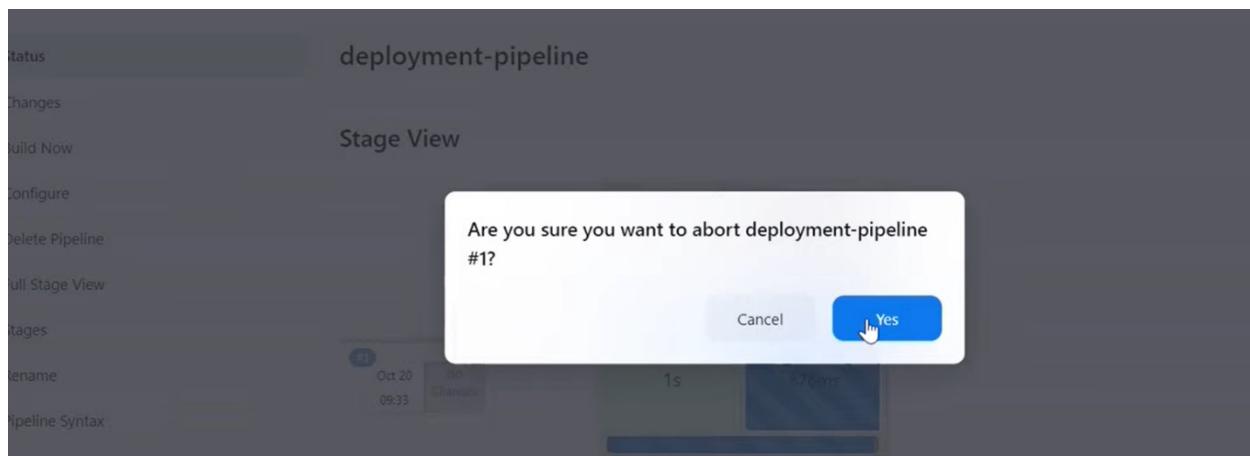
try samp

Use Groovy Sandbox ?

Pipeline Syntax

Save Apply

# Abort it



# Again Refresh it as we need Build with Parameters

Not secure 100.24.54.83:8080/job/deployment-pipeline/

Status deployment-pipeline

</> Changes

Build with Parameters

Configure

Delete Pipeline

Full Stage View

Average stage times:

Login to EKS	Config	Promethe	Grafan
1s	4s		

Stages

Rename

Pipeline Syntax

Stage View

#1 Oct 20 09:33 No Changes

Dashboard > deployment-pipeline >

Status

</> Changes

Build with Parameters

Configure

Delete Pipeline

Full Stage View

Stages

Rename

Pipeline Syntax

## Pipeline deployment-pipeline

This build requires parameters:

CLUSTER\_NAME

Enter your EKS cluster name

Build Cancel

# go to EKS

## # Click Compute

## # 2 nodes

Autoscaling group name	Capacity type	Subnets
<a href="#">eks-panda-node-2024102003505340600000001</a>	Spot	<a href="#">subnet-07f5bb285990787b0</a> <a href="#">subnet-09b8bc7c887ae415b</a>
<a href="#">4-f0c95406-51d0-704f-49a8-37b1834d56a1</a>	Desired size	Configure remote access to nodes
<a href="#">2 nodes</a>		

## # EC2 Created nodes help of Terraform Code

EC2 Dashboard X

Instances (3) Info Last updated less than a minute ago

Find Instance by attribute or tag (case-sensitive)

All states ▾

Instance state = running X Clear filters

Launch instances ▾

Name	Instance ID	Instance state	Instance type	Status check	Alarm status
panda-node	i-08541432b571e2e72	Running	t2.medium	...	u
panda-node	i-01c4f0308b9b1ddb1	Running	t2.medium	...	u
JENKINS-SERVER	i-0858872b9a500cf8e	Running	t2.medium	...	u

Select an instance

```
# jenkins
```

Dashboard > deployment-pipeline >

Status

## Pipeline deployment-pipeline

Changes

This build requires parameters:

Build with Parameters

CLUSTER\_NAME

Configure

Enter your EKS cluster name

Delete Pipeline

amazon-prime-cluster

Full Stage View

Build

Cancel

Stages

Rename

Pipeline Syntax

Status

deployment-pipeline

- > Changes
- > Build with Parameters
- > Configure
- > Delete Pipeline
- > Full Stage View
- > Stages
- > Rename
- > Pipeline Syntax

Stage View

Average stage times:

Login to EKS	Configure Prometheus & Grafana	Configure ArgoCD
1s	7s	190ms

#2 Oct 20 09:35 No Changes

Configure Prometheus & Grafana	Configure ArgoCD
1s	10s

#1 Oct 20 09:33 No Changes

Configure Prometheus & Grafana	Configure ArgoCD
1s	4s

Build History trend Filter...

Stage Logs (Configure Prometheus & Grafana)

Shell Script (self time 27s)

```
"prometheus-community" already exists with the same configuration, skipping
+ kubectl get namespace prometheus
+ kubectl create namespace prometheus
namespace/prometheus created
+ helm install stable/prometheus-community/kube-prometheus-stack -n prometheus
NAME: stable
LAST DEPLOYED: Sun Oct 20 04:05:26 2024
NAMESPACE: prometheus
STATUS: deployed
REVISION: 1
NOTES:
kube-prometheus-stack has been installed. Check its status by running:
  kubectl --namespace prometheus get pods -l "release=stable"

Visit https://github.com/prometheus-operator/kube-prometheus for instructions on how to create & configure Alertmanager and Prometheus instances using the Operator.
+ kubectl patch svc stable-kube-prometheus-sta-prometheus -n prometheus -p {"spec": {"type": "LoadBalancer"}}
service/stable-kube-prometheus-sta-prometheus patched
+ kubectl patch svc stable-grafana -n prometheus -p {"spec": {"type": "LoadBalancer"}}
service/stable-grafana patched
```

Shell Script -- # Install ArgoCD kubectl create namespace argocd || true kubectl apply -n argocd -f https://raw.githubusercontent.com/argoproj/argo-cd/stable/manifests/install.yaml kubectl patch svc argo-server -n argocd -p {"spec": {"type": "LoadBalancer"}} (self time 7s)

```
service/argocd-repo-server created
service/argocd-server created
service/argocd-server-metrics created
deployment.apps/argocd-applicationset-controller created
deployment.apps/argocd-dex-server created
deployment.apps/argocd-notifications-controller created
deployment.apps/argocd-redis created
deployment.apps/argocd-repo-server created
deployment.apps/argocd-server created
statefulset.apps/argocd-application-controller created
networkpolicy.networking.k8s.io/argocd-application-controller-network-policy created
networkpolicy.networking.k8s.io/argocd-applicationset-controller-network-policy created
networkpolicy.networking.k8s.io/argocd-dex-server-network-policy created
networkpolicy.networking.k8s.io/argocd-notifications-controller-network-policy created
networkpolicy.networking.k8s.io/argocd-redis-network-policy created
networkpolicy.networking.k8s.io/argocd-repo-server-network-policy created
+ kubectl patch svc argocd-server -n argocd -p {"spec": {"type": "LoadBalancer"}}
service/argocd-server patched
```

# We need URL of ArgoCd/Grafana / Prometheus??

# My\_repo - <https://github.com/Siddhartha082/AWS-DevOps-Final-Chapter-Project/blob/main/access.sh>

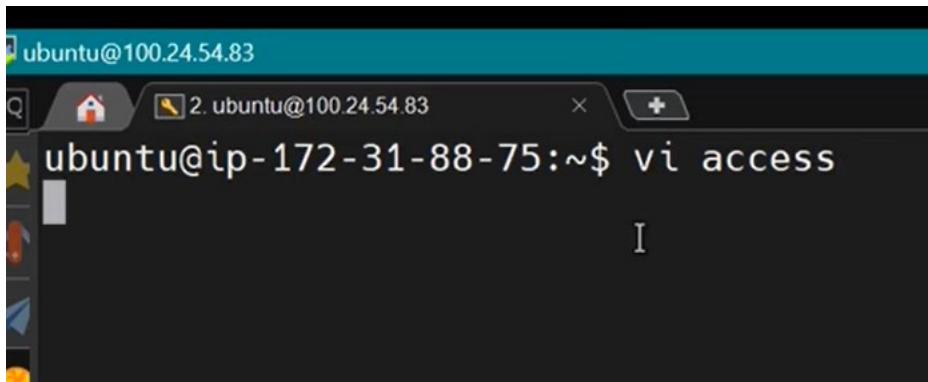
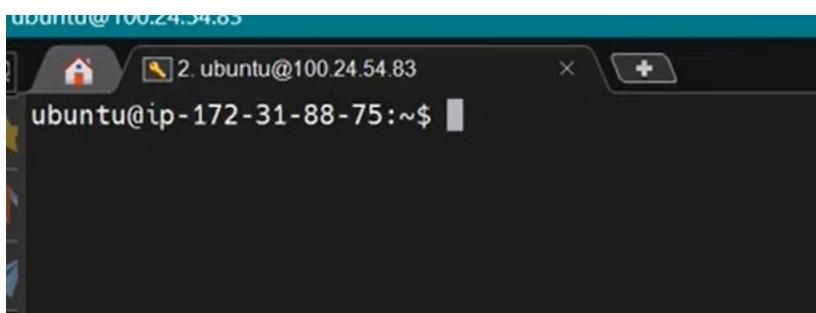
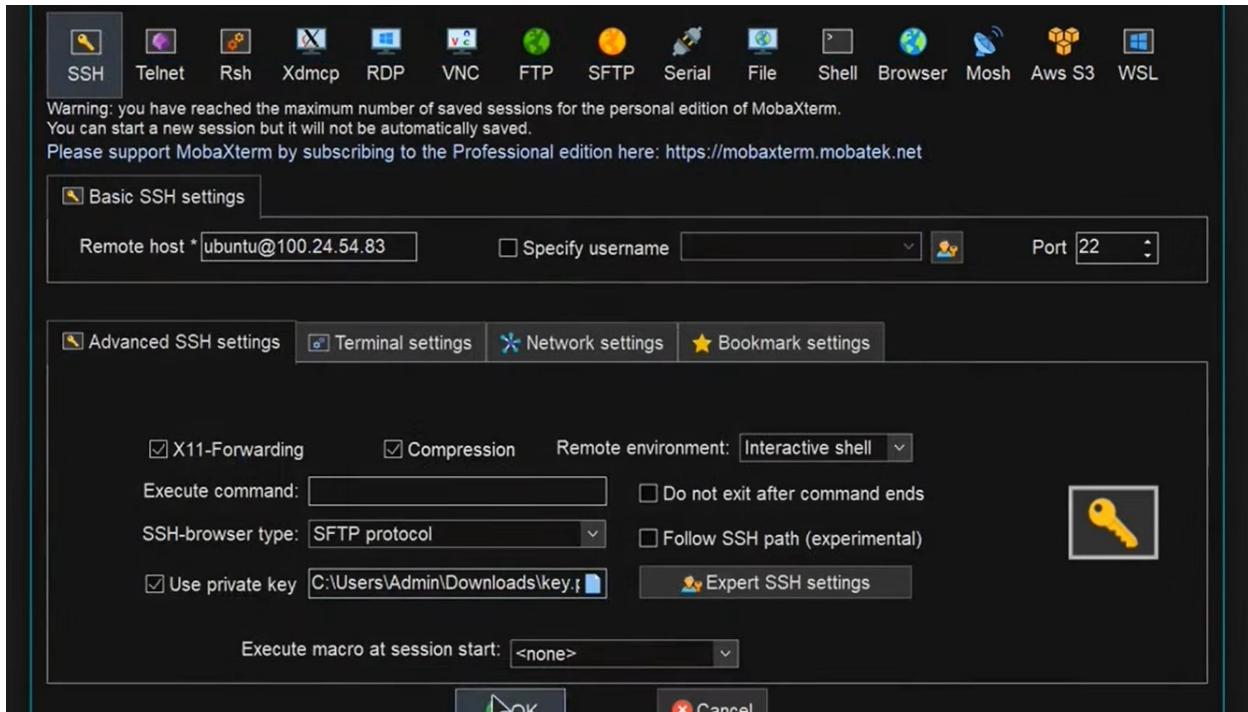
```
1 #!/bin/bash
2 # This script is used to get the argocd, prometheus & grafana urls & credentials
3
4 aws configure
5 aws eks update-kubeconfig --region "us-east-1" --name "amazon-prime-cluster"
6
7 # ArgoCD Access
8 argo_url=$(kubectl get svc -n argocd | grep argocd-server | awk '{print$4}' | head -n 1)
9 argo_initial_password=$(argocd admin initial-password -n argocd)
10
11 # ArgoCD Credentials
12 argo_user="admin"
13
14 argo_password=$(kubectl -n argocd get secret argocd-initial-admin-secret -o jsonpath=".data.password" | base64
15
16 # Prometheus and Grafana URLs and credentials
17 prometheus_url=$(kubectl get svc -n prometheus | grep stable-kube-prometheus-sta-prometheus | awk '{print $4}')
18 grafana_url=$(kubectl get svc -n prometheus | grep stable-grafana | awk '{print $4}')
19 grafana_user="admin"
20 grafana_password=$(kubectl get secret stable-grafana -n prometheus -o jsonpath=".data.admin-password" | base64
21
22 # Print or use these variables
23 echo "-----"
24 echo "ArgoCD URL: $argo_url"
25 echo "ArgoCD User: $argo_user"
```

Ln 1, Col 1 Spaces: 4 UTF-8 CRLF Shell Script Go Live

```
16 # Prometheus and Grafana URLs and credentials
17 prometheus_url=$(kubectl get svc -n prometheus | grep stable-kube-prometheus-sta-prometheus | awk '{print $4}')
18 grafana_url=$(kubectl get svc -n prometheus | grep stable-grafana | awk '{print $4}')
19 grafana_user="admin"
20 grafana_password=$(kubectl get secret stable-grafana -n prometheus -o jsonpath=".data.admin-password" | base64
21
22 # Print or use these variables
23 echo "-----"
24 echo "ArgoCD URL: $argo_url"
25 echo "ArgoCD User: $argo_user"
26 echo "ArgoCD Initial Password: $argo_initial_password" | head -n 1
27 echo
28 echo "Prometheus URL: $prometheus_url":9090
29 echo
30 echo "Grafana URL: $grafana_url"
31 echo "Grafana User: $grafana_user"
32 echo "Grafana Password: $grafana_password"
33 echo "-----"
34
35 # Run below commands
36 # chmod a+x access.sh
37 # ./access.sh
38
```

Ln 1, Col 1 Spaces: 4 UTF-8 CRLF Shell Script Go Live

# go to Jenkin Server



```
# past the above script
```

```
ubuntu@100.24.54.83:~$ vi access.sh
ubuntu@100.24.54.83:~$ chmod a+x access.sh
ubuntu@100.24.54.83:~$ ./access.sh
argo_password=$(kubectl -n argocd get secret argocd-initial-admin-secret -o jsonpath=".data.password" | base64 --decode)
# Prometheus and Grafana URLs and credentials
prometheus_url=$(kubectl get svc -n prometheus | grep stable-kube-prometheus-sta-prometheus | awk '{print $4}')
grafana_url=$(kubectl get svc -n prometheus | grep stable-grafana | awk '{print $4}')
grafana_user="admin"
grafana_password=$(kubectl get secret stable-grafana -n prometheus -o jsonpath=".data.admin-password" | base64 --decode)

# Print or use these variables
echo "-----"
echo "ArgoCD URL: $argo_url"
echo "ArgoCD User: $argo_user"
echo "ArgoCD Initial Password: $argo_initial_password" | head -n 1
echo
echo "Prometheus URL: $prometheus_url":9090
echo
echo "Grafana URL: $grafana_url"
echo "Grafana User: $grafana_user"
echo "Grafana Password: $grafana_password"
echo "-----"

# Run below commands
# chmod a+x access.sh
# ./access.sh
:wq
```

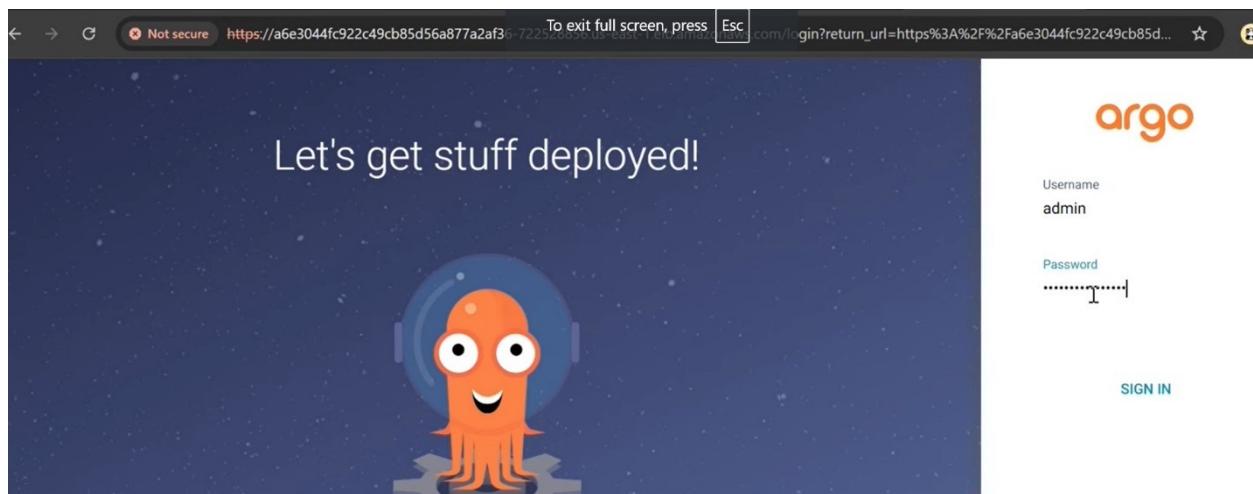
```
ubuntu@ip-172-31-88-75:~$ vi access
ubuntu@ip-172-31-88-75:~$ chmod a+x access
ubuntu@ip-172-31-88-75:~$ ./access
AWS Access Key ID [None]: AKIAUQCLMSLD5NUMX5FZ
AWS Secret Access Key [None]: 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
Default region name [None]: us-east-1
Default output format [None]:
Added new context arn:aws:eks:us-east-1:309395755719:cluster/amazon-prime-cluster to /home/ubuntu/.kube/config
-----
ArgoCD URL: a6e3044fc922c49cb85d56a877a2af36-722528856.us-east-1.elb.amazonaws.com
ArgoCD User: admin
ArgoCD Initial Password: zB4t8-Gb9jWckyp

Prometheus URL: a1fafd10db27240c998b40c176415f6c-1295826516.us-east-1.elb.amazonaws.com:9090

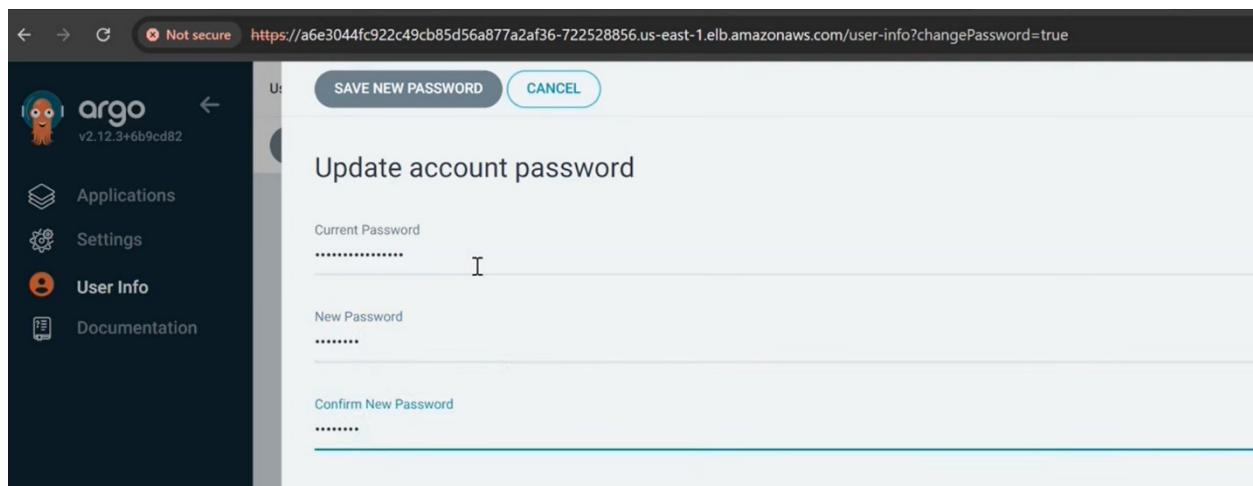
Grafana URL: aa35e7c2975504f238a44cf1734b04eb-1358210530.us-east-1.elb.amazonaws.com
Grafana User: admin
Grafana Password: prom-operator
-----
ubuntu@ip-172-31-88-75:~$
```

## # 1<sup>st</sup> ArgoCD access

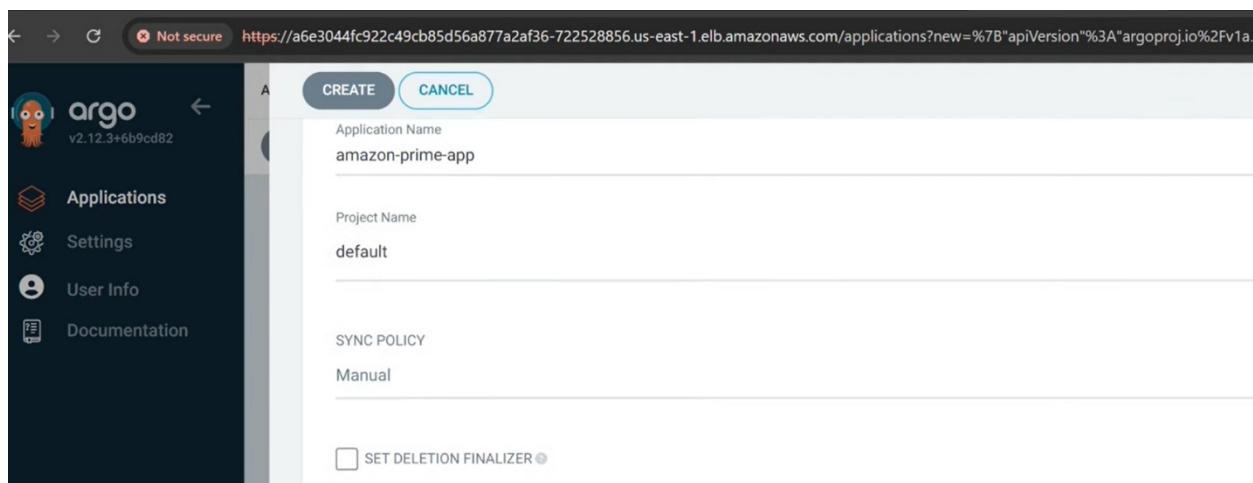
```
ArgoCD URL: a6e3044fc922c49cb85d56a877a2af36-722528856.us-east-1.elb.amazonaws.com
ArgoCD User: admin
ArgoCD Initial Password: zB4t8-Gb9jWckyp
```



## # Change password



## # Create App



## SOURCE

### Repository URL

<https://github.com/Siddhartha082/AWS-DevOps-Final-Chapter-Project>

---

### Revision

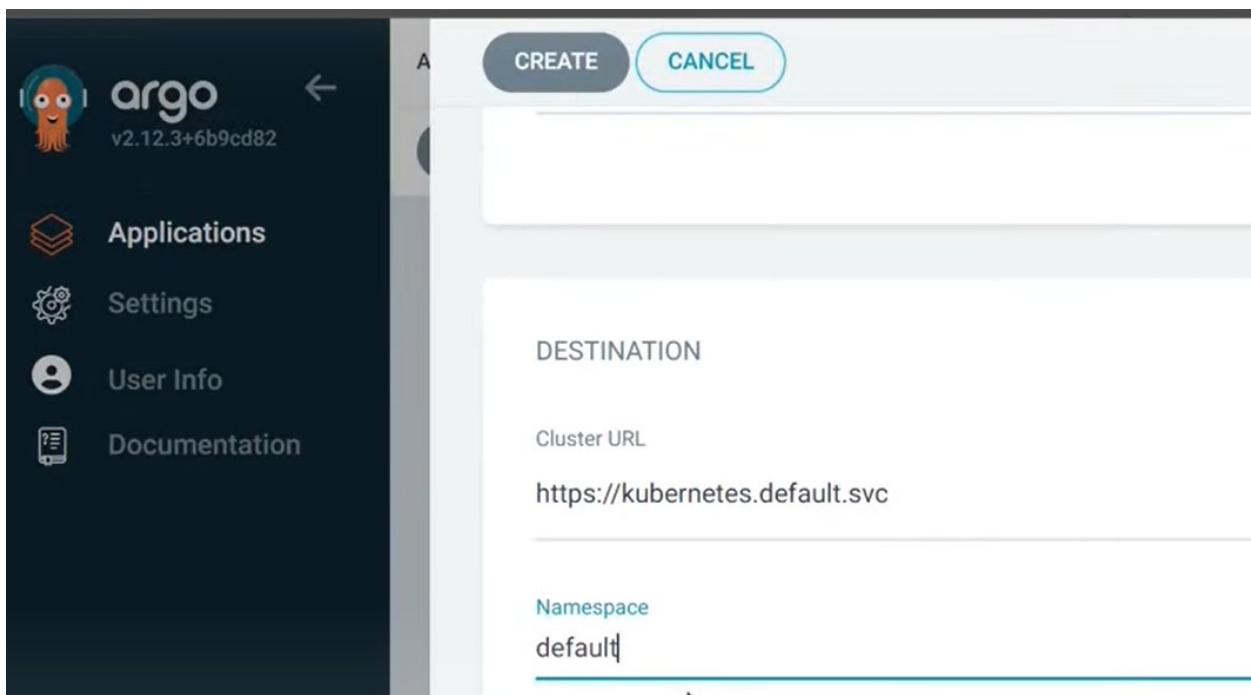
## HEAD

---

## K8s\_files

# path

```
apiVersion: apps/v1
kind: Deployment
metadata:
  name: pandacloud-app
  labels:
    app: pandacloud-app
spec:
  replicas: 1
  selector:
    matchLabels:
      app: pandacloud-app
  template:
    metadata:
      labels:
        app: pandacloud-app
    spec:
      containers:
        - name: pandacloud-container
          image: 851725192411.dkr.ecr.us-east-1.amazonaws.com/amazon-prime:latest
          ports:
            - containerPort: 3000
```



The screenshot shows the ArgoCD application list interface. On the left, there's a sidebar with the Argo logo and version v2.12.3+6b9cd82. The main area features a header with '+ NEW APP', 'SYNC APPS', 'REFRESH APPS', and a search bar. Below the header, a card displays an application named 'amazon-prime-app'. The card includes details such as Project: default, Labels:, Status: Progressing Synced, Repository: https://github.com/Siddhartha082/AWS-DevOps-Final-Cha, Target Revis..., Path: k8s\_files, Destination: in-cluster, Namespace: default, Created At: 10/20/2024 09:42:48 (a few seconds ago), and Last Sync: 10/20/2024 09:42:51 (a few seconds ago). At the bottom of the card are three buttons: 'SYNC', 'REFRESH', and 'DELETE'.

# Go back to jenkins server - ArgoCD manifest

```
ubuntu@ip-172-31-88-75:~$ kubectl get service -n argocd
NAME           TYPE        CLUSTER-IP   EXTERNAL-IP
argocd-applicationset-controller   ClusterIP  172.20.104.251 <none>
                                         7000/TCP,8080/TCP    7m17s
argocd-dex-server      ClusterIP  172.20.174.198 <none>
                                         5556/TCP,5557/TCP,5558/TCP 7m17s
argocd-metrics       ClusterIP  172.20.46.45 <none>
                                         8082/TCP    7m17s
argocd-notifications-controller-metrics ClusterIP  172.20.14.48 <none>
                                         9001/TCP    7m17s
argocd-redis          ClusterIP  172.20.47.3  <none>
                                         6379/TCP    7m17s
argocd-repo-server     ClusterIP  172.20.134.60 <none>
                                         8081/TCP,8084/TCP 7m17s
argocd-server          LoadBalancer 172.20.61.25 a6e3044fc922c49cb85d56a877a2af
22528856.us-east-1.elb.amazonaws.com  80:32509/TCP,443:30251/TCP 7m17s
argocd-server-metrics   ClusterIP  172.20.50.4  <none>
                                         8083/TCP    7m17s
ubuntu@ip-172-31-88-75:~$
```

# endpoint of Prometheus + Grafana

```
ubuntu@100.24.54.83:~$ 2. ubuntu@100.24.54.83
ubuntu@ip-172-31-88-75:~$ kubectl get service -n prometheus
NAME           TYPE        CLUSTER-IP   EXTERNAL-IP
alertmanager-operated   ClusterIP  None        <none>
                                         9093/TCP,9094/TCP,9094/UDP 7m44s
prometheus-operated    ClusterIP  None        <none>
                                         9090/TCP    7m43s
stable-grafana          LoadBalancer 172.20.47.38 aa35e7c2975504f238a44cf1734b04eb-1
358210530.us-east-1.elb.amazonaws.com  80:31308/TCP 7m49s
stable-kube-prometheus-sta-alertmanager ClusterIP  172.20.156.127 <none>
                                         9093/TCP,8080/TCP 7m49s
stable-kube-prometheus-sta-operator       ClusterIP  172.20.56.107 <none>
                                         443/TCP    7m49s
stable-kube-prometheus-sta-prometheus   LoadBalancer 172.20.8.245 a1fafd10db27240c998b40c176415f6c-1
295826516.us-east-1.elb.amazonaws.com   9090:31422/TCP,8080:31670/TCP 7m49s
stable-kube-state-metrics      ClusterIP  172.20.123.243 <none>
                                         8080/TCP    7m49s
stable-prometheus-node-exporter        ClusterIP  172.20.196.201 <none>
                                         9100/TCP    7m49s
ubuntu@ip-172-31-88-75:~$ ./access
AWS Access Key ID [*****X5FZ]:
AWS Secret Access Key [*****8aMU]:
Default region name [us-east-1]:
Default output format [None]:
Updated context arn:aws:eks:us-east-1:309395755719:cluster/amazon-prime-cluster in /home/ubuntu/.kube/config
-----
ArgoCD URL: a6e3044fc922c49cb85d56a877a2af36-722528856.us-east-1.elb.amazonaws.com
ArgoCD User: admin
ArgoCD Initial Password: zB4t8-Gb9jWckyp-
Prometheus URL: a1fd10db27240c998b40c176415f6c-1295826516.us-east-1.elb.amazonaws.com:9090
Grafana URL: aa35e7c2975504f238a44cf1734b04eb-1358210530.us-east-1.elb.amazonaws.com
Grafana User: admin
Grafana Password: prom-operator
-----
ubuntu@ip-172-31-88-75:~$
```

# Prometheus

The screenshot shows the Prometheus web interface. At the top, there's a navigation bar with tabs like 'Alerts', 'Graph', 'Status', and 'Help'. Below the navigation is a toolbar with checkboxes for 'Use local time', 'Enable query history', 'Enable autocomplete' (which is checked), 'Enable highlighting' (checked), and 'Enable linter' (checked). A search bar contains the placeholder 'Expression (press Shift+Enter for newlines)'. To the right of the search bar are icons for 'Table' (selected), 'Graph', and 'Execute'. Below the search bar is a section titled 'Evaluation time' with arrows for navigating between time points. The main area displays the message 'No data queried yet'. At the bottom left is a blue 'Add Panel' button.

# Argocd

The screenshot shows the Argo CD interface for the 'amazon-prime-app'. The left sidebar displays navigation links: Applications, Settings, User Info, and Documentation. The main content area shows the application is healthy and synced to HEAD (revision 1291809). A detailed tree view at the bottom shows the sync status for various components like svc, endpointslice, and rs.

## # Loadbalancer

```
ubuntu@ip-172-31-88-75:~$ kubectl get service
NAME          TYPE      CLUSTER-IP   EXTERNAL-IP
kubernetes    ClusterIP  172.20.0.1   <none>
pandacloud-app LoadBalancer  172.20.10.156  a183d4639631b4a3da83de4bea5e7980-1737361266.us-east-1.elb.amazonaws.com  3000:30261/TCP  2m
3s
```

```
ubuntu@ip-172-31-88-75:~$ kubectl get deployment
NAME           READY   UP-TO-DATE   AVAILABLE   AGE
pandacloud-app 1/1     1           1           2m48s
ubuntu@ip-172-31-88-75:~$
```

#### # Pod

```
ubuntu@ip-172-31-88-75:~$ kubectl get deployment
NAME           READY   UP-TO-DATE   AVAILABLE   AGE
pandacloud-app 1/1     1           1           2m48s
ubuntu@ip-172-31-88-75:~$ kubectl get pods
NAME                           READY   STATUS    RESTARTS   AGE
pandacloud-app-c4f87b4b5-hhr6x 1/1     Running   0          3m8s
ubuntu@ip-172-31-88-75:~$
```

#### # Graphically



#### # Endpoint

```
ubuntu@ip-172-31-88-75:~$ kubectl get service
NAME      TYPE      CLUSTER-IP   EXTERNAL-IP
kubernetes   ClusterIP  172.20.0.1   <none>
pandacloud-app   LoadBalancer  172.20.10.156  a183d4639631b4a3da83de4bea5e7980-1737361266.us-east-1.elb.amazonaws.com
ubuntu@ip-172-31-88-75:~$
```

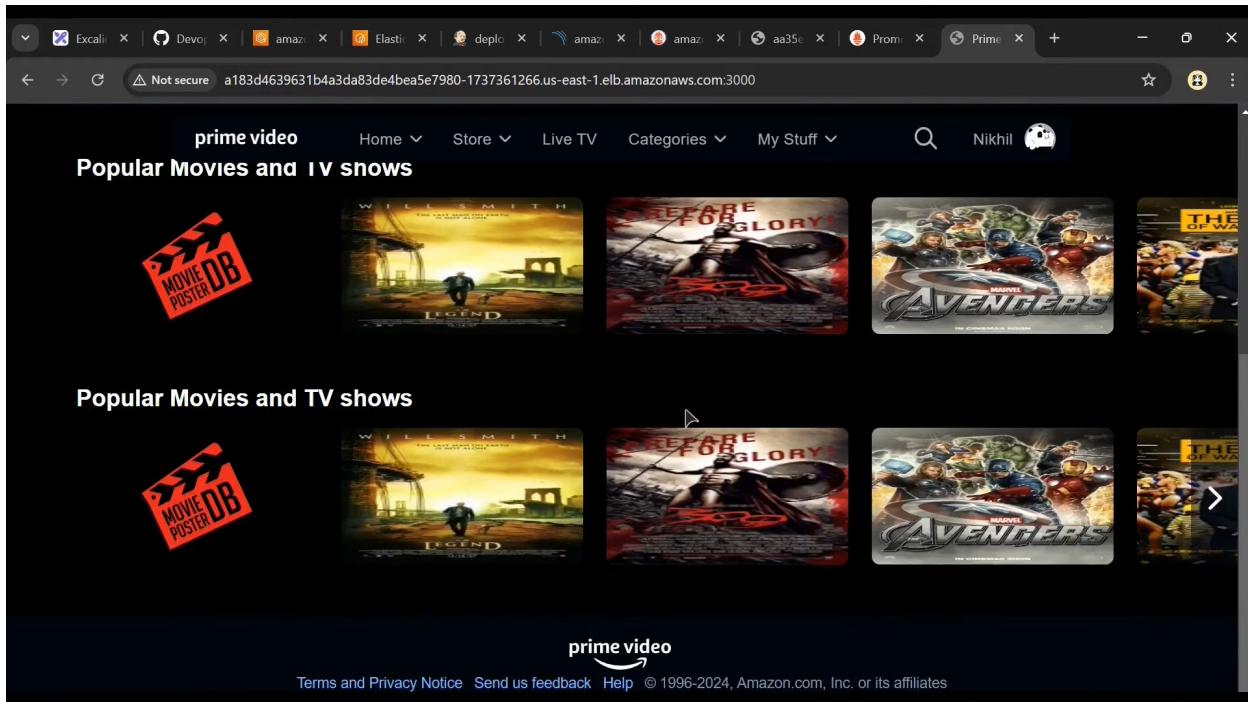
The screenshot shows the Argo UI interface. On the left, there's a sidebar with a logo, the name 'argo v2.12.3+6b9cd82', and navigation links for 'Applications', 'Settings', 'User Info', and 'Documentation'. Below these are dropdown menus for 'NAME', 'KINDS', and 'SYNC STATUS' (Synced: 2, OutOfSync: 0). The main area displays a service named 'pandacloud-app' with the following details:

KIND	Service
NAME	pandacloud-app
NAMESPACE	default
CREATED AT	10/20/2024 09:42:51 (4 minutes ago)
TYPE	LoadBalancer
HOSTNAMES	a183d4639631b4a3da83de4bea5e7980-1737361266.us-east-1.elb.amazonaws.com
STATUS	Synced
HEALTH	Healthy

At the top right of the main area are 'SYNC' and 'DELETE' buttons.

# Run – Amazon Prime -app @ Port 3000

The screenshot shows a web browser window displaying the Amazon Prime Video homepage. The URL in the address bar is 'a183d4639631b4a3da83de4bea5e7980-1737361266.us-east-1.elb.amazonaws.com:3000'. The page features a large movie poster for 'AVATAR' with the release date '18 Dec 2009' and rating 'PG-13'. Below the poster is a 'Play' button. The page also includes a search bar, user profile 'Nikhil', and sections for 'Popular Movies and TV shows' featuring 'PRETEND FOR GLORY!' and 'Avengers: Endgame'.



# Route53 Server .. where Big Line of .com .. can be shorten using DNS naming

ArgoCD URL: a6e3044fc922c49cb85d56a877a2af36-722528856.us-east-1.elb.amazonaws.com  
 ArgoCD User: admin  
 ArgoCD Initial Password: zB4t8-Gb9jWckyp-

Prometheus URL: a1fafd10db27240c998b40c176415f6c-1295826516.us-east-1.elb.amazonaws.com:9090

Grafana URL: aa35e7c2975504f238a44cf1734b04eb-1358210530.us-east-1.elb.amazonaws.com  
 Grafana User: admin  
 Grafana Password: prom-operator

aws Services Search route53 X

Search results for 'route53'

**Services**

- Route 53 ★ Scalable DNS and Domain Name Registration
- Route 53 Resolver Resolve DNS queries in your Amazon VPC and on-premises network.
- Amazon Location Service ★ Securely and easily add location data to applications.

**Features**

- Clusters
- Amazon EKS Anywhere Enterprise Subscriptions
- Related services Amazon ECR AWS Batch
- Console settings

**Route 53**

**Route 53 Dashboard**

**DNS management**

**Traffic management**  
A visual tool that lets you easily create policies for multiple endpoints in complex configurations.

**Availability monitoring**  
Health checks monitor your applications and web resources, and direct DNS queries to healthy resources.

**Domain registration**

**Hosted zone**

**Create policy**

**Create health check**

**Domain**

**Registered domains**

## # Create a URL

**aws Services Search [Alt+S]**

**Route 53**

**Route 53 > Hosted zones > pandacloud.click**

**pandacloud.click** **Info**

**Public**

**Delete zone** **Test record** **Configure query logging**

**Hosted zone details** **Edit hosted zone**

**Records (3)** **DNSSEC signing** **Hosted zone tags**

**Records (3) Info**

**Create record**

Automatic mode is the current search behavior optimized for best filter results. [To change modes go to settings.](#)

## # Create Records

EC2 S3 VPC RDS EFS Simple Notification Service CloudWatch CloudTrail Trusted Advisor ElastiCache Route 53 Support CloudForms

**Record 1**

Record name **Info** argocd.pandacloud.click Record type **Info** A – Routes traffic to an IPv4 address and some AWS resources

Keep blank to create a record for the root domain.

Alias

Route traffic to **Info** Alias to Application and Classic Load Balancer

US East (N. Virginia)

dualstack.a6e3044fc922c49cb85d56a877a2af36-722528856.us-east-1.elb.amazonaws.com

Alias hosted zone ID: Z35SXDOTRQ7X7K

Routing policy **Info** Simple routing Evaluate target health  Yes

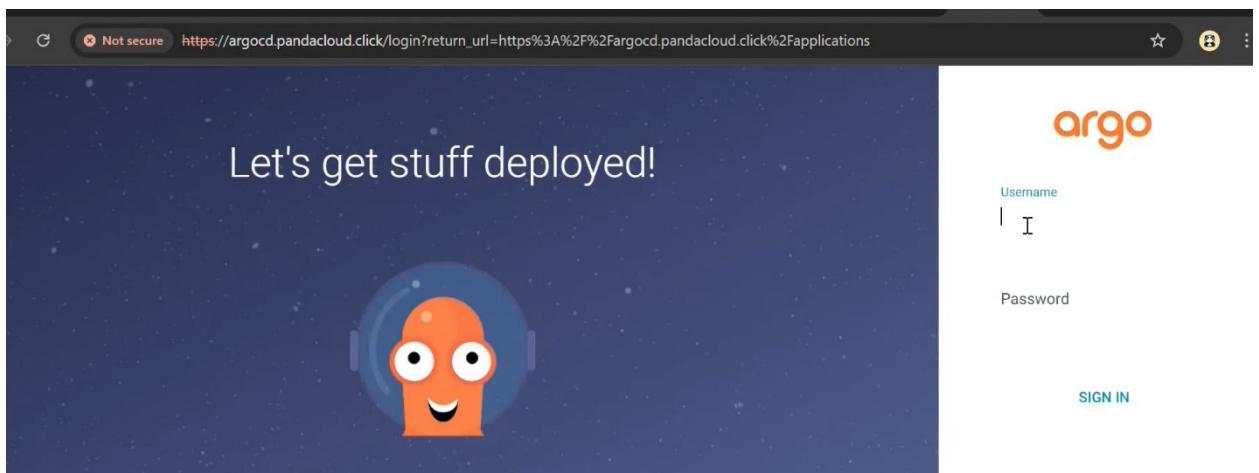
**Route 53**

Record for pandacloud.click was successfully created. Route 53 propagates your changes to all of the Route 53 authoritative DNS servers within 60 seconds. Use "View status" button to check propagation status.

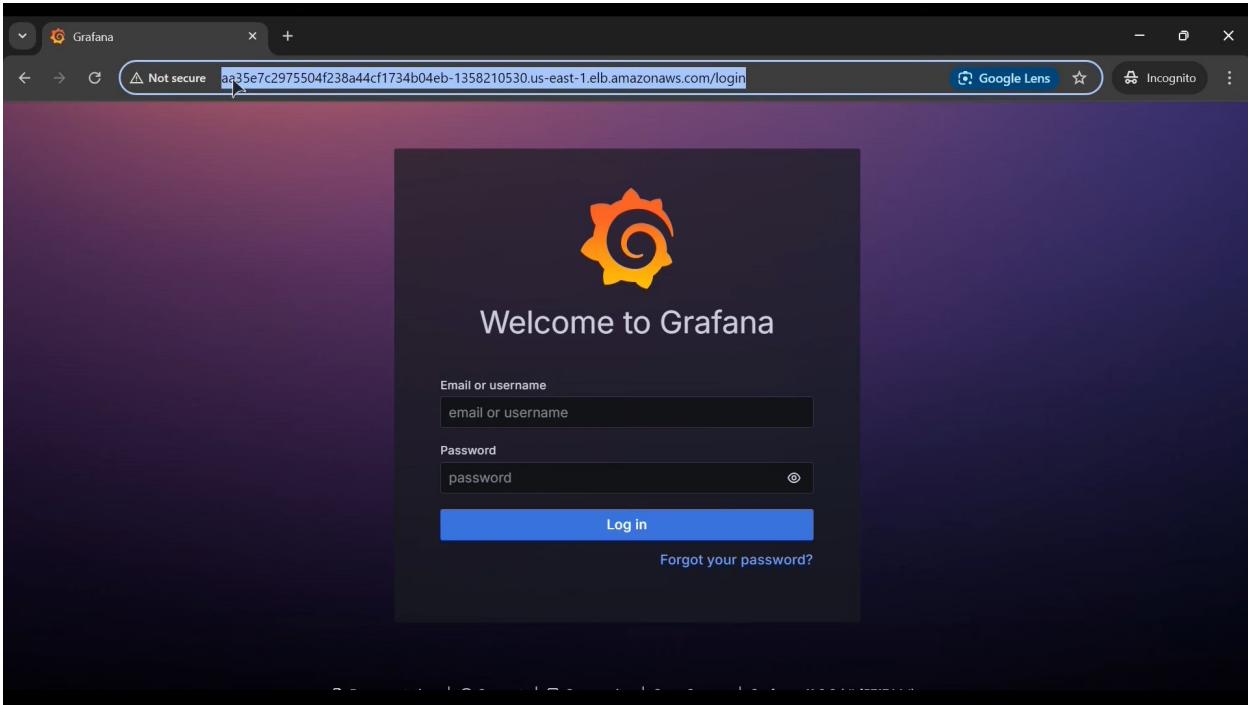
**Record details**

Record name: argocd.pandacloud.click  
Record type: A  
Value: dualstack.a6e3044fc922c49cb85d56a877a2af36-722528856.us-east-1.elb.amazonaws.com  
Alias: Yes  
TTL (seconds): -  
Routing policy: Simple

Record ...	Type	Routing policy	Differ...	Alias
pandaclo...	NS	Simple	-	No
pandaclo...	SOA	Simple	-	No
<b>argocd.pa...</b>	<b>A</b>	<b>Simple</b>	-	<b>Yes</b>
www.pan...	A	Simple	-	Yes



```
# check for Prometheus + Grafana status
```



```
Grafana URL: aa35e7c2975504f238a44cf1734b04eb-1358210530.us-east-1.elb.amazonaws.com  
Grafana User: admin  
Grafana Password: prom-operator
```

```
# Note if grafana does not work ... type http://URL...
```

Welcome to Grafana

Need help? [Documentation](#) [Tutorials](#) [Community](#) [Public Slack](#)

**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

**COMPLETE**  
Create your first dashboard

Remove this panel

```
# Create Custom Dashboard
```

Welcome to Grafana

Need help? [Documentation](#) [Tutorials](#) [Community](#) [Public Slack](#)

**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 ↗

[Remove this panel](#)

## # Click on import Dashboard

New dashboard - Dashboards

Not secure aa35e7c2975504f238a44cf1734b04eb-1358210530.us-east-1.elb.amazonaws.com/dashboard/new?orgId=1

Home > Dashboards > New dashboard

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

# Import dashboard

Import dashboard from file or Grafana.com



Upload dashboard JSON file

Drag and drop here or click to browse  
Accepted file types: .json, .txt

Find and import dashboards for common applications at [grafana.com/dashboards](https://grafana.com/dashboards)

1860

Load

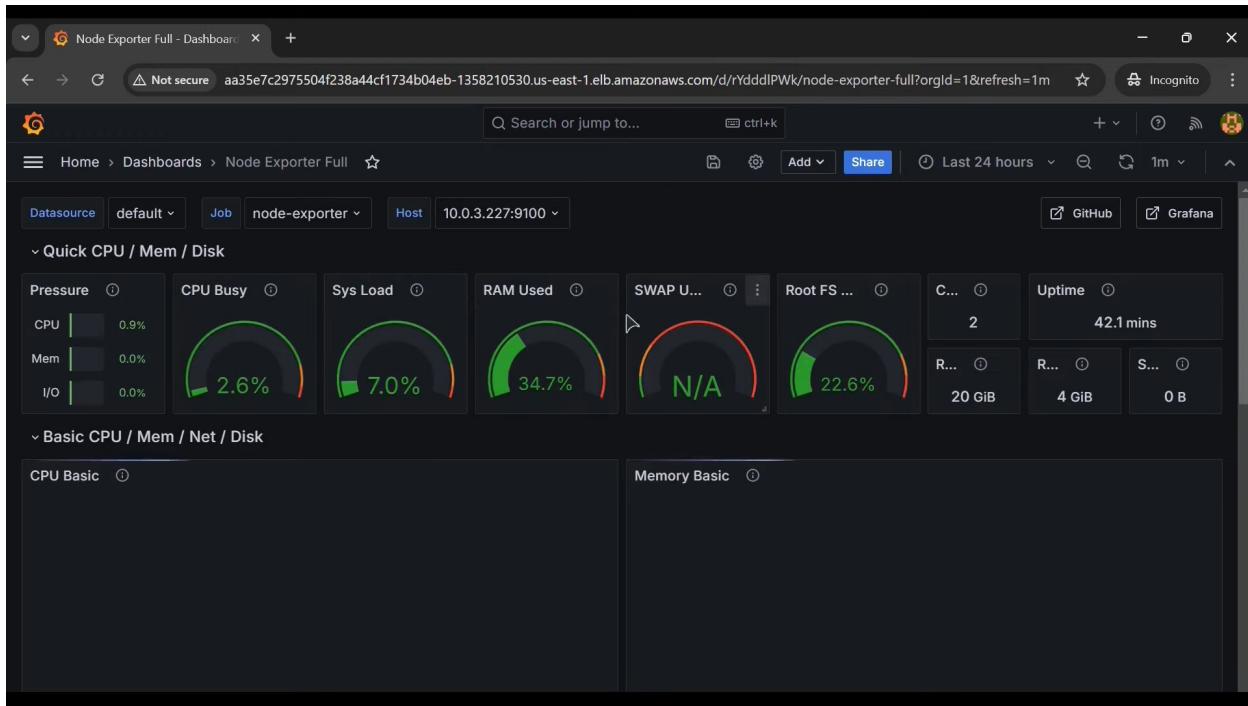


Import via dashboard JSON model

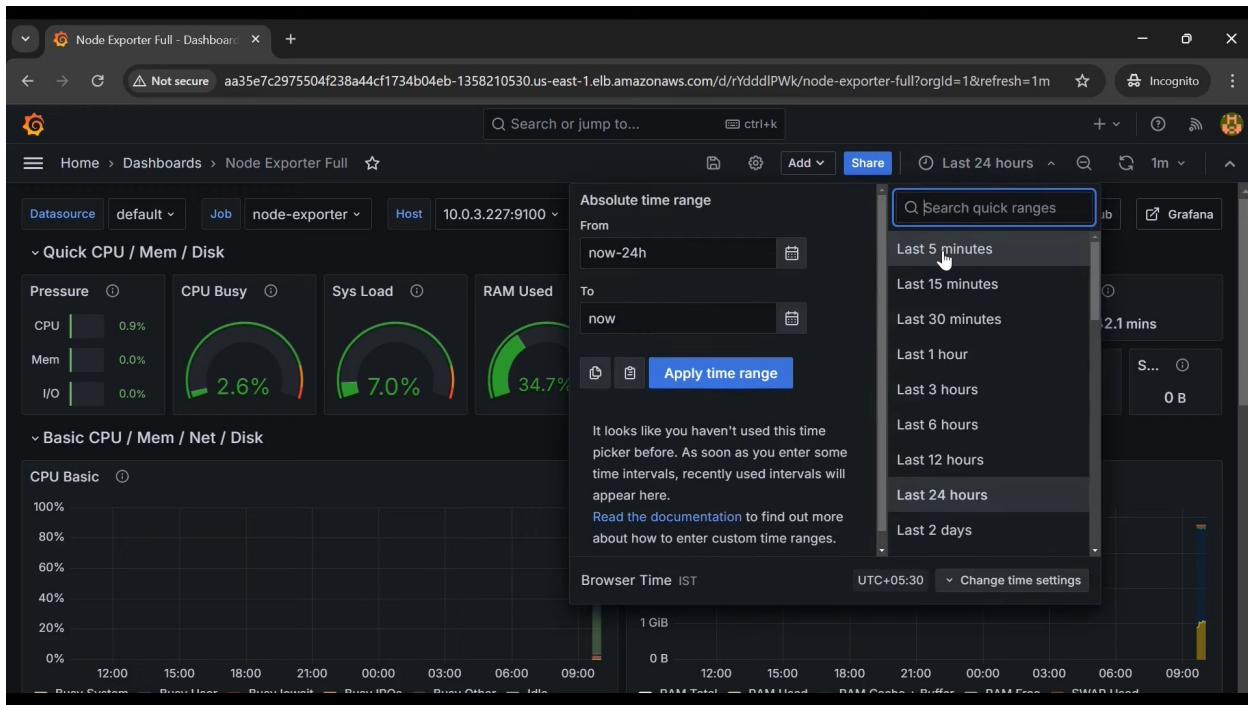
```
{  
  "title": "Example - Repeating Dictionary variables",  
  "uid": "_0HnEoN4z",  
  "panels": [...]}
```

The screenshot shows the 'Import dashboard' dialog in a browser. The URL in the address bar is [aa35e7c2975504f238a44cf1734b04eb-1358210530.us-east-1.elb.amazonaws.com/dashboard/import](https://aa35e7c2975504f238a44cf1734b04eb-1358210530.us-east-1.elb.amazonaws.com/dashboard/import). The page displays dashboard metadata: 'Published by rfmoz' and 'Updated on 2024-05-22 21:37:35'. Under 'Options', there are fields for 'Name' (set to 'Node Exporter Full') and 'Folder' (set to 'Dashboards'). A 'Unique identifier (UID)' section explains that the UID allows consistent URLs for dashboards. The current UID is 'rYdddIPWk', with a 'Change uid' button. A dropdown menu for 'Prometheus' shows 'Prometheus' selected. At the bottom are 'Import' and 'Cancel' buttons.

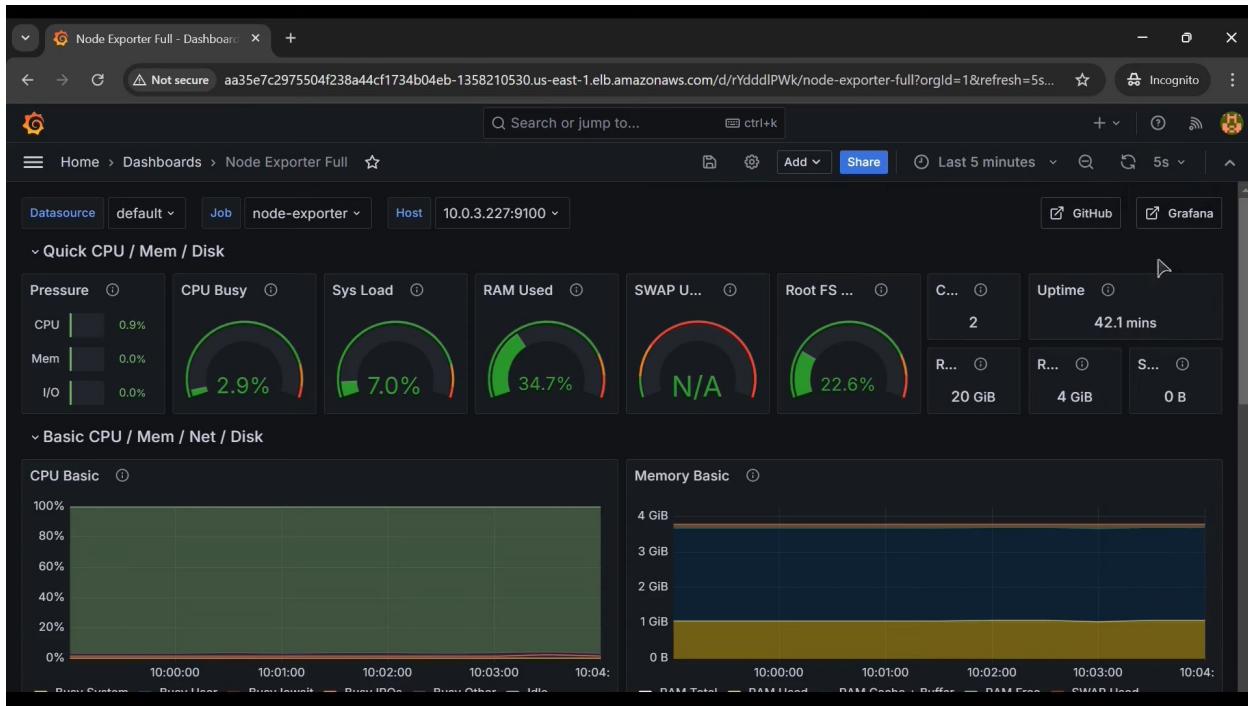
```
# check CPU/RAM Utilization
```



# last 5 mins



# Refresh 5 sec



# this is how we Monitor Grafana Dashboard to check status of Pods

# cleanup

---

# CLEANUP PIPELINE



# AWS KMS will be created default in EKS Cluster ... We need to remove it

---

**Amazon Elastic Container Registry**

Private registry

Repositories

**Features & Settings**

- Permissions →
- Pull through cache
- Replication
- Repository creation templates
- Scanning

Public registry

Repositories

Settings

**Amazon ECR > Private registry**

## Private registry Info

**Account settings**

**Permissions** Info Edit

loading permissions

**Feature settings**

**Pull through cache** Info Edit

loading pull through cache

**Replication** Info Edit

loading replication

**Amazon Elastic Container Registry**

Private registry

**Repositories**

Features & Settings

Public registry

Repositories

Settings

ECR public gallery ↗

Amazon ECS ↗

Amazon EKS ↗

**Amazon ECR > Private registry > Repositories**

## Private repositories

Create repository

Repositories (1)		<span>C</span>	<span>View push commands</span>	<span>Delete</span>	<span>Actions ▾</span>
Repository name	URI	Created at	Tag immutability	Encryption type	
amazon-prime	1.amazonaws.com/amazon-prime	October 20, 2024, 08:56:09 (UTC+05.5)	Mutable	AES-256	

## # AWS KMS Key –

**Key Management Service (KMS)**

AWS managed keys

**Customer managed keys**

Custom key stores

AWS CloudHSM key stores

External key stores

**KMS > Customer managed keys**

## Customer managed keys (8)

Key actions ▾ Create key

Filter keys by properties or tags

<input type="checkbox"/>	Aliases	Key ID	Status	Key type	Key spec ⓘ
<input type="checkbox"/>	-	42de03f8-3066...	Not authorized	Not authorized	Not authorized
<input type="checkbox"/>	eks/amazon-prime-cluster	4c5eeddb-04a4...	Not authorized	Not authorized	Not authorized
<input type="checkbox"/>	-	4da97536-ebf6...	Not authorized	Not authorized	Not authorized
<input type="checkbox"/>	-	4ddc229a-12b7...	Not authorized	Not authorized	Not authorized
<input type="checkbox"/>	-	99a8cf52-a134...	Not authorized	Not authorized	Not authorized
<input type="checkbox"/>	-	a9eb5f2e-48a8...	Not authorized	Not authorized	Not authorized
<input type="checkbox"/>	-	ad5f7c68-4162...	Not authorized	Not authorized	Not authorized

## # cleanup pipeline

```
1 pipeline {
2     agent any
3
4     environment {
5         KUBECTL = '/usr/local/bin/kubectl'
6     }
7
8     parameters {
9         string(name: 'CLUSTER_NAME', defaultValue: 'amazon-prime-cluster', description: 'Enter your EKS cluster name')
10    }
11
12    stages {
13
14        stage("Login to EKS") {
15            steps {
16                script {
17                    withCredentials([string(credentialsId: 'access-key', variable: 'AWS_ACCESS_KEY'),
18                                   string(credentialsId: 'secret-key', variable: 'AWS_SECRET_KEY')]) {
19                        sh "aws eks --region us-east-1 update-kubeconfig --name ${params.CLUSTER_NAME}"
20                    }
21                }
22            }
23        }
24
25        stage('Cleanup K8s Resources') {
26            steps {
27                script {
28                    // Step 1: Delete services and deployments
29                    sh 'kubectl delete svc kubernetes || true'
30                    sh 'kubectl delete deploy pandacloud-app || true'
31                    sh 'kubectl delete svc pandacloud-app || true'
32
33                    // Step 2: Delete ArgoCD installation and namespace
34                    sh 'kubectl delete -n argocd -f https://raw.githubusercontent.com/argoproj/argo-cd/stable/manifests/install.yaml || true'
35                    sh 'kubectl delete namespace argocd || true'
36
37                    // Step 3: List and uninstall Helm releases in prometheus namespace
38                    sh 'helm list -n prometheus || true'
39                    sh 'helm uninstall kube-stack -n prometheus || true'
40
41                    // Step 4: Delete prometheus namespace
42                    sh 'kubectl delete namespace prometheus || true'
43
44                    // Step 5: Remove Helm repositories
45                    sh 'helm repo remove stable || true'
46                    sh 'helm repo remove prometheus-community || true'
47                }
48            }
49        }
50
51        stage('Delete ECR Repository and KMS Keys') {
52            steps {
53                script {
54                    // Step 1: Delete ECR Repository
55                    sh ''
56                    aws ecr delete-repository --repository-name amazon-prime --region us-east-1 --force
57
58
59                    // Step 2: Delete KMS Keys
60                    sh ''
61
62                    for key in $(aws kms list-keys --region us-east-1 --query "Keys[*].KeyId" --output text); do
63                        aws kms disable-key --key-id $key --region us-east-1
64                        aws kms schedule-key-deletion --key-id $key --pending-window-in-days 7 --region us-east-1
65                    done
66
67                }
68            }
69        }
70    }
71 }
```

## # Jenkins

## New Item

Enter an item name

cleanup-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.

**# this will Clean up existing + default Keys**

**Script**

```

54 // Step 1: Delete ECR Repository
55 sh """
56 aws ecr delete-repository --repository-name amazon-prime --region us-east-1 --force
57 """
58
59 // Step 2: Delete KMS Keys
60 sh """
61 for key in $(aws kms list-keys --region us-east-1 --query "Keys[*].KeyId" --output text); do
62     aws kms disable-key --key-id $key --region us-east-1
63     aws kms schedule-key-deletion --key-id $key --pending-window-in-days 7 --region us-east-1
64 done| I
65 """
66 }
67 }
68
69
70

```

## Jenkins

## Status

**cleanup-pipeline**

Changes

Build scheduled

Build Now

Configure

Delete Pipeline

Full Stage View

**Stage View**

No data available. This Pipeline has not yet run.

**Permalinks**

Stages

Rename

Pipeline Syntax

```
← → ⌂ Not secure 100.24.54.83:8080/job/cleanup-pipeline/  
Stage Logs (Cleanup K8s Resources)  
Shell Script -- kubectl delete svc kubernetes || true (self time 836ms)  
Shell Script -- kubectl delete deploy pandacloud-app || true (self time 836ms)
```

+ kubectl delete deploy pandacloud-app  
deployment.apps "pandacloud-app" deleted

```
Shell Script -- kubectl delete svc pandacloud-app || true (self time 27s)  
Shell Script -- kubectl delete -n argocd -f https://raw.githubusercontent.com/argoproj/argo-cd/stable/manifests/install.yaml || true (self time 19s)
```

Delete Pipeline

Login to EKS

Cleanup K8s  
Resources

```
Shell Script -- kubectl delete svc kubernetes || true (self time 836ms)  
Shell Script -- kubectl delete deploy pandacloud-app || true (self time 836ms)  
Shell Script -- kubectl delete svc pandacloud-app || true (self time 27s)  
Shell Script -- kubectl delete -n argocd -f https://raw.githubusercontent.com/argoproj/argo-cd/stable/manifests/install.yaml || true (self time 30s)  
Shell Script -- kubectl delete namespace argocd || true (self time 7s)  
Shell Script -- helm list -n prometheus || true (self time 836ms)  
Shell Script -- helm uninstall kube-stack -n prometheus || true (self time 834ms)  
Shell Script -- kubectl delete namespace prometheus || true (self time 35s)
```

+ kubectl delete namespace prometheus  
namespace "prometheus" deleted

# ECR Repo cleaned up

Amazon Elastic Container Registry

Private registry

Repositories

Features & Settings

Public registry

Repositories

Settings

ECR public gallery

Amazon ECS

Amazon ECR > Private registry > Repositories

Private repositories

Create repository

Repositories

Search by repository substring

Repository name

URI

Created at

Tag immutability

Encryption type

No repositories

No repositories were found

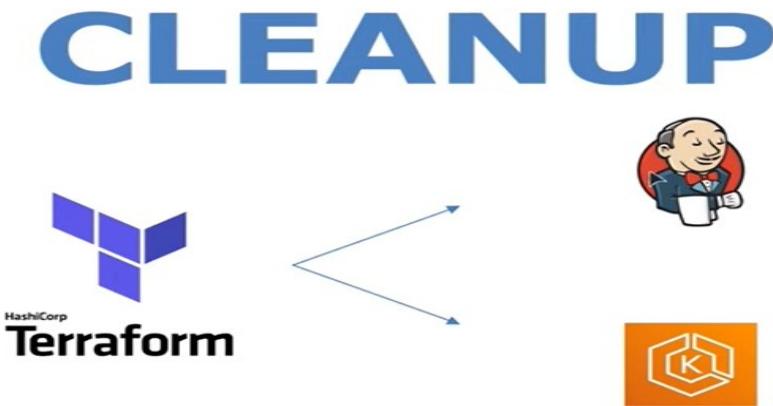
# EKS Still Running

The screenshot shows the AWS EKS service page. On the left, there's a sidebar for 'Amazon Elastic Kubernetes Service' with sections for 'Clusters', 'Amazon EKS Anywhere', and 'Related services'. The main area is titled 'Clusters (1) Info' and shows a table with one row. The row details the cluster name 'amazon-prime-cluster' as 'Active', running 'Kubernetes version 1.31', with 'Support period Standard support until November 26, 2025', and an 'Upgrade policy' of 'Extended'.

# Even EC2 – server stil running

The screenshot shows the AWS EC2 service page. The left sidebar includes 'EC2 Dashboard', 'EC2 Global View', 'Events', and sections for 'Instances' (with 'Instances', 'Instance Types', 'Launch Templates', and 'Spot Requests'). The main table lists three instances: 'panda-node' (ID i-08541432b571e2e72), 'panda-node' (ID i-01c4f0308b9b1ddb1), and 'JENKINS-SERVER' (ID i-0858872b9a500cf8e). All three instances are shown as 'Running' in the 'Instance state' column.

# 2<sup>nd</sup> part Clean up Jenkins + EKS using Terraform Destroy



# EC2 instances

```
1 module "eks" {
2   source  = "terraform-aws-modules/eks/aws"
3   version = "19.15.1"
4
5 SERVER-SSH-ACCESS = "ubuntu@100.24.54.83"
6 PS>terraform destroy --auto-approve
7 aws_security_group.my-sg: Refreshing state... [id=sg-0376c4a89f26497fd]
8 aws_instance.my-ec2: Refreshing state... [id=i-0858872b9a500cf8e]
```

## # EKS Server

The screenshot shows a Visual Studio Code interface with several panes:

- Left Sidebar:** Shows the project structure with files like `k8s_files`, `pipeline_script`, `public`, `src`, `terraform_code`, `ec2_server`, `eks_code`, `.terraform`, `.terraform.lockfile`, `eks.tf` (selected), `provider.tf`, and `terraform.tfstate`.
- Top Status Bar:** Displays the current file path: `terraform_code > eks_code > eks.tf > eks module "eks"`.
- Terminal:** Shows the command `PS>terraform destroy --auto-approve` being run.
- Output:** Displays the results of the Terraform destroy operation, showing the destruction of various resources over time (e.g., ingress rules, IAM roles, security groups, VPCs).
- Bottom Status Bar:** Shows the message `Destroy complete! Resources: 62 destroyed.`

## # AWS EC2 – no instances

Screenshot of the AWS EC2 Instances page showing no matching instances found.

The search bar shows "Instance state = running".

The message "No matching instances found" is displayed.

# No Cluster EKS

Screenshot of the AWS Amazon Elastic Kubernetes Service (EKS) Clusters page showing no clusters.

The search bar shows "Filter clusters".

The message "No clusters" and "You do not have any clusters." is displayed.

A "Create cluster" button is present.

# THANK YOU !



