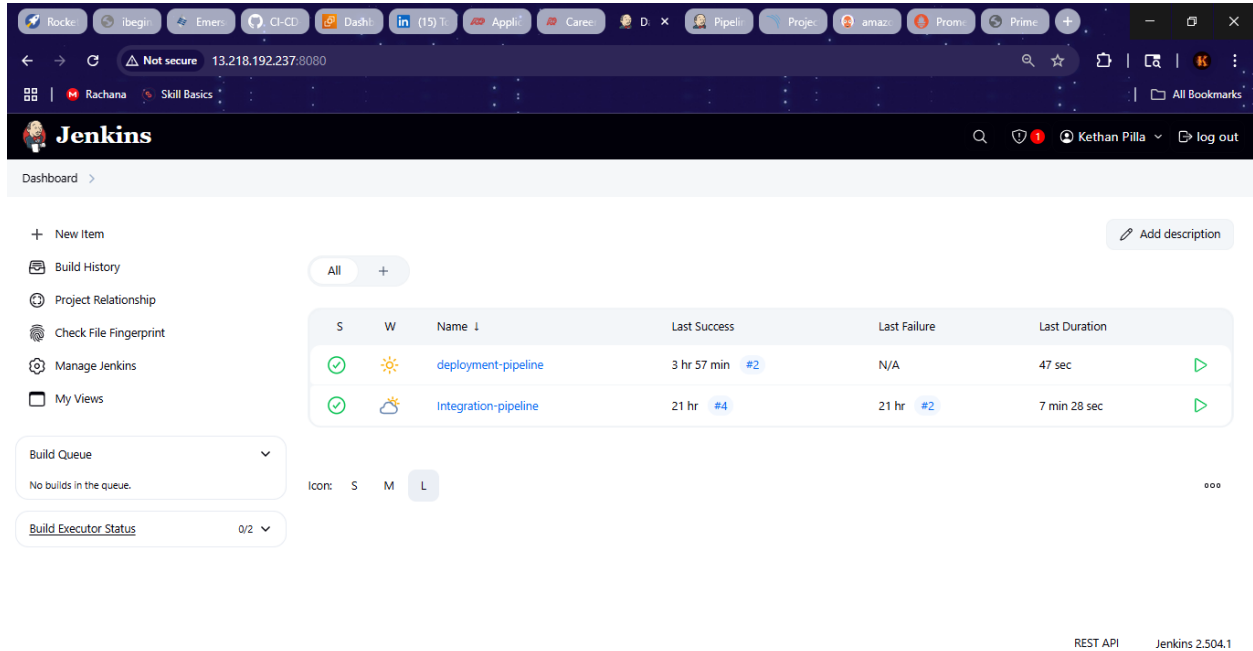


Jenkins Dashboard



The screenshot shows the Jenkins Dashboard interface. At the top, there's a navigation bar with the Jenkins logo and a search bar. Below the navigation bar, there's a sidebar with links to 'New Item', 'Build History', 'Project Relationship', 'Check File Fingerprint', 'Manage Jenkins', and 'My Views'. The main content area displays a table of pipelines. The table has columns for 'S' (Status), 'W' (Webhook), 'Name', 'Last Success', 'Last Failure', and 'Last Duration'. Two pipelines are listed: 'deployment-pipeline' and 'integration-pipeline'. The 'integration-pipeline' is highlighted. Below the table, there's a 'Build Queue' section showing 'No builds in the queue.' and a 'Build Executor Status' section showing '0/2'.

S	W	Name	Last Success	Last Failure	Last Duration
✓	☀	deployment-pipeline	3 hr 57 min #2	N/A	47 sec
✓	☁	integration-pipeline	21 hr #4	21 hr #2	7 min 28 sec

Build Queue: No builds in the queue.

Build Executor Status: 0/2

REST API Jenkins 2.504.1

Integration pipeline script

pipeline {

agent any

parameters {

string(name: 'ECR_REPO_NAME', defaultValue: 'amazon-prime', description: 'Enter repository name')

string(name: 'AWS_ACCOUNT_ID', defaultValue: '123456789012', description: 'Enter AWS Account ID') // Added missing quote

}

tools {

jdk 'JDK'

```
nodejs 'NodeJS'
}

environment {
    SCANNER_HOME = tool 'SonarQube Scanner'
}

stages {
    stage('1. Git Checkout') {
        steps {
            git branch: 'main', url: 'https://github.com/pandacloud1/DevopsProject2.git'
        }
    }

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

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

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

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

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

```
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.ECR_REPO_NAME} --
region us-east-1 || \
            aws ecr create-repository --repository-name ${params.ECR_REPO_NAME} --
region us-east-1
        """
    }
}

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
            """
        }
    }
}

```

```
        """"
    }
}
}
```

```
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
            """
        }
    }
}
```

```
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
```

```

}

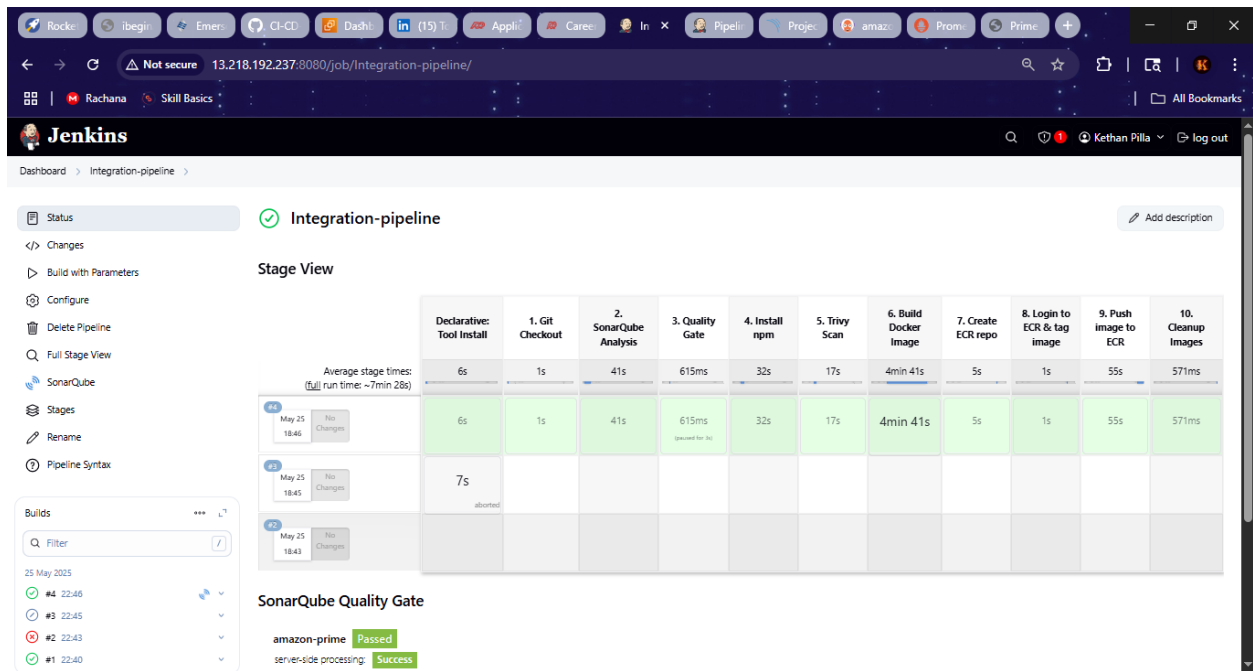
}

}

}

```

Integration pipeline



Deployment pipeline script

```

pipeline {
    agent any

    environment {
        KUBECTL = '/usr/local/bin/kubectl'
    }
}

```

```

parameters {
    string(name: 'CLUSTER_NAME', defaultValue: 'amazon-prime-cluster', description:
'Enter your EKS cluster name')
}

stages {
    stage("Login to EKS") {
        steps {
            script {
                withCredentials([string(credentialsId: 'access-key', variable: 'AWS_ACCESS_KEY'),
                                string(credentialsId: 'secret-key', variable: 'AWS_SECRET_KEY')]) {
                    sh "aws eks --region us-east-1 update-kubeconfig --name
${params.CLUSTER_NAME}"
                }
            }
        }
    }

    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"}}'

    """"

    }

}

}

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/install.yaml

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

```


Jenkins Deployment pipeline

The image shows a Jenkinsfile configuration and the Jenkins web interface. The Jenkinsfile defines a pipeline with stages for cloning, building, and deploying to EKS. The web interface shows the pipeline's status and stage view.

```
pipeline {
    agent any
    stages {
        stage('Clone') {
            steps {
                checkout scm
            }
        }
        stage('Build') {
            steps {
                sh 'mvn clean package'
            }
        }
        stage('Deploy') {
            steps {
                sh 'kubectl apply -f deployment.yaml'
            }
        }
    }
}
```

Jenkins Web Interface:

- Dashboard > deployment-pipeline
- Status: deployment-pipeline (Success)
- Stage View:
- Average stage times: (full run time: ~47s)
- Stages: Login to EKS, Configure Prometheus & Grafana, Configure ArgoCD
- Builds: #2 (May 26 12:31, No Changes), #1 (May 26 12:30, No Changes)

Stage	Build #2 (May 26 12:31)	Build #1 (May 26 12:30)
Login to EKS	1s	2s
Configure Prometheus & Grafana	37s	2s (aborted)
Configure ArgoCD	8s	236ms (aborted)

AWS EC2 instances

The screenshot shows the AWS Management Console for the us-east-1 region. The left sidebar contains navigation links for EC2, including Dashboard, EC2 Global View, Events, Instances (selected), Instance Types, Launch Templates, Spot Requests, Savings Plans, Reserved Instances, Dedicated Hosts, Capacity Reservations, Images, AMIs, and AMI Catalog. The main content area is titled 'Instances (3)' and includes a search bar, a filter dropdown set to 'All states', and a table of three instances. The table columns are Name, Instance ID, Instance state, Instance type, Status check, Alarm status, and Availability. The instances listed are JENKINS-SERV..., panda-node, and another panda-node, all in a 'Running' state with '2/2 checks passed'. Below the table is a 'Select an instance' section. The bottom of the console shows the AWS logo, a search bar, and the user's name 'Kethan Pilla'.

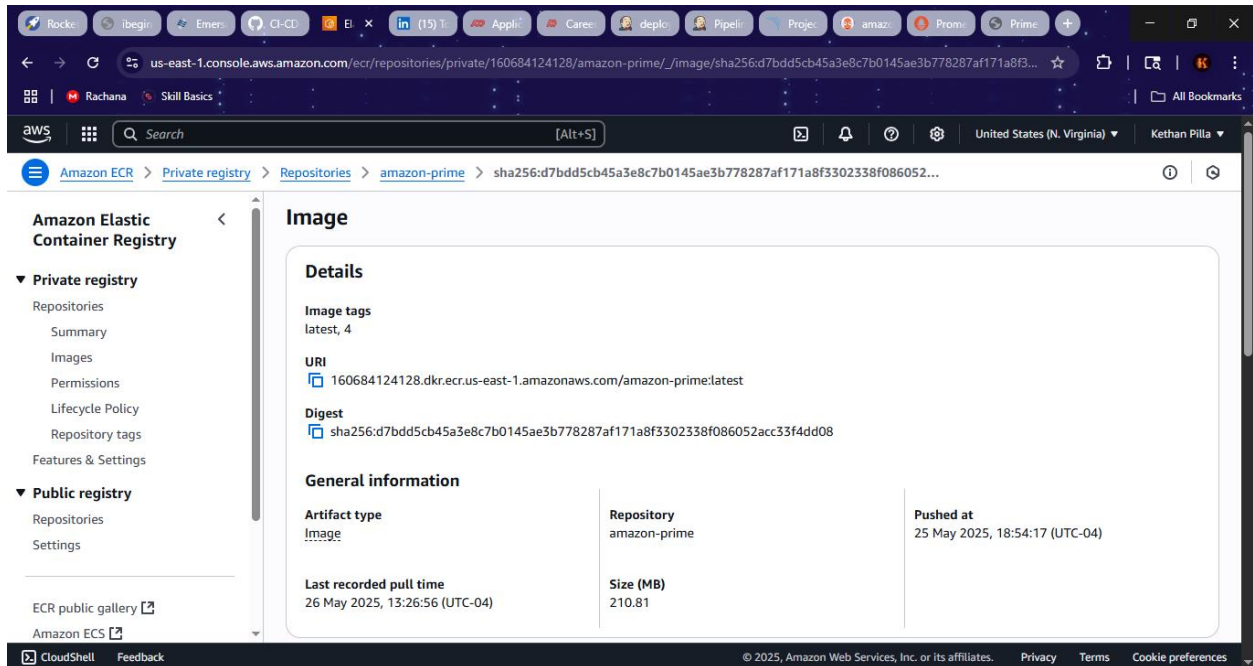
Name	Instance ID	Instance state	Instance type	Status check	Alarm status	Availability
JENKINS-SERV...	i-0bee45b1519dd2833	Running	t2.medium	2/2 checks passed	View alarms +	us-east-1d
panda-node	i-077b799e3613d62cd	Running	t2.medium	2/2 checks passed	View alarms +	us-east-1a
panda-node	i-03533914c7f9eb134	Running	t2.medium	2/2 checks passed	View alarms +	us-east-1b

Repositories

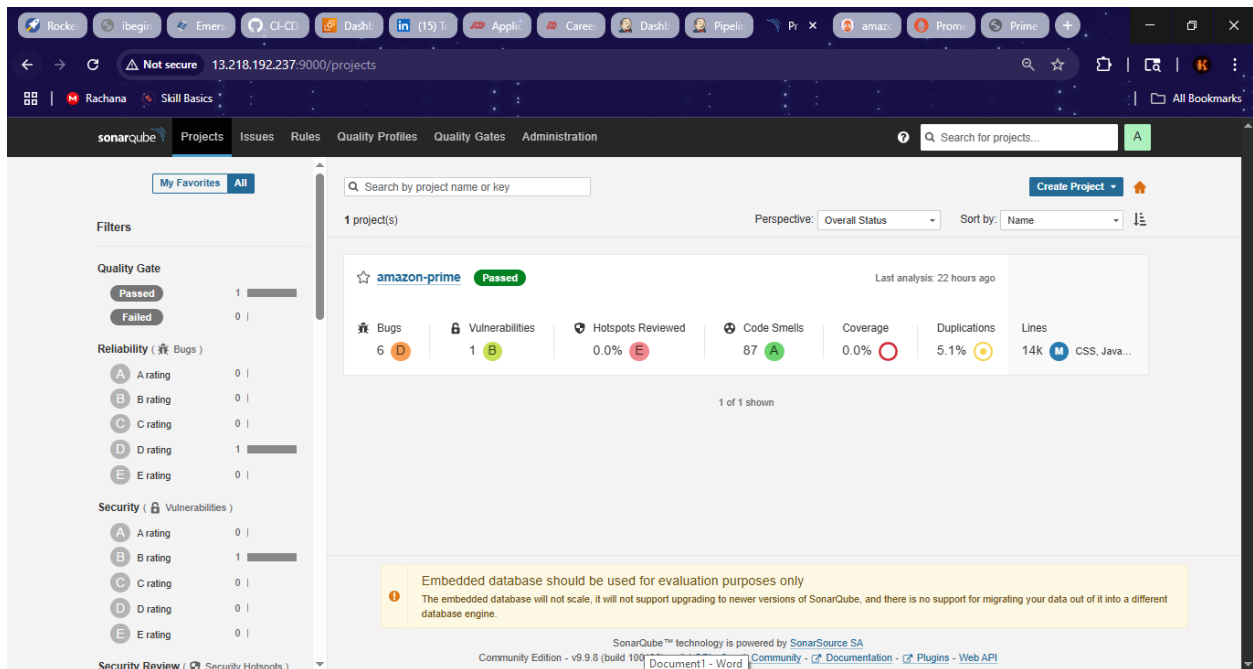
The screenshot shows the AWS Management Console for the us-east-1 region, specifically the 'Private repositories' page in the Amazon Elastic Container Registry (ECR). The left sidebar contains navigation links for Amazon ECR, including Private registry (selected), Public registry, ECR public gallery, Amazon ECS, Amazon EKS, Getting started, and Documentation. The main content area is titled 'Private repositories (2)' and includes a search bar, a 'View push commands' button, a 'Delete' button, an 'Actions' dropdown, and a 'Create repository' button. The table below lists two repositories: 'amazon-prime' and 'test'. The table columns are Repository name, URI, Created at, Tag immutability, and Encryption type. Both repositories were created on May 25, 2025, and have 'Mutable' tag immutability and 'AES-256' encryption.

Repository name	URI	Created at	Tag immutability	Encryption type
amazon-prime	160684124128.dkr.ecr.us-east-1.amazonaws.com/amazon-prime	25 May 2025, 18:53:20 (UTC-04)	Mutable	AES-256
test	160684124128.dkr.ecr.us-east-1.amazonaws.com/test	25 May 2025, 17:48:12 (UTC-04)	Mutable	AES-256

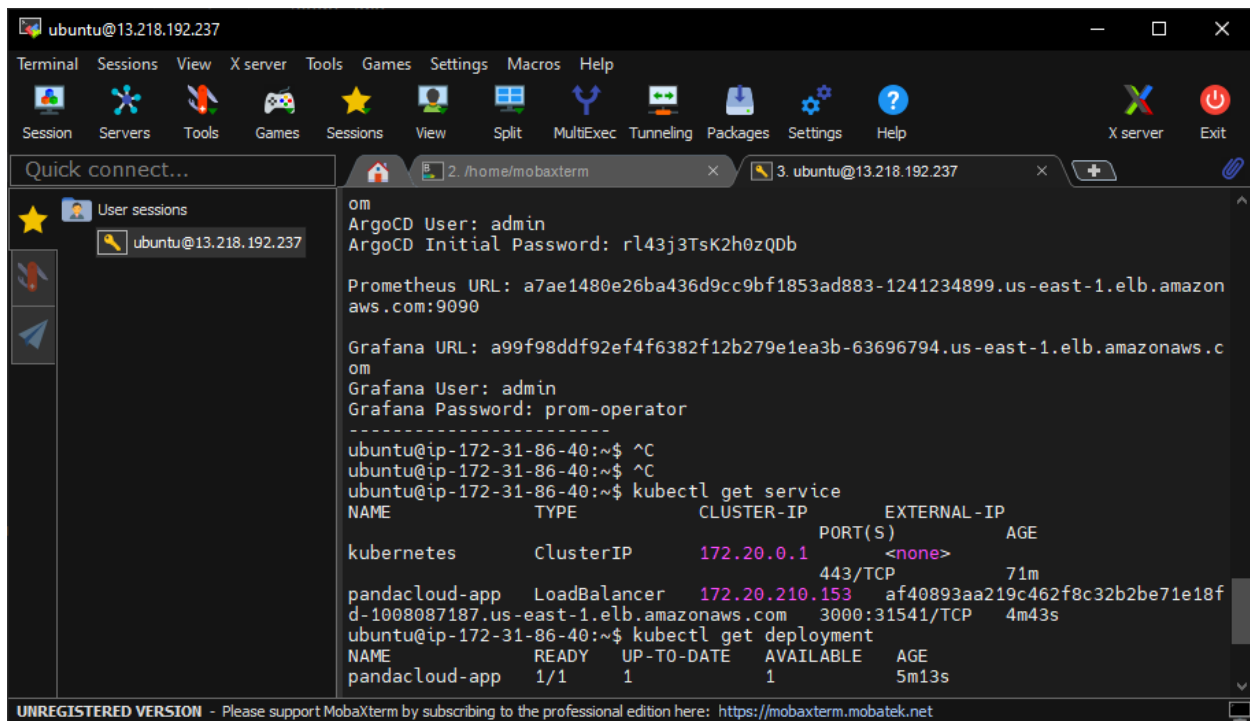
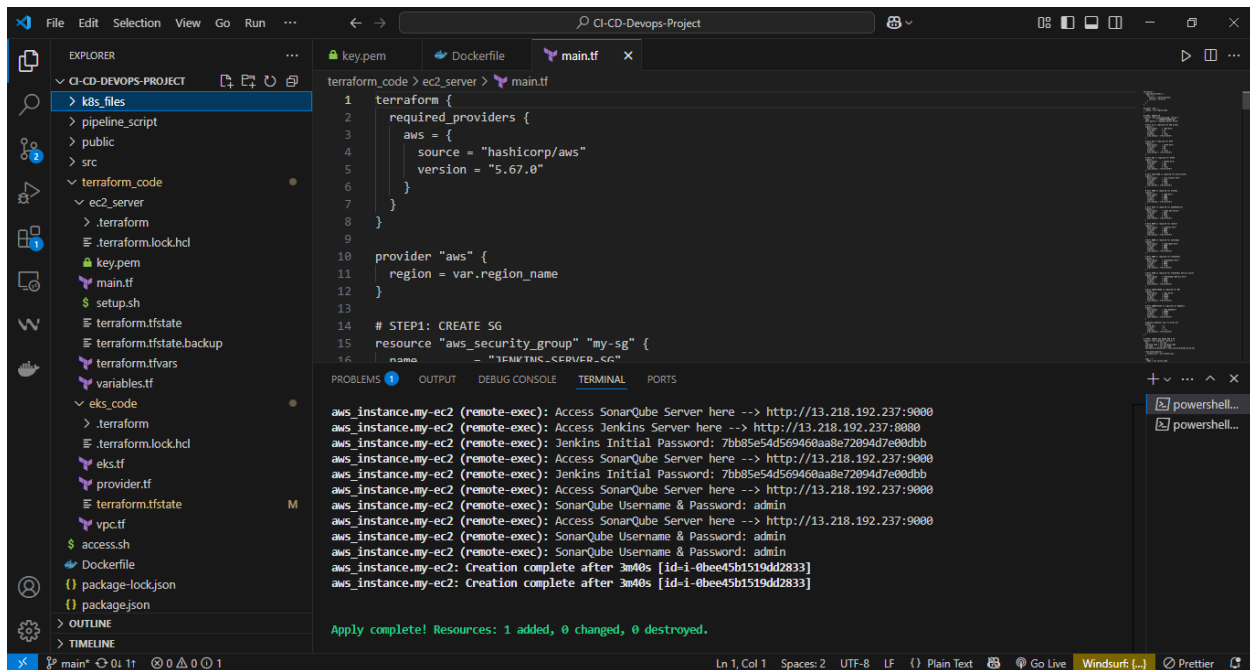
Amazon Elastic Container Registry (ECR)



SonarQube



Terraform



Argo application

The screenshot displays the Argo CD web interface for the 'amazon-prime-app'. The left sidebar shows navigation options: Applications, Settings, User Info, and Documentation. The main panel shows the application's health and sync status.

APP HEALTH: Healthy

SYNC STATUS: Synced to HEAD (b27574d)

LAST SYNC: Sync OK to b27574d

The application is a Kubernetes application named 'amazon-prime-app' with a sync status of 'Synced'. It is a deployment of a service named 'pandacloud app' with a sync status of 'Synced'. The application is a deployment of a service named 'pandacloud app' with a sync status of 'Synced'. The application is a deployment of a service named 'pandacloud app' with a sync status of 'Synced'.

Prometheus

The screenshot displays the Prometheus web interface. The left sidebar shows navigation options: Query, Alerts, and Status > Runtime & build information. The main panel shows the 'Build information' and 'Runtime information' sections.

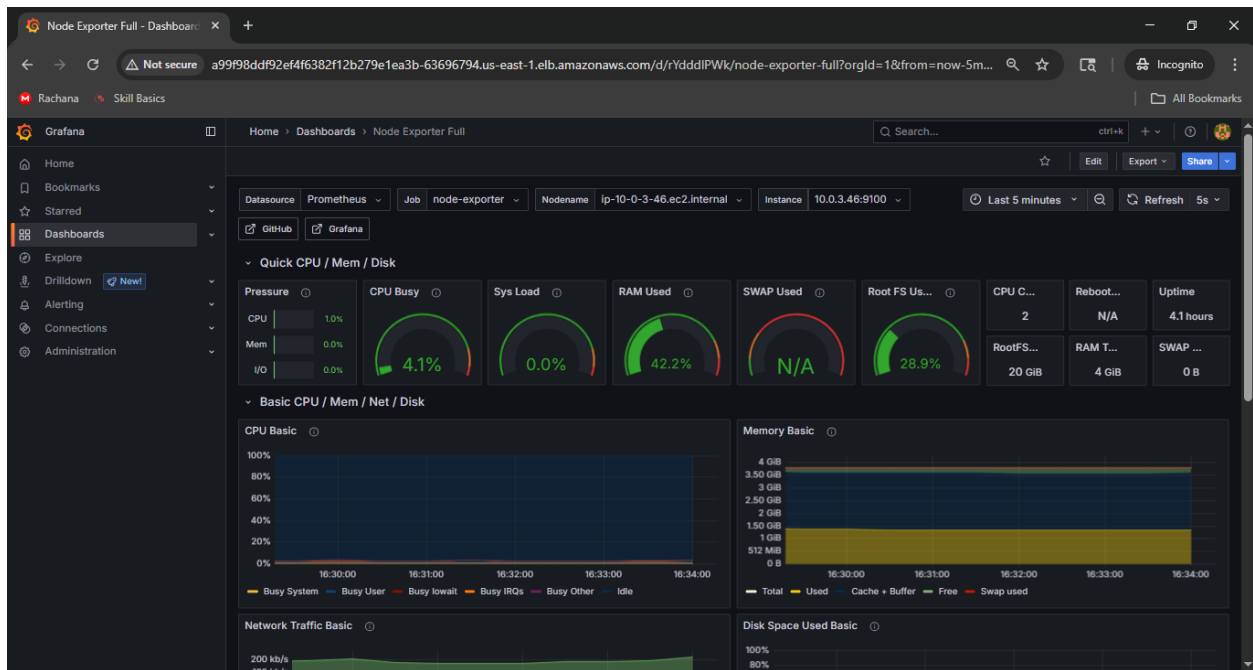
Build information

Field	Value
Version	3.4.0
Revision	546b1d242e209ed4228aa01a248dbf3e41e573ea
Branch	HEAD
BuildUser	root@b1749ffe17d4
BuildDate	20250517-06:54:08
GoVersion	go1.24.3

Runtime information

Field	Value
Start Time	2025-05-26T16:31:58Z
Working Directory	/prometheus
Hostname	prometheus-stable-kube-prometheus-sta-prometheus-0
Server Time	2025-05-26T20:32:59Z

Grafana dashboard



Deployed app

