MultiCloud DevOps Project Documentation

Table of Contents

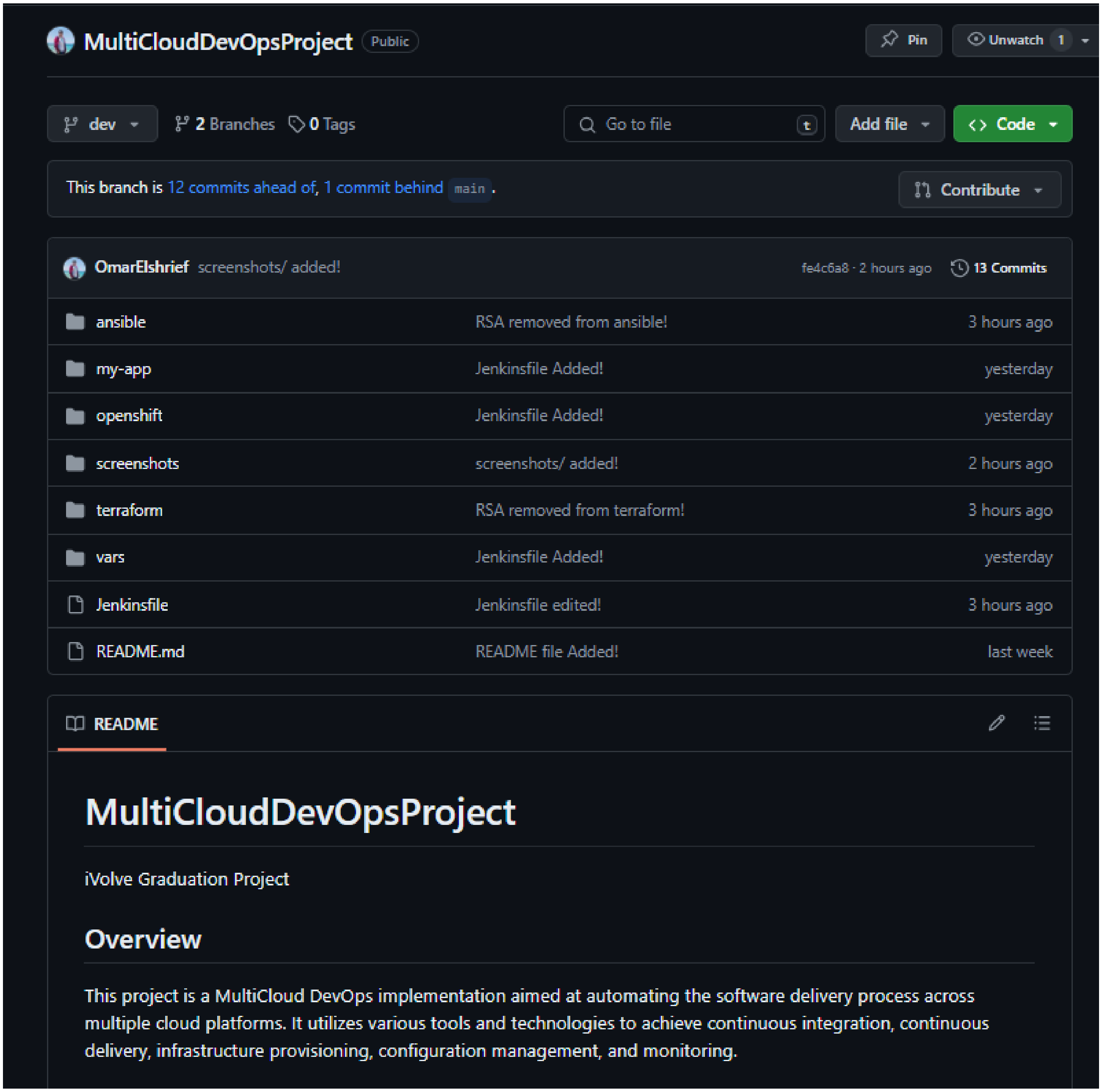
Introduction

This documentation provides a detailed guide to setting up and managing the MultiCloud DevOps Project, which aims to create a multi-cloud environment using AWS and OpenShift, leveraging various DevOps tools and practices.

Project Setup

GitHub Repository Setup

1. Create a new GitHub repository named `MultiCloudDevOpsProject`.
2. Initialize the repository with a README.
3. Create `main` and `dev` branches.
4. Push all code to the `dev` branch.
5. Before delivering the project, create a pull request to merge `dev` into `main`.



This branch is 12 commits ahead of, 1 commit behind `main`.

OmarElshrief screenshots/ added! fe4c6a8 · 2 hours ago

ansible	RSA removed from ansible!	3 hours ago
my-app	Jenkinsfile Added!	yesterday
openshift	Jenkinsfile Added!	yesterday
screenshots	screenshots/ added!	2 hours ago
terraform	RSA removed from terraform!	3 hours ago
vars	Jenkinsfile Added!	yesterday
Jenkinsfile	Jenkinsfile edited!	3 hours ago
README.md	README file Added!	last week

MultiCloudDevOpsProject

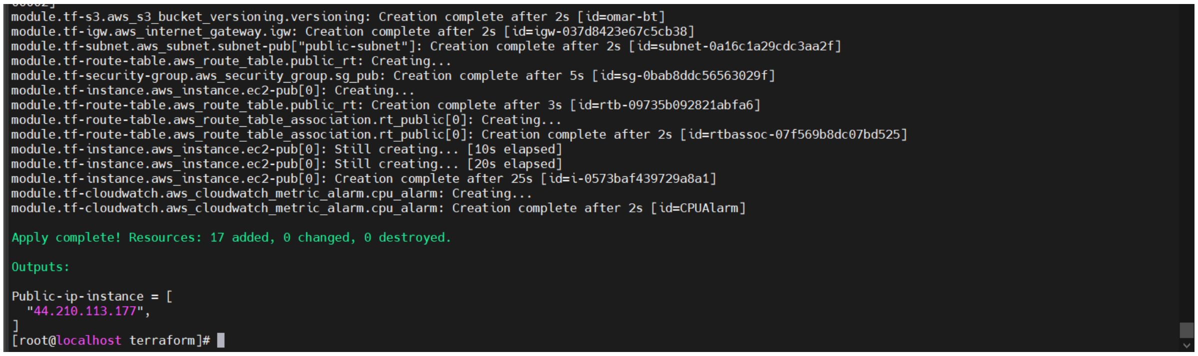
iVolve Graduation Project

Overview

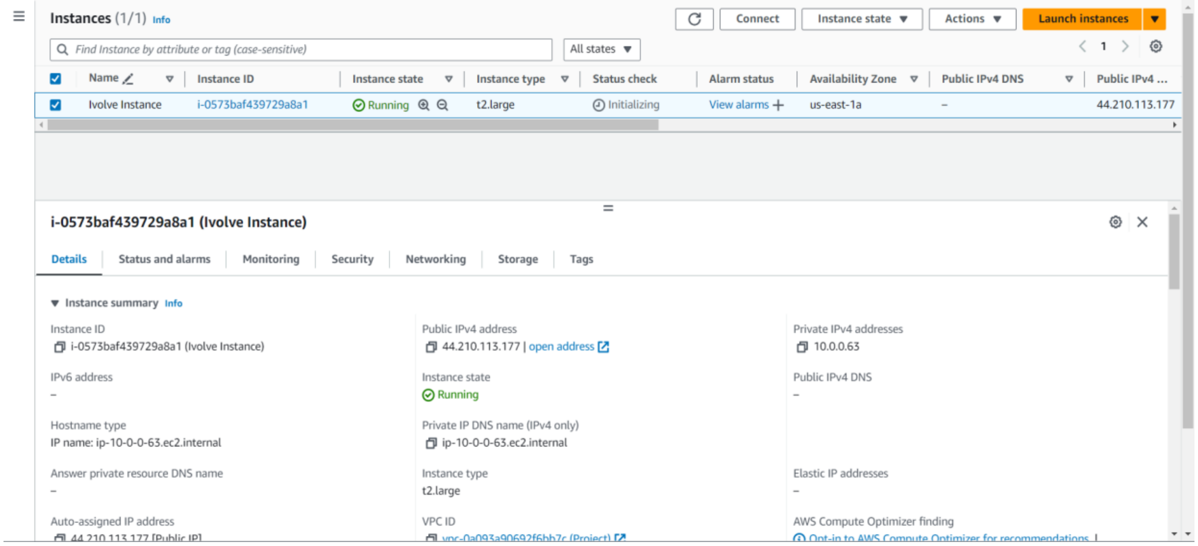
This project is a MultiCloud DevOps implementation aimed at automating the software delivery process across multiple cloud platforms. It utilizes various tools and technologies to achieve continuous integration, continuous delivery, infrastructure provisioning, configuration management, and monitoring.

Infrastructure Provisioning with Terraform

1. Write Terraform scripts to provision the following AWS resources:
   - VPC
   - Subnets
   - Security Groups
   - EC2 instances for application deployment
2. Use Terraform modules for better organization and reusability.
   - cd <repository-directory>/terraform
   - Edit the terraform.tfvars file to set values for your AWS setup.
   - terraform init   // Initialize Terraform
   - terraform plan   // Review the Plan
   - terraform apply // Apply configuration



```
module.tf-s3.aws_s3_bucket_versioning.versioning: Creation complete after 2s [id=omar-bt]
module.tf-igw.aws_internet_gateway.igw: Creation complete after 2s [id=igw-037d8423e67c5cb38]
module.tf-subnet.aws_subnet.subnet-pub["public-subnet"]: Creation complete after 2s [id=subnet-0a16c1a29cdc3aa2f]
module.tf-route-table.aws_route_table.public_rt: Creating...
module.tf-security-group.aws_security_group.sg_pub: Creation complete after 5s [id=sg-0bab8ddc56563029f]
module.tf-instance.aws_instance.ec2-pub[0]: Creating...
module.tf-route-table.aws_route_table.public_rt: Creation complete after 3s [id=rtb-09735b092821abfa6]
module.tf-route-table.aws_route_table_association.rt_public[0]: Creating...
module.tf-route-table.aws_route_table_association.rt_public[0]: Creation complete after 2s [id=rtbassoc-07f569b8dc07bd525]
module.tf-instance.aws_instance.ec2-pub[0]: Still creating... [10s elapsed]
module.tf-instance.aws_instance.ec2-pub[0]: Still creating... [20s elapsed]
module.tf-instance.aws_instance.ec2-pub[0]: Creation complete after 25s [id=i-0573baf439729a8a1]
module.tf-cloudwatch.aws_cloudwatch_metric_alarm.cpu_alarm: Creating...
module.tf-cloudwatch.aws_cloudwatch_metric_alarm.cpu_alarm: Creation complete after 2s [id=CPUAlarm]

Apply complete! Resources: 17 added, 0 changed, 0 destroyed.

Outputs:

Public-ip-instance = [
  "44.210.113.177",
]
[root@localhost terraform]#
```



☑	Name	Instance ID	Instance state	Instance type	Status check	Alarm status	Availability Zone	Public IPv4 DNS	Public IPv4 ...
☑	Ivolve Instance	i-0573baf439729a8a1	Running	t2.large	Initializing	View alarms +	us-east-1a	–	44.210.113.177

i-0573baf439729a8a1 (Ivolve Instance)

▼ Instance summary Info

Instance ID
i-0573baf439729a8a1 (Ivolve Instance)

Public IPv4 address
44.210.113.177 | open address

Private IPv4 addresses
10.0.0.63

IPv6 address
–

Instance state
Running

Public IPv4 DNS
–

Hostname type
IP name: ip-10-0-0-63.ec2.internal

Private IP DNS name (IPv4 only)
ip-10-0-0-63.ec2.internal

Answer private resource DNS name
–

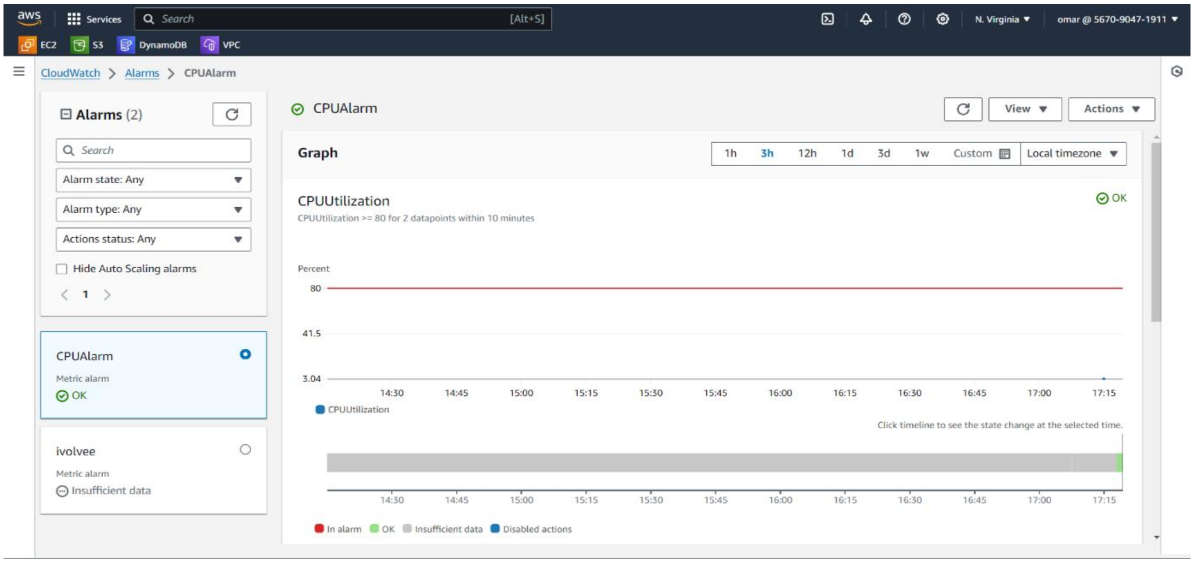
Instance type
t2.large

Elastic IP addresses
–

Auto-assigned IP address
44.210.113.177 [Public IP]

VPC ID
vpc-0a093a90692f6bb7c (Project)

AWS Compute Optimizer finding
Opt-in to AWS Compute Optimizer for recommendations.



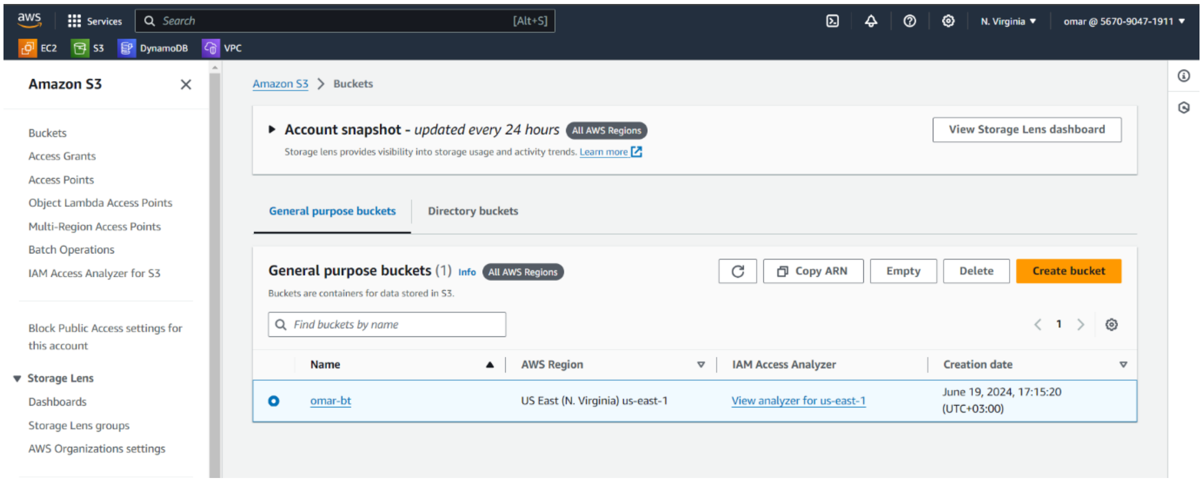
CPUAlarm
Metric alarm
OK

ivolvee
Metric alarm
Insufficient data

CPUAlarm

CPUUtilization
CPUUtilization >= 80 for 2 datapoints within 10 minutes
OK

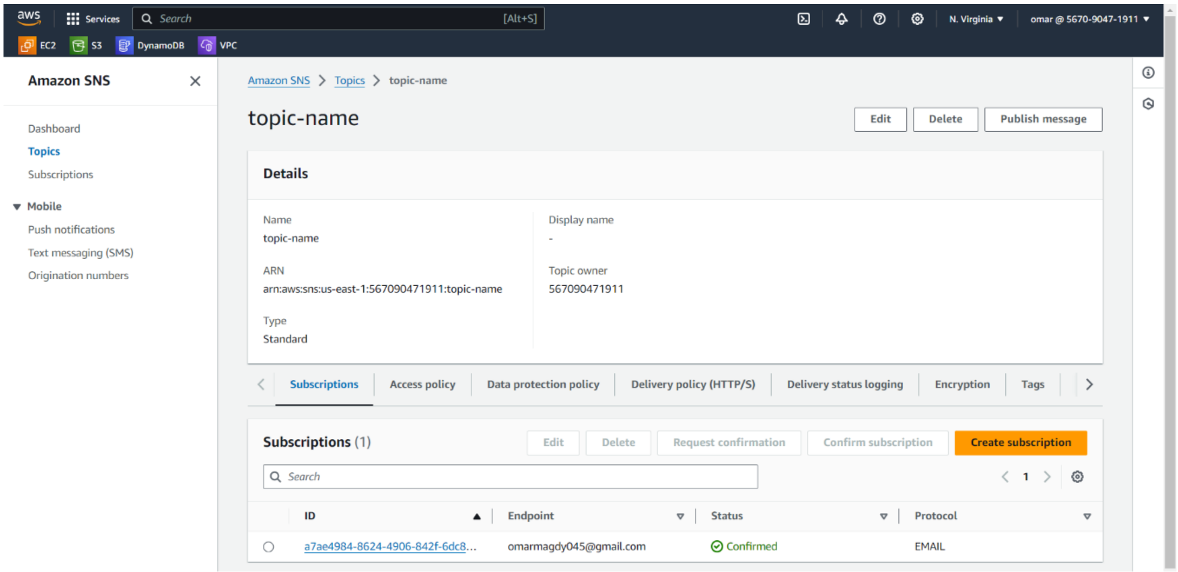
Click timeline to see the state change at the selected time.



Account snapshot - *updated every 24 hours* All AWS Regions
Storage lens provides visibility into storage usage and activity trends. Learn more

General purpose buckets (1)
Buckets are containers for data stored in S3.

	Name	AWS Region	IAM Access Analyzer	Creation date
●	omar-bt	US East (N. Virginia) us-east-1	View analyzer for us-east-1	June 19, 2024, 17:15:20 (UTC+03:00)



topic-name

Details

Name
topic-name

Display name
-

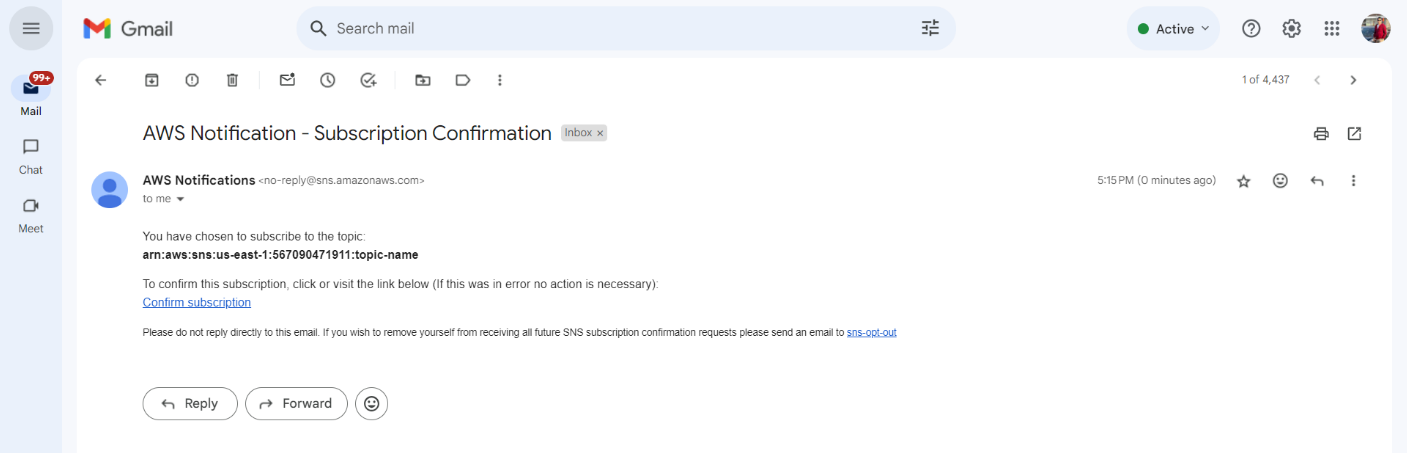
ARN
arn:aws:sns:us-east-1:567090471911:topic-name

Topic owner
567090471911

Type
Standard

Subscriptions (1)

	ID	Endpoint	Status	Protocol
○	a7ae4984-8624-4906-842f-6dc8...	omarmagdy045@gmail.com	Confirmed	EMAIL



AWS Notification - Subscription Confirmation

AWS Notifications <no-reply@sns.amazonaws.com>
to me

You have chosen to subscribe to the topic:
arn:aws:sns:us-east-1:567090471911:topic-name

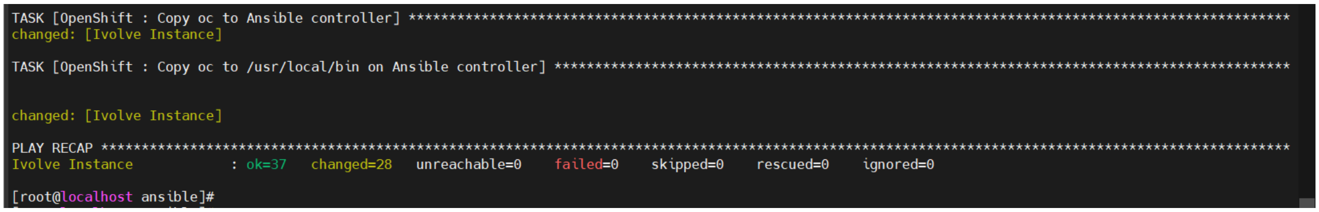
To confirm this subscription, click or visit the link below (If this was in error no action is necessary):
Confirm subscription

Please do not reply directly to this email. If you wish to remove yourself from receiving all future SNS subscription confirmation requests please send an email to sns-opt-out

Configuration Management with Ansible

1. Create Ansible playbooks for configuring EC2 instances:
   - Install required packages (e.g., Git, Docker, Java).
   - Install Jenkins and SonarQube.
   - Set up necessary environment variables.
2. Use Ansible roles to organize tasks.

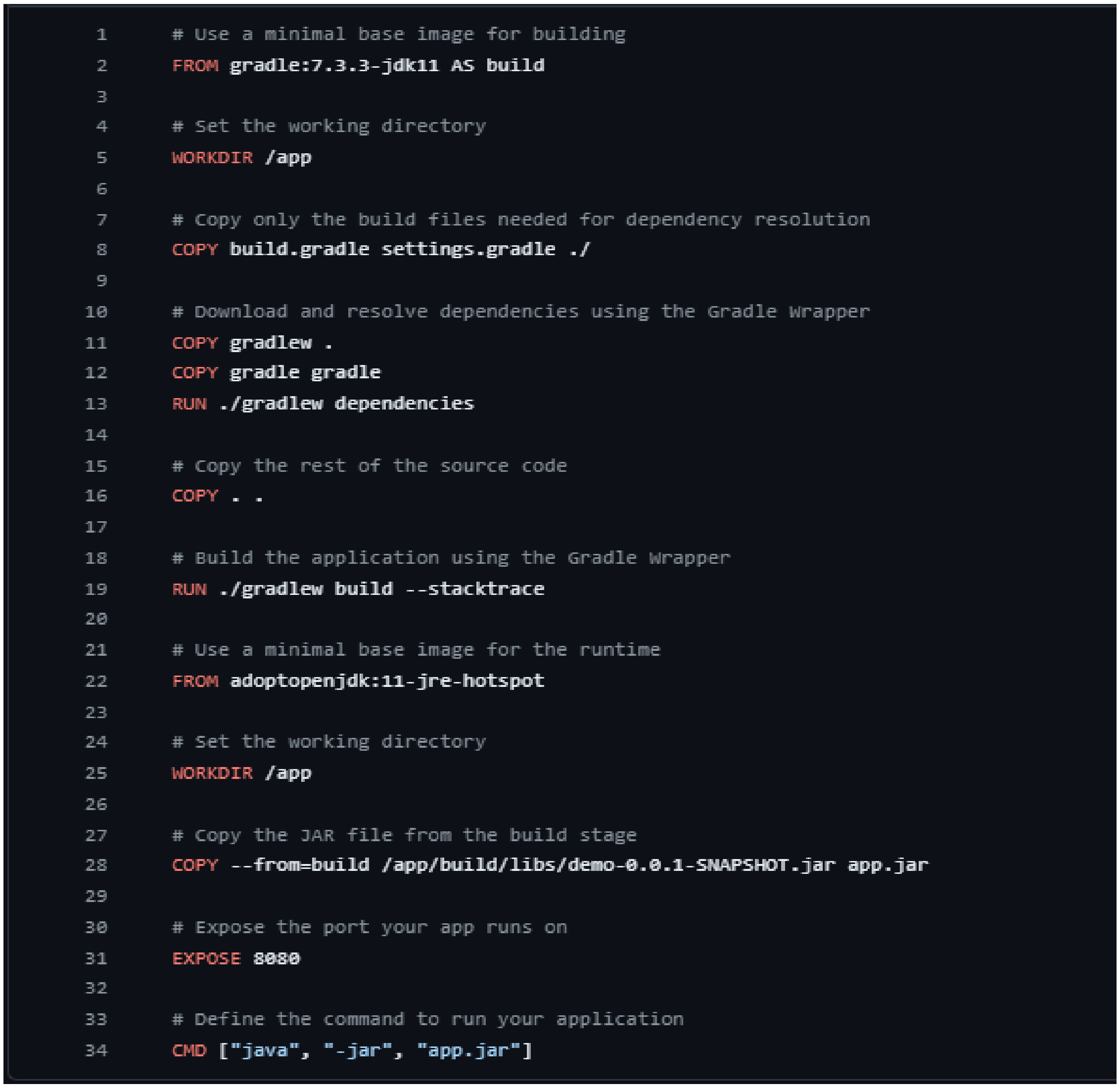
- cd ../Ansible // navigate to ansible directory
- ansible-playbook -i aws_ec2.yml playbook.yml // run ansible playbook



```
TASK [OpenShift : Copy oc to Ansible controller] ******************************************************************************************
changed: [Ivolve Instance]

TASK [OpenShift : Copy oc to /usr/local/bin on Ansible controller] ************************************************************************


changed: [Ivolve Instance]

PLAY RECAP ********************************************************************************************************************************
Ivolve Instance            : ok=37   changed=28   unreachable=0    failed=0    skipped=0    rescued=0    ignored=0

[root@localhost ansible]#
```

Containerization with Docker

1. Write a Dockerfile to build the application image.

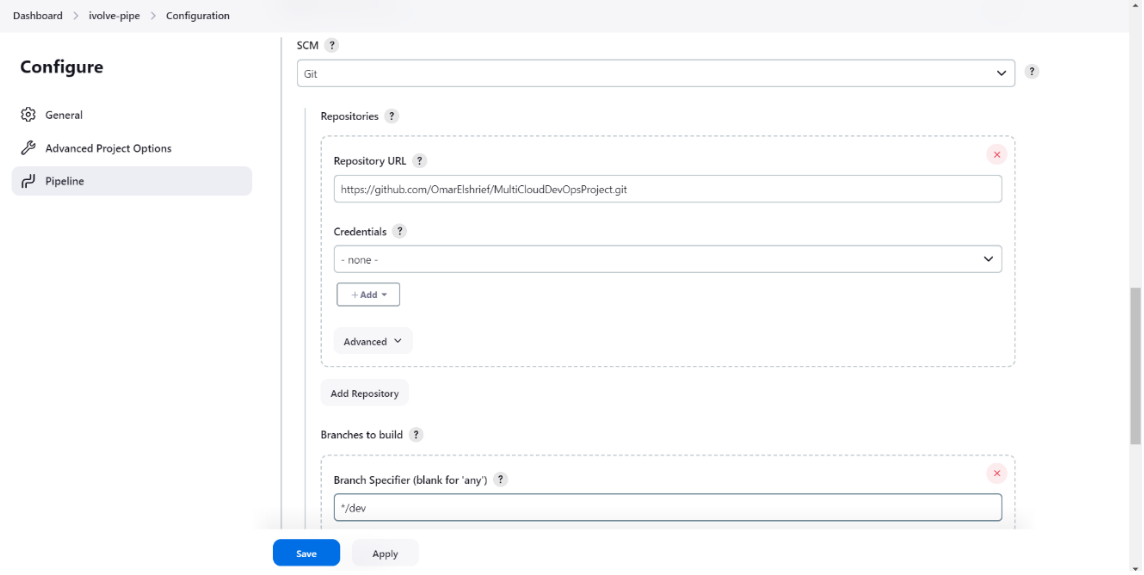


```
# Use a minimal base image for building
FROM gradle:7.3.3-jdk11 AS build

# Set the working directory
WORKDIR /app

# Copy only the build files needed for dependency resolution
COPY build.gradle settings.gradle ./

# Download and resolve dependencies using the Gradle Wrapper
COPY gradlew .
COPY gradle gradle
RUN ./gradlew dependencies

# Copy the rest of the source code
COPY . .

# Build the application using the Gradle Wrapper
RUN ./gradlew build --stacktrace

# Use a minimal base image for the runtime
FROM adoptopenjdk:11-jre-hotspot

# Set the working directory
WORKDIR /app

# Copy the JAR file from the build stage
COPY --from=build /app/build/libs/demo-0.0.1-SNAPSHOT.jar app.jar

# Expose the port your app runs on
EXPOSE 8080

# Define the command to run your application
CMD ["java", "-jar", "app.jar"]
```

Continuous Integration with Jenkins

1. Configure Jenkins jobs to build Docker images on code commits.

   - Choose SCM and add repository URL and branch name.



SCM: Git

Repositories

Repository URL: https://github.com/OmarElshrief/MultiCloudDevOpsProject.git

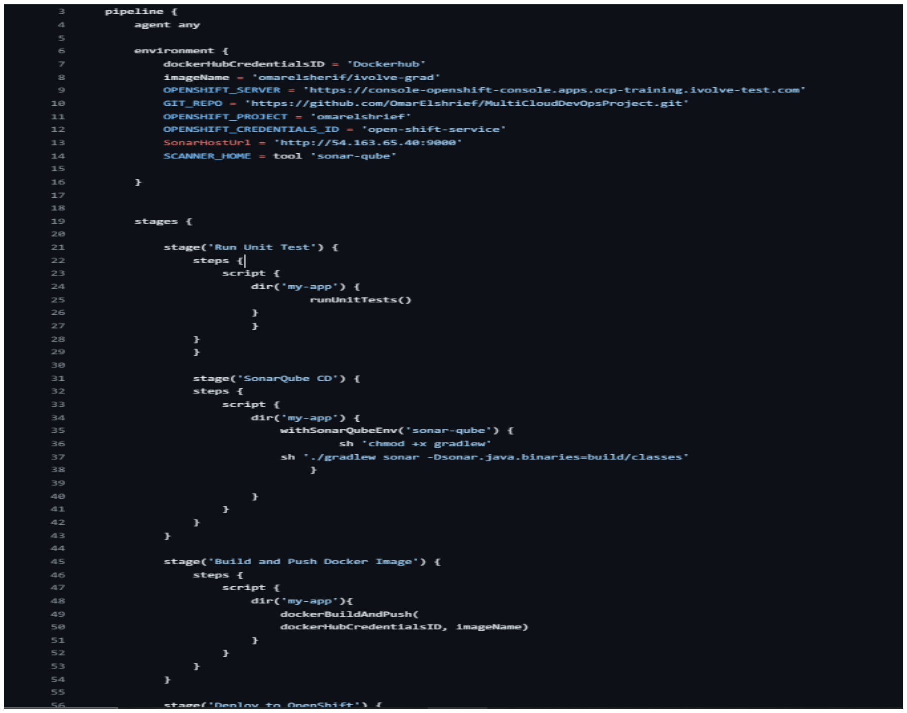
Credentials: - none -

Branches to build

Branch Specifier (blank for 'any'): */dev

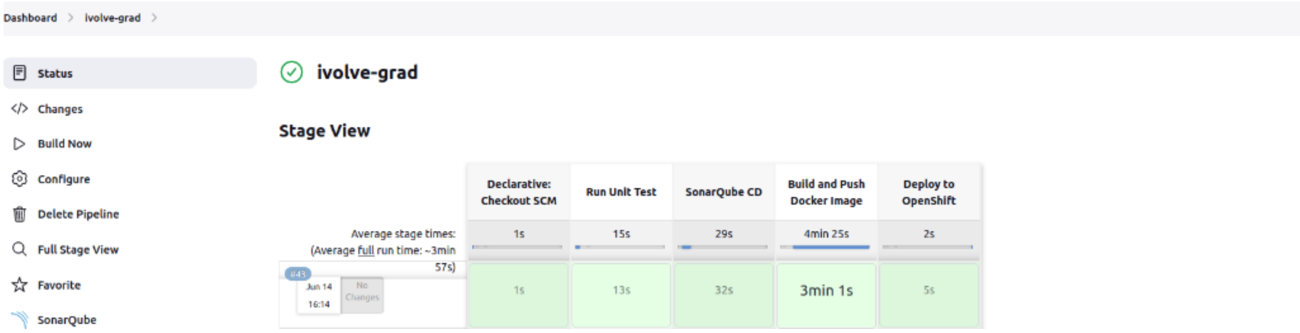
Automated Deployment Pipeline

1. Configure a Jenkins pipeline in `Jenkinsfile` with the following stages:
   - Git Checkout
   - Build
   - Unit Test
   - SonarQube Test
   - Deploy on OpenShift
2. Use a shared Jenkins library for reusable pipeline code.



```
pipeline {
    agent any

    environment {
        dockerHubCredentialsID = 'Dockerhub'
        imageName = 'omarelsherif/ivolve-grad'
        OPENSHIFT_SERVER = 'https://console-openshift-console.apps.ocp-training.ivolve-test.com'
        GIT_REPO = 'https://github.com/OmarElshrief/MultiCloudDevOpsProject.git'
        OPENSHIFT_PROJECT = 'omarelshrief'
        OPENSHIFT_CREDENTIALS_ID = 'open-shift-service'
        SonarHostUrl = 'http://54.163.65.40:9000'
        SCANNER_HOME = tool 'sonar-qube'

    }


    stages {

        stage('Run Unit Test') {
            steps {
                script {
                    dir('my-app') {
                            runUnitTests()
                    }
                    }
            }
            }

            stage('SonarQube CD') {
            steps {
                script {
                    dir('my-app') {
                        withSonarQubeEnv('sonar-qube') {
                                sh 'chmod +x gradlew'
                        sh './gradlew sonar -Dsonar.java.binaries=build/classes'
                            }

                    }
                }
            }
        }

        stage('Build and Push Docker Image') {
            steps {
                script {
                    dir('my-app'){
                        dockerBuildAndPush(
                        dockerHubCredentialsID, imageName)
                    }
                }
            }
        }

        stage('Deploy to OpenShift') {
```

- Run Pipeline: Execute the pipeline from Jenkins. Ensure it’s running successfully..



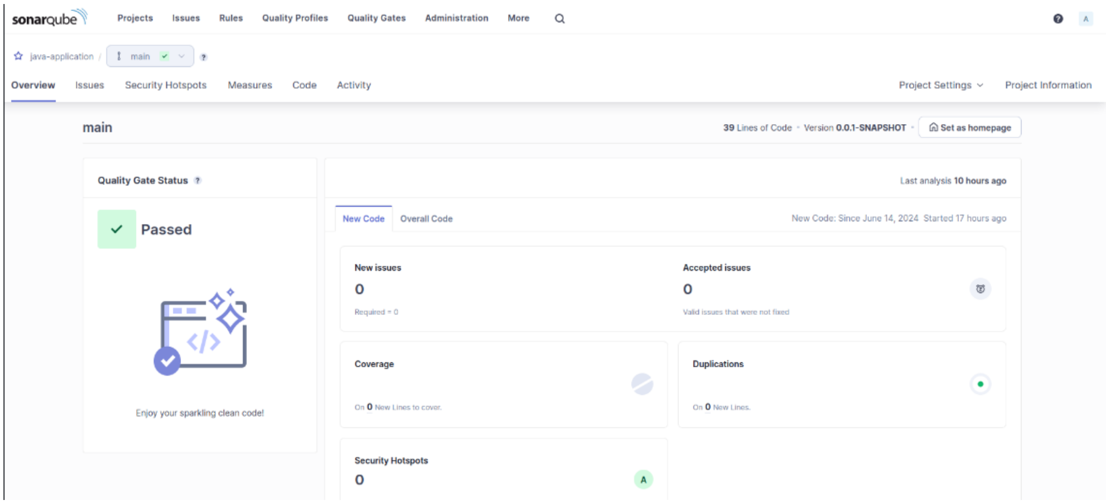
ivolve-grad

Stage View

	Declarative: Checkout SCM	Run Unit Test	SonarQube CD	Build and Push Docker Image	Deploy to OpenShift
Average stage times: (Average full run time: ~3min 57s)	1s	15s	29s	4min 25s	2s
#43 Jun 14 16:14 No Changes	1s	13s	32s	3min 1s	5s

SonarQube

- Review code quality reports on Sonarqube..



main

39 Lines of Code · Version 0.0.1-SNAPSHOT

Quality Gate Status

Passed

Enjoy your sparkling clean code!

Last analysis 10 hours ago

New Code: Since June 14, 2024 Started 17 hours ago

New issues
0
Required = 0

Accepted issues
0
Valid issues that were not fixed

Coverage
On 0 New Lines to cover.

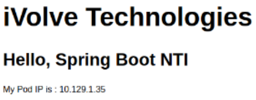
Duplications
On 0 New Lines.

Security Hotspots
0

Application Deployment

Verify your application is running on the OpenShift cluster.

- Login In Your Cluster and Run
- oc get all -n namespace



iVolve Technologies

Hello, Spring Boot NTI

My Pod IP is : 10.129.1.35

Architecture Overview

The MultiCloud DevOps Project architecture includes the following components:

1. **Version Control (GitHub)**: Central repository for all project code, including Terraform scripts, Ansible playbooks, Dockerfiles, and Jenkins configuration.
2. **Infrastructure Provisioning (Terraform)**: Scripts to provision AWS resources such as VPC, subnets, security groups, and EC2 instances.
3. **Configuration Management (Ansible)**: Playbooks to configure EC2 instances with necessary packages and environment settings.
4. **Containerization (Docker)**: Dockerfiles to create container images for the application.
5. **CI/CD Pipeline (Jenkins)**: Jenkins jobs and pipelines to automate code integration, testing, and deployment.
6. **Deployment (OpenShift)**: Platform for deploying and managing application containers.
7. **Monitoring and Logging**: Centralized logging setup on OpenShift and AWS CloudWatch integration for monitoring.

Troubleshooting Guidelines

Common Issues and Solutions

1. **Terraform Errors**:
   - **Issue**: Terraform script fails to provision resources.
   - **Solution**: Check the Terraform logs for error messages. Ensure AWS credentials and permissions are correctly configured.
2. **Ansible Playbook Failures**:
   - **Issue**: Ansible playbook fails to run.
   - **Solution**: Verify the playbook syntax and roles. Ensure SSH connectivity to the target EC2 instances.
3. **Docker Build Issues**:
   - **Issue**: Docker image build fails.
   - **Solution**: Check the Dockerfile for errors. Ensure all dependencies and build commands are correctly specified.
4. **Jenkins Job Failures**:
   - **Issue**: Jenkins job fails to execute.
   - **Solution**: Review the Jenkins job configuration and console output. Ensure Jenkins has the necessary permissions and access to required resources.
5. **SonarQube Analysis Errors**:
   - **Issue**: SonarQube analysis fails.
   - **Solution**: Verify SonarQube configuration and connectivity. Ensure the SonarQube server is running and accessible.
6. **Deployment Issues**:
   - **Issue**: Application fails to deploy on OpenShift.
   - **Solution**: Check the deployment logs on OpenShift. Ensure the Docker image is correctly built and available in the registry.
7. **Centralized Logging Problems**:
   - **Issue**: Logs are not being collected centrally.
   - **Solution**: Verify the logging configuration on OpenShift. Ensure the logging service is running and correctly set up.

8. **AWS Integration Issues**:
   - **Issue**: AWS services (S3, CloudWatch) integration fails.
   - **Solution**: Check AWS IAM policies and permissions. Ensure the services are correctly referenced in the Terraform code.

Additional Resources

- **Terraform Documentation**: https://www.terraform.io/docs/
- **Ansible Documentation**: https://docs.ansible.com/
- **Docker Documentation**: https://docs.docker.com/
- **Jenkins Documentation**: https://www.jenkins.io/doc/
- **SonarQube Documentation**: https://docs.sonarqube.org/
- **OpenShift Documentation**: https://docs.openshift.com/
- **AWS Documentation**: https://docs.aws.amazon.com/