Accelerating Cloud-Native: Multi-Region EKS Deployment with Hybrid IaC & DevOps Automation

1. Introduction

1.1. Project Overview

This document provides a comprehensive overview of a modern DevOps project focused on deploying a full-stack web application across multiple AWS regions using Infrastructure as Code (IaC) and a robust CI/CD pipeline. The application consists of an Angular frontend, a Spring Boot backend, and an Amazon RDS database. The infrastructure is orchestrated on Amazon EKS (Elastic Kubernetes Service) clusters, ensuring scalability, resilience, and automated management.

1.2. Goals and Objectives

The primary goals of this project were to:

- Automate end-to-end infrastructure provisioning and application deployment.
- Achieve multi-region high availability and disaster recovery for the application.
- Implement robust CI/CD practices for rapid and reliable software delivery.
- Ensure code quality through integrated static analysis.
- Establish comprehensive monitoring and alerting for operational insights.
- Implement a robust backup strategy for critical data and recovery.
- Route 53 Global DNS Failover

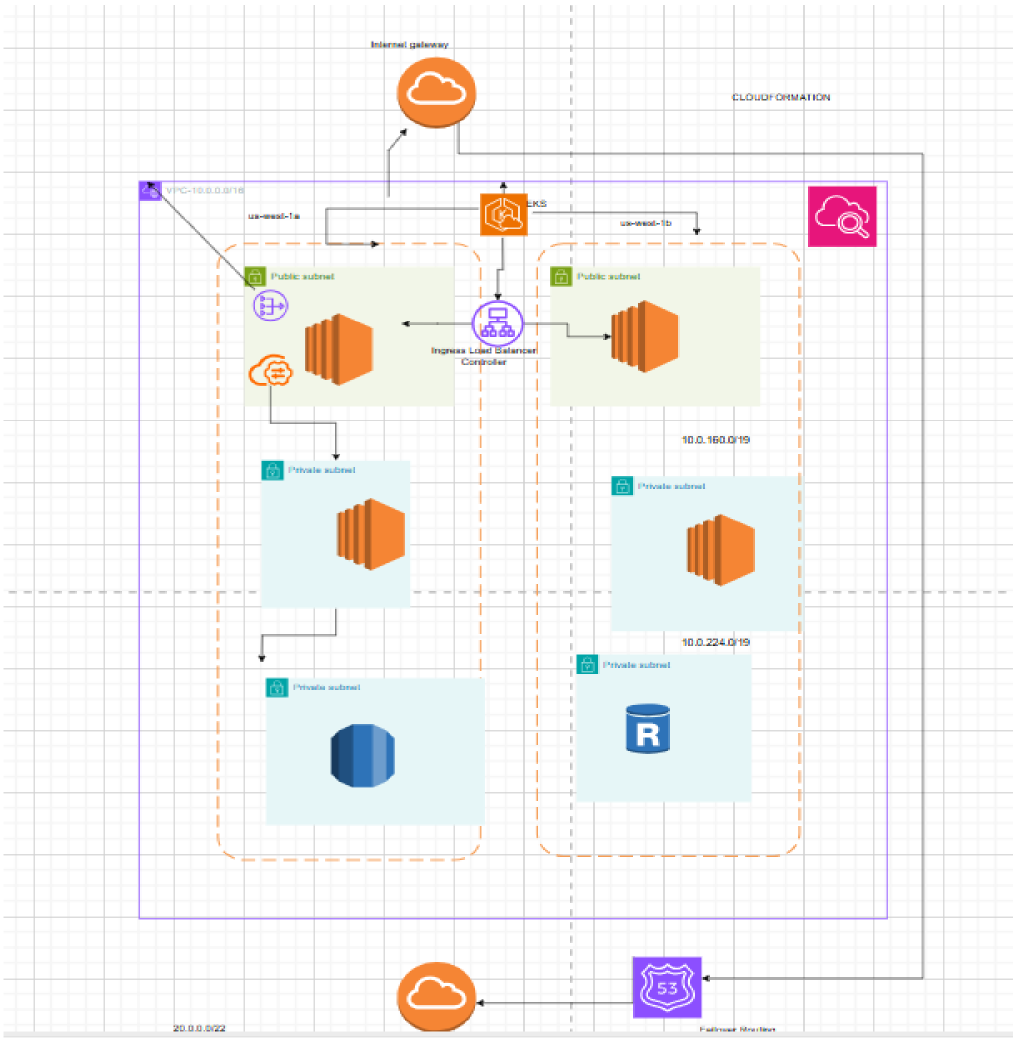
1.3. Key Technologies Utilized

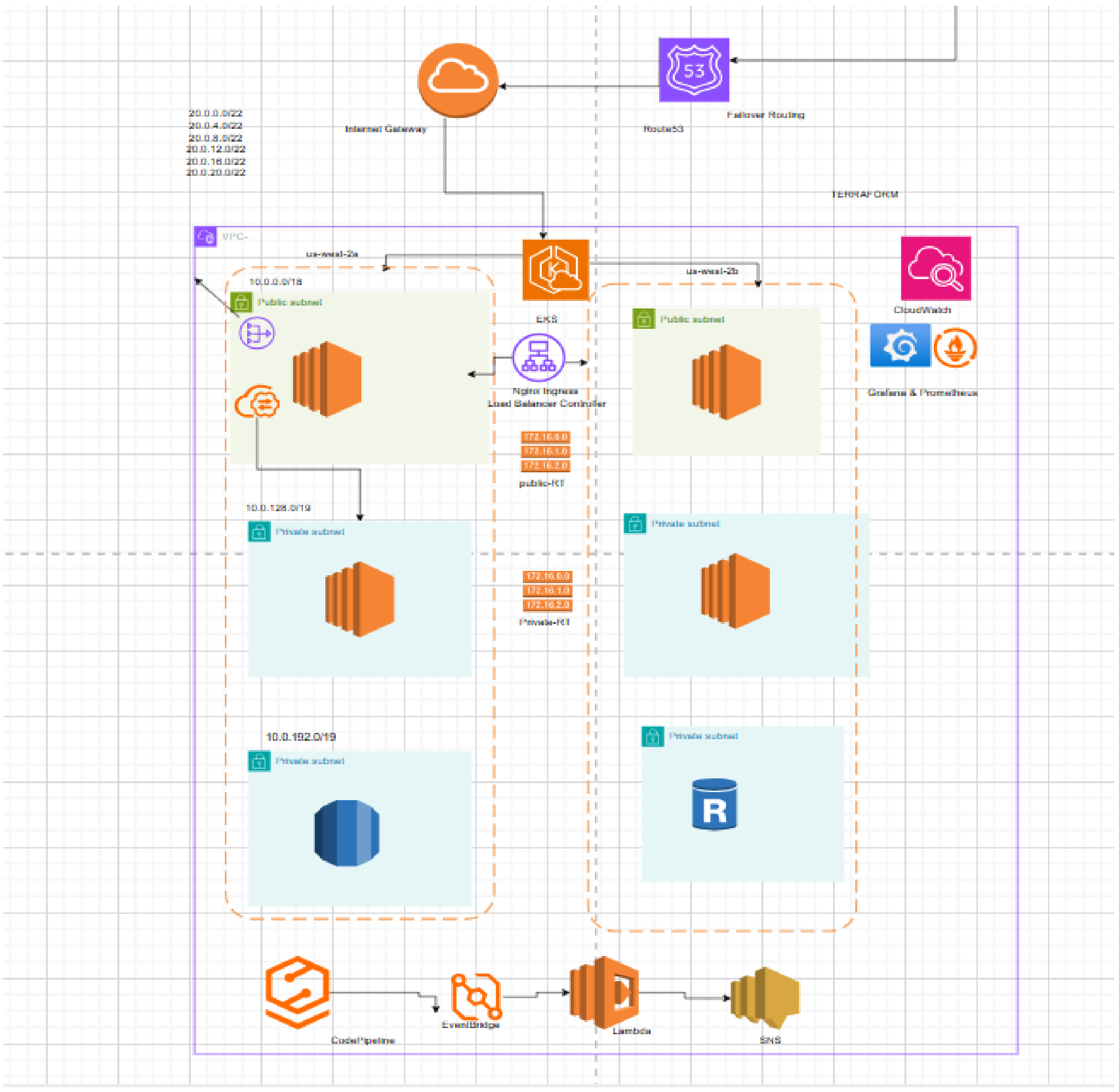
- **Cloud Provider:** Amazon Web Services (AWS)
- **Infrastructure as Code (IaC):** AWS CloudFormation (CFT), Terraform
- **Container Orchestration:** Amazon Elastic Kubernetes Service (EKS)
- **Container Registry:** Amazon Elastic Container Registry (ECR)
- **CI/CD:** AWS CodePipeline, AWS CodeBuild
- **Code Quality:** SonarQube
- **Application Frameworks:** Angular (Frontend), Spring Boot (Backend)
- **Database:** Amazon Relational Database Service (RDS)

- **Networking & Access:** Ingress NGINX, Amazon Route 53, Application Load Balancer (ALB)
- **Monitoring & Alerting:** Amazon CloudWatch, Amazon EventBridge, AWS Lambda, Amazon SNS
- **Backup:** AWS Backup and Recovery
- **Route53:** Route 53 Global DNS Failover

2.1. High-Level Multi-Region Architecture

The application is deployed across two distinct AWS regions, us-west-1 and us-west-2, providing redundancy and improving user experience by serving traffic from the nearest region. Amazon Route 53 handles intelligent routing and failover between the regions, directing users to the healthy application endpoint.





2.2. Regional Architecture Deep Dive (us-west-1 & us-west-2)

Each AWS region hosts a self-contained, identical (or near-identical) application stack. This "active-active" or "active-passive" setup ensures that if one region experiences an outage, the other can continue to serve traffic with minimal interruption.

2.2.1. Network Topology (VPC, Subnets, Route Tables, IGW, NAT Gateway)

In each region, a dedicated Virtual Private Cloud (VPC) is established, providing a logically isolated network.

- **Public Subnets:** Designed for resources that require direct internet access, such as Application Load Balancers and NAT Gateways.
- **Private Subnets:** Dedicated to internal resources like EKS worker nodes and RDS instances, ensuring they are not directly exposed to the internet.

- **Internet Gateway (IGW):** Enables internet connectivity for resources in public subnets.
- **NAT Gateway:** Allows instances in private subnets to initiate outbound connections to the internet (e.g., for pulling Docker images, software updates) without being publicly accessible.
- **Route Tables:** Configured to direct traffic appropriately between subnets and to the IGW or NAT Gateway.

2.3. Application Stack (Angular, Spring Boot, RDS)

- **Angular Frontend:** The user interface responsible for user interaction and making API calls to the Spring Boot backend.
- **Spring Boot Backend:** A RESTful API that handles business logic, processes requests from the Angular frontend, and interacts with the Amazon RDS database.
- **Amazon RDS:** A managed relational database service providing a highly available and scalable data store for the application.

3. Infrastructure as Code (IaC)

3.1. IaC Strategy: Hybrid Approach

To demonstrate versatility and accommodate potential organizational preferences, a hybrid IaC strategy was adopted:

- **AWS CloudFormation (CFT):** Used for infrastructure provisioning in us-west-1.
- **Terraform:** Used for infrastructure provisioning in us-west-1. Both tools manage the complete set of AWS resources, from VPC to EKS cluster and RDS instances.

3.2. Region 1: us-west-1 with AWS CloudFormation (CFT)

CloudFormation templates define the desired state of all infrastructure resources in us-west-1.

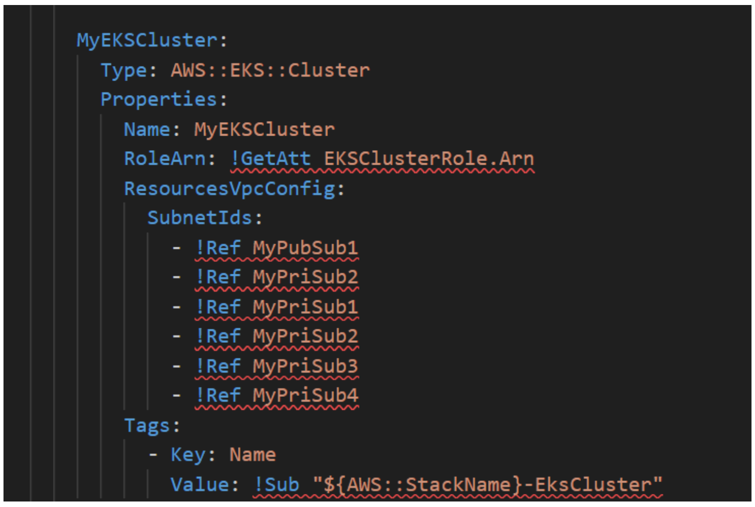
3.2.1. CFT Template Structure

Templates are modularized (e.g., separate templates for VPC, EKS, RDS) and can be nested for better organization and reusability. Parameters are used to customize deployments (e.g., instance types, CIDR blocks).

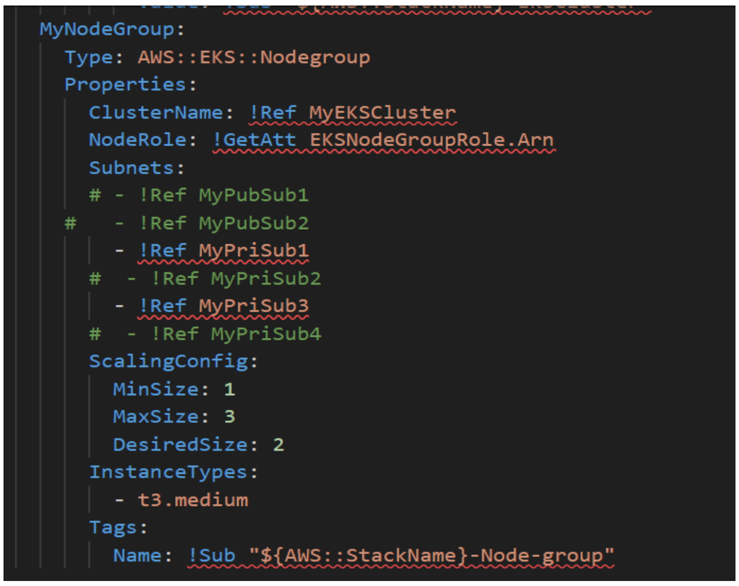
3.2.2. EKS Cluster Resources Provisioned via CFT

- AWS::EKS::Cluster: The EKS control plane.
- AWS::EKS::Nodegroup: Managed node groups for worker nodes.
- AWS::EC2::SecurityGroup: Security groups for EKS control plane and worker nodes.

- AWS::IAM::Role: IAM roles for EKS service and node groups.



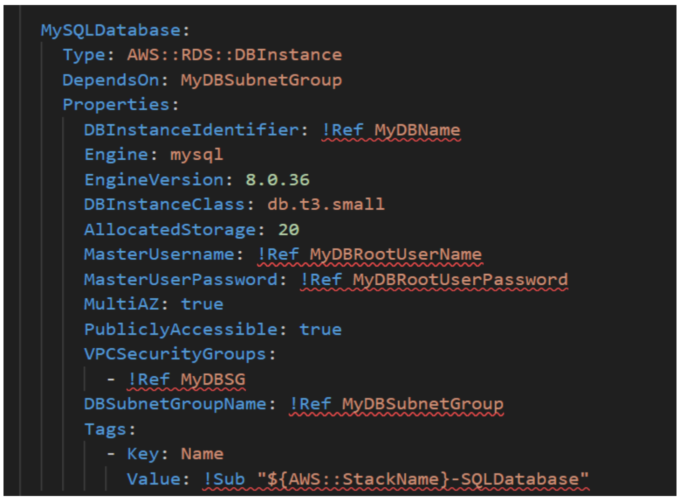
```yaml
    MyEKSCluster:
      Type: AWS::EKS::Cluster
      Properties:
        Name: MyEKSCluster
        RoleArn: !GetAtt EKSClusterRole.Arn
        ResourcesVpcConfig:
          SubnetIds:
            - !Ref MyPubSub1
            - !Ref MyPriSub2
            - !Ref MyPriSub1
            - !Ref MyPriSub2
            - !Ref MyPriSub3
            - !Ref MyPriSub4
        Tags:
          - Key: Name
            Value: !Sub "${AWS::StackName}-EksCluster"
```



```yaml
  MyNodeGroup:
    Type: AWS::EKS::Nodegroup
    Properties:
      ClusterName: !Ref MyEKSCluster
      NodeRole: !GetAtt EKSNodeGroupRole.Arn
      Subnets:
      # - !Ref MyPubSub1
    #   - !Ref MyPubSub2
        - !Ref MyPriSub1
      #  - !Ref MyPriSub2
        - !Ref MyPriSub3
      #  - !Ref MyPriSub4
      ScalingConfig:
        MinSize: 1
        MaxSize: 3
        DesiredSize: 2
      InstanceTypes:
        - t3.medium
      Tags:
        Name: !Sub "${AWS::StackName}-Node-group"
```

3.2.3. RDS Instance Provisioned via CFT

- AWS::RDS::DBInstance: The relational database instance.
- AWS::RDS::DBSubnetGroup: Defines the subnets for the RDS instance.
- AWS::EC2::SecurityGroup: Security group for RDS, allowing inbound traffic from EKS worker nodes.



```
MySQLDatabase:
  Type: AWS::RDS::DBInstance
  DependsOn: MyDBSubnetGroup
  Properties:
    DBInstanceIdentifier: !Ref MyDBName
    Engine: mysql
    EngineVersion: 8.0.36
    DBInstanceClass: db.t3.small
    AllocatedStorage: 20
    MasterUsername: !Ref MyDBRootUserName
    MasterUserPassword: !Ref MyDBRootUserPassword
    MultiAZ: true
    PubliclyAccessible: true
    VPCSecurityGroups:
      - !Ref MyDBSG
    DBSubnetGroupName: !Ref MyDBSubnetGroup
    Tags:
      - Key: Name
        Value: !Sub "${AWS::StackName}-SQLDatabase"
```

3.2.4. IAM Permissions for EKS via CFT

- Appropriate IAM roles (e.g., AmazonEKSClusterPolicy, AmazonEKSWorkerNodePolicy, AmazonEC2ContainerRegistryReadOnly) are defined and attached to ensure EKS components have necessary permissions to operate and scale.

3.3. Region 2: us-west-2 with Terraform

Terraform configurations define the desired state of infrastructure resources in us-west-2.

3.3.1. EKS Cluster Resources Provisioned via Terraform

- aws_eks_cluster: The EKS control plane resource.
- aws_eks_node_group: Managed node groups.
- aws_security_group: Security groups for EKS.
- aws_iam_role: IAM roles for EKS.

3.3.2. RDS Instance Provisioned via Terraform

- aws_db_instance: The relational database instance.
- aws_db_subnet_group: Defines the subnets for RDS.
- aws_security_group: Security group for RDS.

3.3.3. IAM Permissions for EKS via Terraform

IAM roles and policies (aws_iam_role, aws_iam_policy, aws_iam_role_policy_attachment) are defined to grant EKS and its worker nodes the necessary permissions to function and interact with AWS services.

3.4. Key IaC Challenges & Solutions

- **Challenge:** Ensuring consistency in resource naming and configuration parameters between CloudFormation and Terraform for multi-region symmetry.
  - **Solution:** Strict naming conventions, comprehensive documentation, and parameterization of IaC templates/modules.
- **Challenge:** Managing IAM permissions complexity for EKS (Service Linked Roles, Worker Node Roles, CNI permissions, etc.).
  - **Solution:** Leveraging AWS-managed policies where possible and carefully defining custom policies with the principle of least privilege.

4. Continuous Integration / Continuous Delivery (CI/CD)

4.1. CI/CD Philosophy

The project embraces a fully automated CI/CD philosophy to enable rapid, reliable, and consistent delivery of application features and infrastructure updates. This minimizes manual errors and speeds up the deployment cycle.

4.2. AWS CodePipeline Orchestration

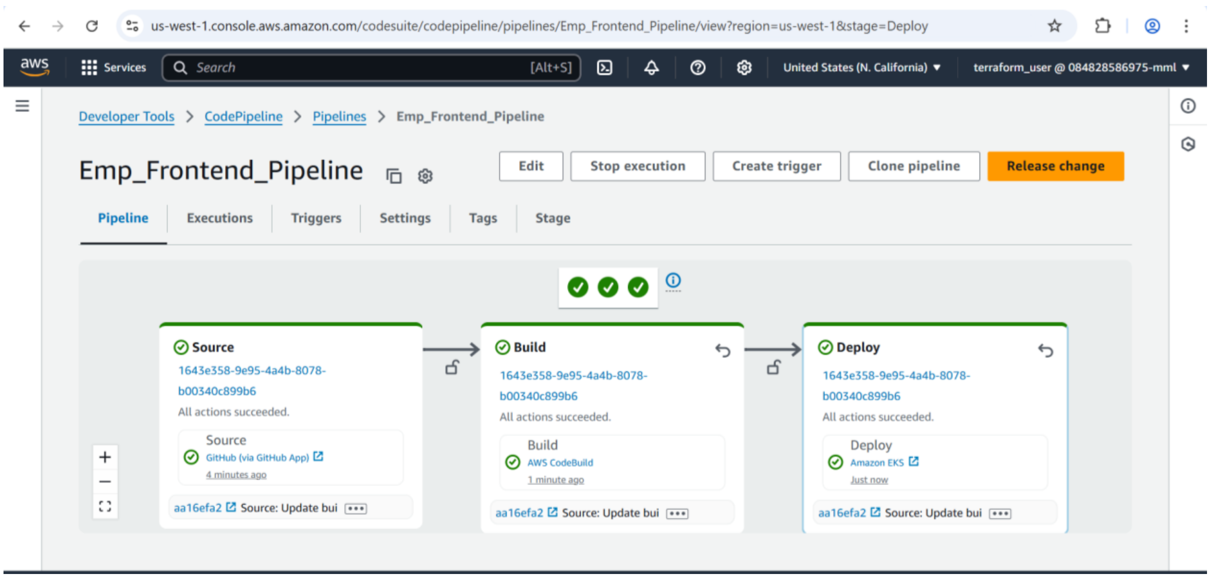
AWS CodePipeline serves as the orchestrator for the entire CI/CD workflow. Separate pipelines are set up for:

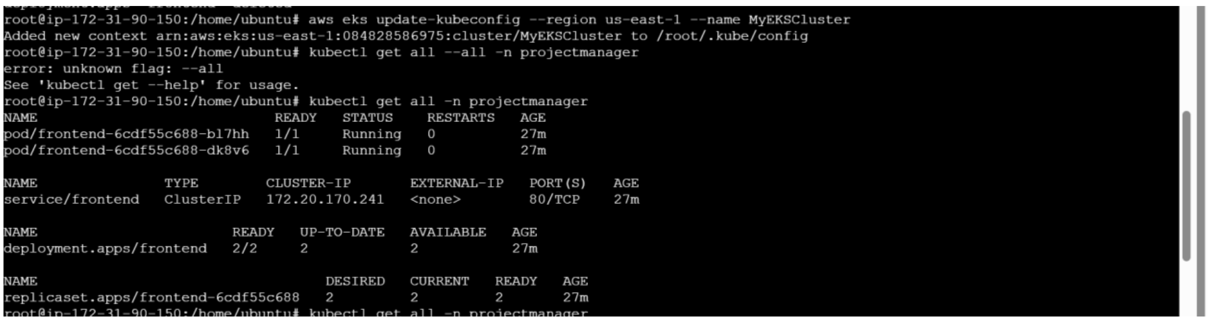
- **Infrastructure Pipelines:** One for CloudFormation (targeting us-west-1) and one for Terraform (targeting us-west-2). These are triggered by changes in their respective IaC repositories.
- **Application Pipelines:** Single pipeline for the application (Angular + Spring Boot), which builds and deploys to EKS clusters in both regions. This pipeline is triggered by changes in the application's source code.

4.2.1. Pipeline Structure (Source, Build, Deploy Stages)

Each pipeline typically consists of the following stages:

- **Source Stage:** Pulls code from a Git repository (e.g., AWS CodeCommit, GitHub).
- **Build Stage:** Uses AWS CodeBuild to compile code, run tests, and perform other build-related tasks.
- **Deploy Stage:** Deploys the built artifacts to the target environment (EKS cluster for applications, AWS accounts for IaC).


us-west-1.console.aws.amazon.com/codesuite/codepipeline/pipelines/Emp_Frontend_Pipeline/view?region=us-west-1&stage=Deploy
Services
Search [Alt+S]
United States (N. California)
terraform_user @ 084828586975-mml
Developer Tools > CodePipeline > Pipelines > Emp_Frontend_Pipeline
Emp_Frontend_Pipeline
Edit
Stop execution
Create trigger
Clone pipeline
Release change
Pipeline
Executions
Triggers
Settings
Tags
Stage
Source
1643e358-9e95-4a4b-8078-b00340c899b6
All actions succeeded.
Source
GitHub (via GitHub App)
4 minutes ago
aa16efa2 Source: Update bui
Build
1643e358-9e95-4a4b-8078-b00340c899b6
All actions succeeded.
Build
AWS CodeBuild
1 minute ago
aa16efa2 Source: Update bui
Deploy
1643e358-9e95-4a4b-8078-b00340c899b6
All actions succeeded.
Deploy
Amazon EKS
Just now
aa16efa2 Source: Update bui



```
root@ip-172-31-90-150:/home/ubuntu# aws eks update-kubeconfig --region us-east-1 --name MyEKSCluster
Added new context arn:aws:eks:us-east-1:084828586975:cluster/MyEKSCluster to /root/.kube/config
root@ip-172-31-90-150:/home/ubuntu# kubectl get all --all -n projectmanager
error: unknown flag: --all
See 'kubectl get --help' for usage.
root@ip-172-31-90-150:/home/ubuntu# kubectl get all -n projectmanager
NAME                              READY   STATUS    RESTARTS   AGE
pod/frontend-6cdf55c688-b17hh    1/1     Running   0          27m
pod/frontend-6cdf55c688-dk8v6    1/1     Running   0          27m

NAME               TYPE        CLUSTER-IP      EXTERNAL-IP   PORT(S)   AGE
service/frontend   ClusterIP   172.20.170.241  <none>        80/TCP    27m

NAME                       READY   UP-TO-DATE   AVAILABLE   AGE
deployment.apps/frontend   2/2     2            2           27m

NAME                                  DESIRED   CURRENT   READY   AGE
replicaset.apps/frontend-6cdf55c688   2         2         2       27m
root@ip-172-31-90-150:/home/ubuntu# kubectl get all -n projectmanager
```

4.3. CodeBuild Configuration (buildspec.yml)

The buildspec.yml file defines the build commands and artifacts for CodeBuild.

4.3.1. Code Quality Analysis with SonarQube

Within the buildspec.yml, a step is included to perform static code analysis using SonarQube.

- The CodeBuild environment is configured to have access to the SonarQube server.
- Commands are executed to run the SonarQube scanner against the source code.
- Quality gates can be configured in SonarQube to fail the build if code quality metrics (e.g., critical bugs, high-security vulnerabilities) fall below predefined thresholds.

4.3.2. Docker Image Build and Push to ECR

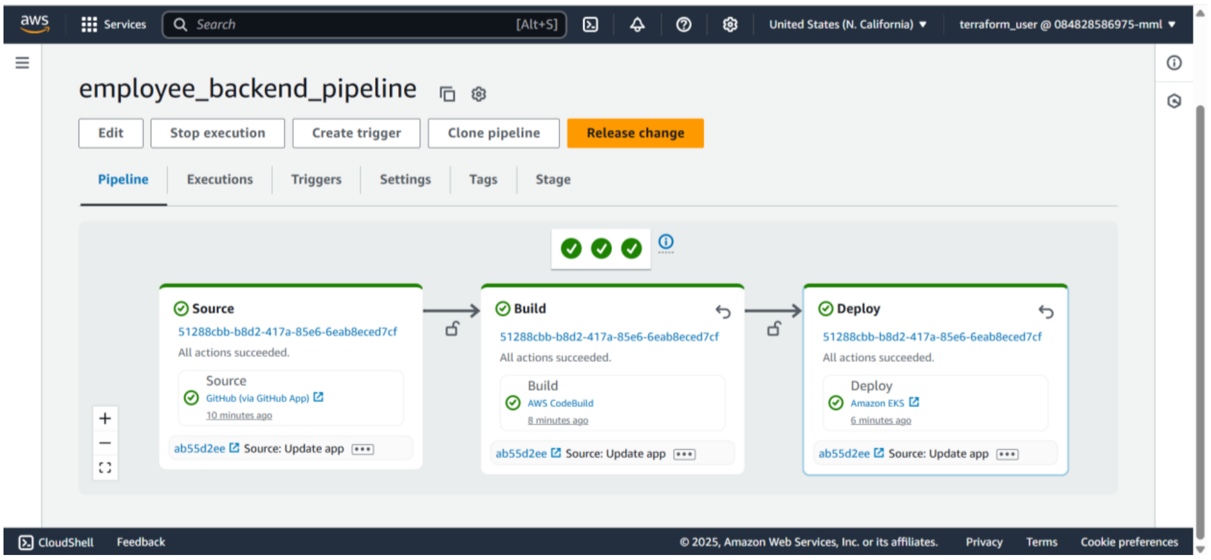
After successful code analysis and compilation, the buildspec.yml builds Docker images for both the Angular frontend and Spring Boot backend.

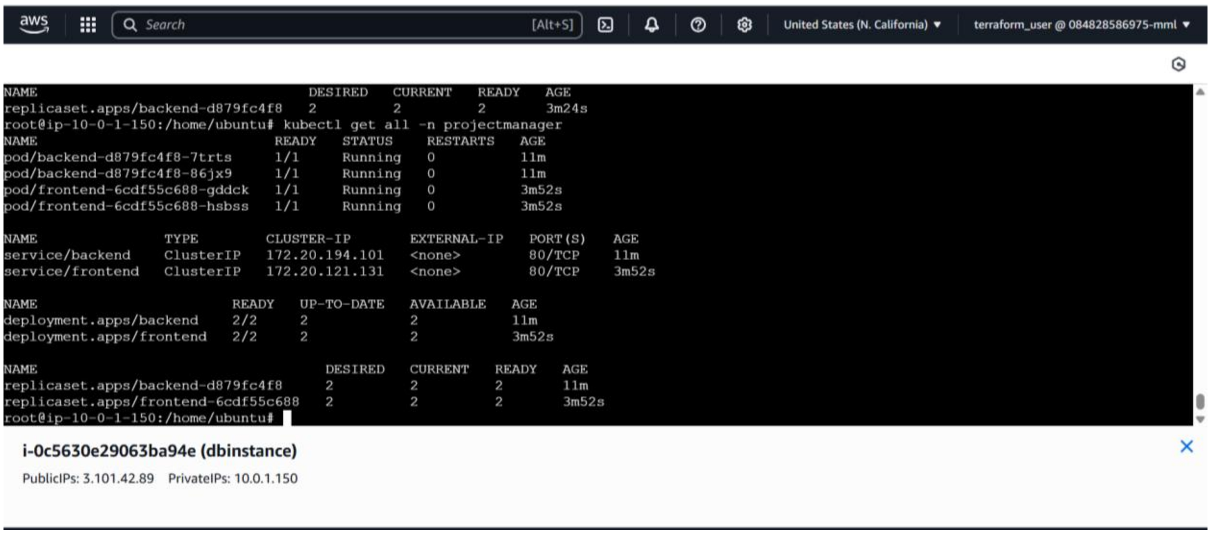
- The images are tagged appropriately (e.g., with commit ID or pipeline execution ID).
- AWS CLI commands are used to authenticate with ECR.
- The built Docker images are pushed to their respective Amazon ECR repositories.

4.3.3. Kubernetes Manifest Application (EKS Deployment)

The final step in the application's buildspec.yml (or a dedicated deploy stage) involves applying Kubernetes manifests to the EKS cluster.

- kubectl is configured to connect to the target EKS cluster.
- Kubernetes deployment files (deployment.yaml, service.yaml, ingress.yaml) are updated with the new ECR image tags.
- kubectl apply commands are executed to deploy the updated applications to the EKS cluster.





Search [Alt+S]
United States (N. California)
terraform_user @ 084828586975-mml

```
NAME                                DESIRED   CURRENT   READY   AGE
replicaset.apps/backend-d879fc4f8   2         2         2       3m24s
root@ip-10-0-1-150:/home/ubuntu# kubectl get all -n projectmanager
NAME                               READY   STATUS    RESTARTS   AGE
pod/backend-d879fc4f8-7trts        1/1     Running   0          11m
pod/backend-d879fc4f8-86jx9        1/1     Running   0          11m
pod/frontend-6cdf55c688-gddck      1/1     Running   0          3m52s
pod/frontend-6cdf55c688-hsbss      1/1     Running   0          3m52s

NAME               TYPE        CLUSTER-IP       EXTERNAL-IP   PORT(S)   AGE
service/backend    ClusterIP   172.20.194.101   <none>        80/TCP    11m
service/frontend   ClusterIP   172.20.121.131   <none>        80/TCP    3m52s

NAME                       READY   UP-TO-DATE   AVAILABLE   AGE
deployment.apps/backend    2/2     2            2           11m
deployment.apps/frontend   2/2     2            2           3m52s

NAME                                  DESIRED   CURRENT   READY   AGE
replicaset.apps/backend-d879fc4f8     2         2         2       11m
replicaset.apps/frontend-6cdf55c688   2         2         2       3m52s
root@ip-10-0-1-150:/home/ubuntu#
```

i-0c5630e29063ba94e (dbinstance)

PublicIPs: 3.101.42.89 PrivateIPs: 10.0.1.150

5. Application Details & Connectivity

5.1. Frontend Application (Angular)

The Angular application provides the user interface. It is configured to communicate with the Spring Boot backend via HTTP/S requests.

5.2. Backend Application (Spring Boot)

The Spring Boot application acts as the API layer, processing requests from the Angular frontend. It contains the business logic and interacts with the Amazon RDS database for data persistence.

5.3. Database (AWS RDS)

An Amazon RDS instance (e.g., PostgreSQL, MySQL) serves as the primary data store. It's configured for high availability (Multi-AZ) and automated backups.

5.4. Inter-Application Communication

5.4.1. Frontend to Backend (environment.prod.ts / environment.ts)

The Angular frontend's build process incorporates environment-specific configuration files (environment.ts for development, environment.prod.ts for production). These files contain the URL/endpoint of the Spring Boot backend API. During the build, the correct environment file is used to ensure the frontend connects to the appropriate backend in each deployed region.

5.4.2. Backend to RDS (application.properties)

The Spring Boot backend connects to the Amazon RDS instance using connection details provided in its application.properties file. This includes:

- spring.datasource.url: The JDBC URL with the RDS endpoint.
- spring.datasource.username: Database username.
- spring.datasource.password: Database password.

5.5. External Access: Ingress NGINX Controller

An Ingress NGINX Controller is deployed within each EKS cluster to manage external access to the applications.

5.5.1. Ingress.yaml for Path-Based Routing

The Ingress.yaml Kubernetes manifest defines rules for routing incoming HTTP/S traffic. Path-based routing is used to direct requests:

- /api/*: Routes traffic to the Spring Boot backend service.
- /: Routes all other traffic to the Angular frontend service.

5.5.2. ALB Integration with Ingress Controller

The Ingress NGINX Controller automatically provisions and manages an AWS Application Load Balancer (ALB) to expose the application to the internet. The ALB's DNS name is then used for public access.

5.6. Multi-Region Access: Route 53 Failover Routing

Amazon Route 53 is configured with a failover routing policy to ensure high availability across regions.

- **Health Checks:** Route 53 performs health checks on the ALBs in both us-west-1 and us-west-2.
- **Primary/Secondary:** One region is designated as primary, and the other as secondary.
- **Failover:** If the primary region's ALB fails its health checks, Route 53 automatically directs all traffic to the secondary region's ALB, providing seamless disaster recovery.

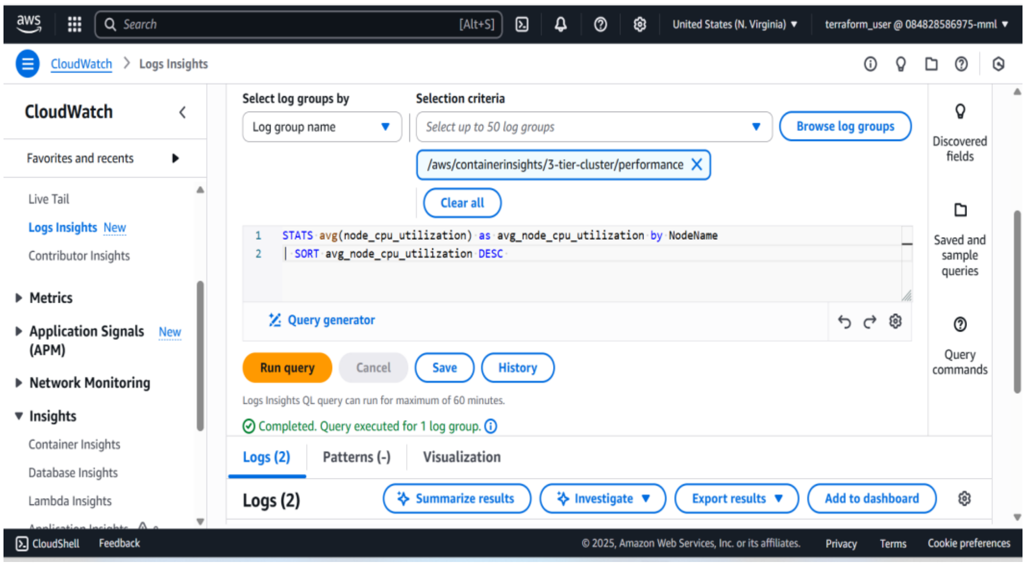
6. Monitoring, Logging, and Alerting

6.1 EKS Container Insights Dashboard

AWS CloudWatch Container Insights can be enabled for your EKS clusters (both us-west-1 and us-west-2) to monitor performance and operational health of pods, containers, and nodes.

Container Insights Metrics Available:

- pod_cpu_utilization
- container_memory_utilization
- node_cpu_utilization



Select log groups by: Log group name

Selection criteria: Select up to 50 log groups

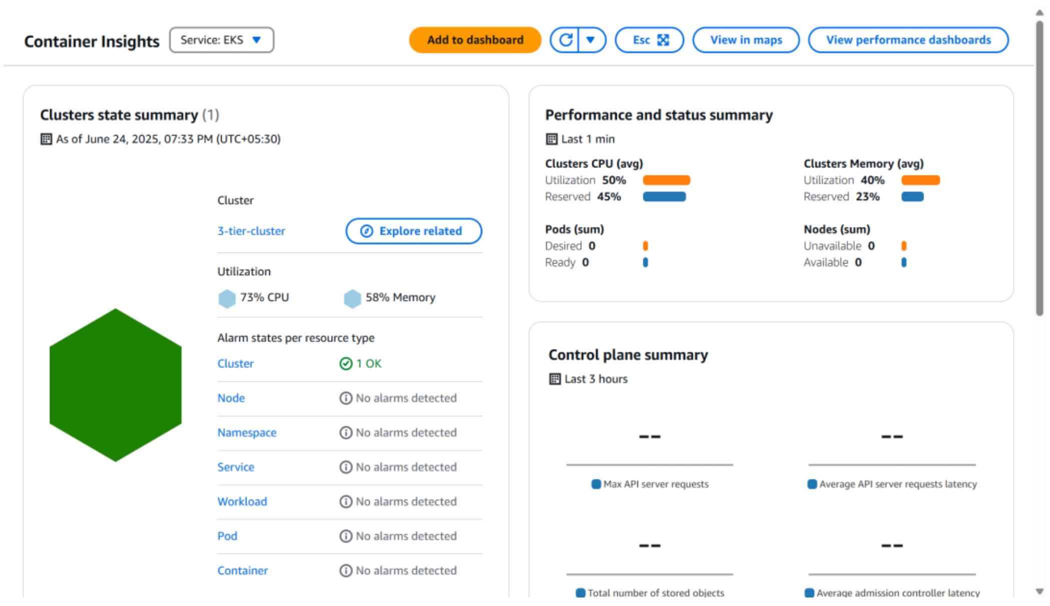
/aws/containerinsights/3-tier-cluster/performance

```
STATS avg(node_cpu_utilization) as avg_node_cpu_utilization by NodeName
| SORT avg_node_cpu_utilization DESC
```

Logs Insights QL query can run for maximum of 60 minutes.

Completed. Query executed for 1 log group.

© 2025, Amazon Web Services, Inc. or its affiliates.



Container Insights Service: EKS Add to dashboard Esc View in maps View performance dashboards

Clusters state summary (1)
As of June 24, 2025, 07:33 PM (UTC+05:30)

Cluster
3-tier-cluster Explore related

Utilization
73% CPU 58% Memory

Alarm states per resource type

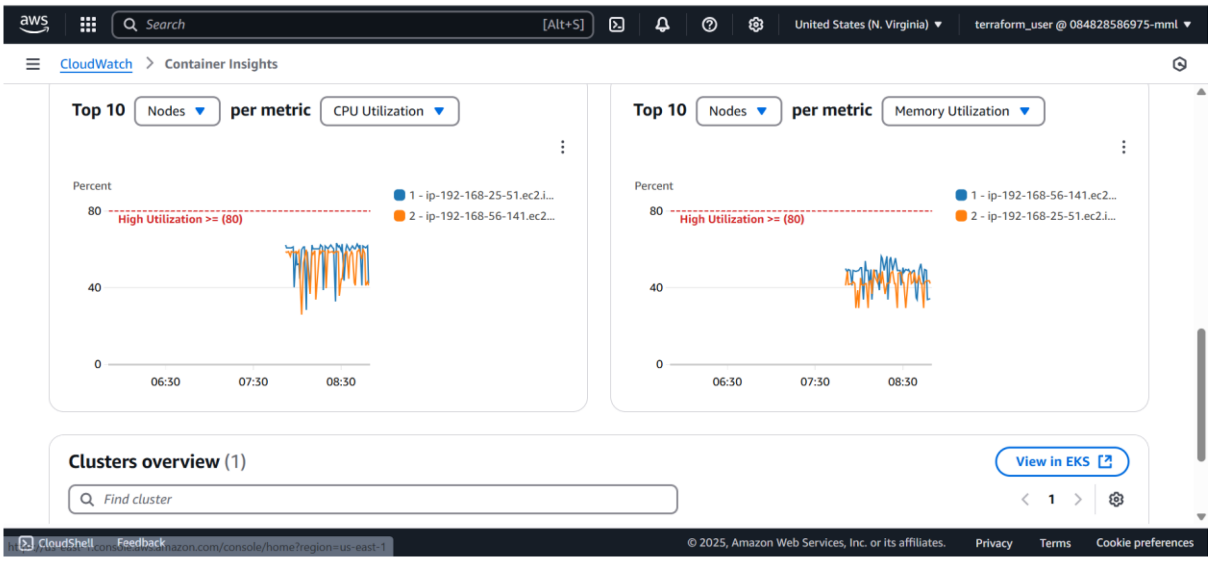
Cluster	✓ 1 OK
Node	No alarms detected
Namespace	No alarms detected
Service	No alarms detected
Workload	No alarms detected
Pod	No alarms detected
Container	No alarms detected

Performance and status summary
Last 1 min

Clusters CPU (avg)	**Clusters Memory (avg)**
Utilization 50%	Utilization 40%
Reserved 45%	Reserved 23%
Pods (sum)	**Nodes (sum)**
Desired 0	Unavailable 0
Ready 0	Available 0

Control plane summary
Last 3 hours

--	--
Max API server requests	Average API server requests latency
--	--
Total number of stored objects	Average admission controller latency



United States (N. Virginia) terraform_user @ 084828586975-mml

CloudWatch > Container Insights

Top 10 Nodes per metric CPU Utilization

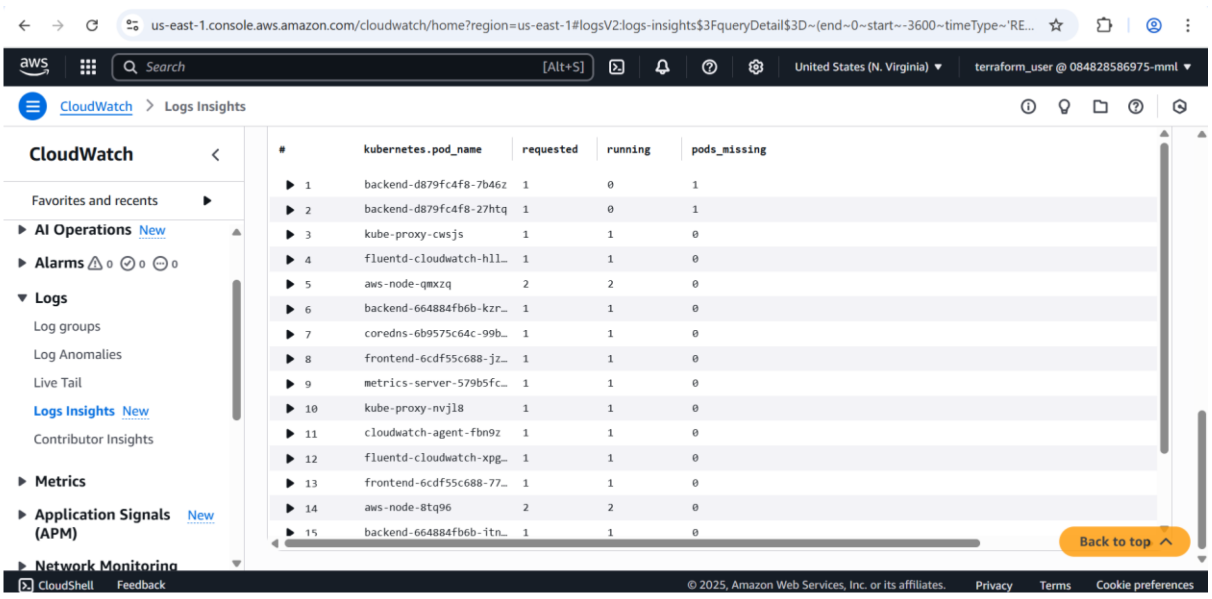
Percent
80
40
0
High Utilization >= (80)
06:30
07:30
08:30
1 - ip-192-168-25-51.ec2.i...
2 - ip-192-168-56-141.ec2...

Top 10 Nodes per metric Memory Utilization

Percent
80
40
0
High Utilization >= (80)
06:30
07:30
08:30
1 - ip-192-168-56-141.ec2...
2 - ip-192-168-25-51.ec2.i...

Clusters overview (1) View in EKS
Find cluster

CloudShell Feedback © 2025, Amazon Web Services, Inc. or its affiliates. Privacy Terms Cookie preferences



us-east-1.console.aws.amazon.com/cloudwatch/home?region=us-east-1#logsV2:logs-insights$3FqueryDetail$3D~(end~0~start~-3600~timeType~'RE...

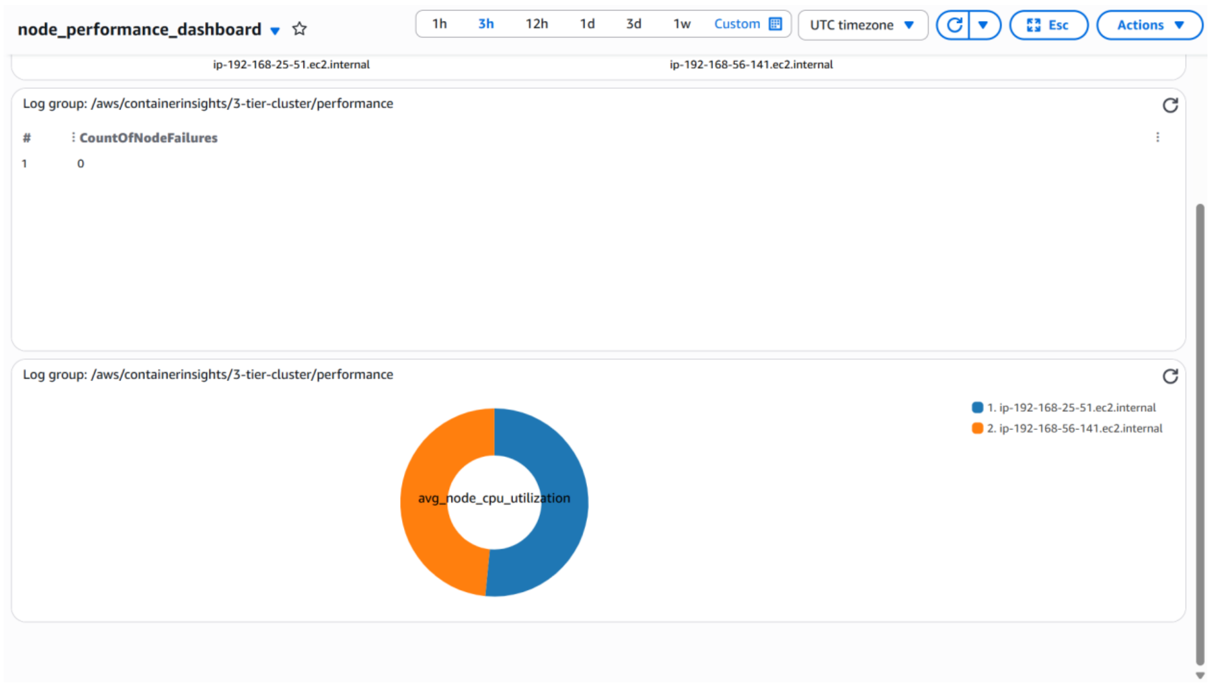
United States (N. Virginia) terraform_user @ 084828586975-mml

CloudWatch > Logs Insights

#	kubernetes.pod_name	requested	running	pods_missing
1	backend-d879fc4f8-7b46z	1	0	1
2	backend-d879fc4f8-27htq	1	0	1
3	kube-proxy-cwsjs	1	1	0
4	fluentd-cloudwatch-hll…	1	1	0
5	aws-node-qmxzq	2	2	0
6	backend-664884fb6b-kzr…	1	1	0
7	coredns-6b9575c64c-99b…	1	1	0
8	frontend-6cdf55c688-jz…	1	1	0
9	metrics-server-579b5fc…	1	1	0
10	kube-proxy-nvjl8	1	1	0
11	cloudwatch-agent-fbn9z	1	1	0
12	fluentd-cloudwatch-xpg…	1	1	0
13	frontend-6cdf55c688-77…	1	1	0
14	aws-node-8tq96	2	2	0
15	backend-664884fb6b-itn…	1	1	0

Back to top

CloudShell Feedback © 2025, Amazon Web Services, Inc. or its affiliates. Privacy Terms Cookie preferences

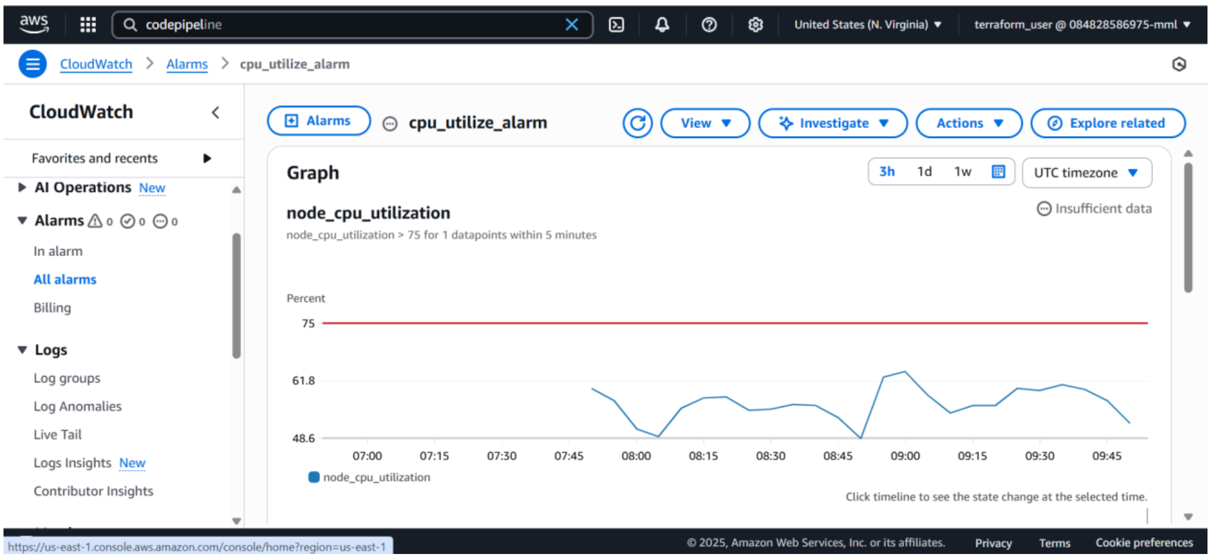

node_performance_dashboard
1h
3h
12h
1d
3d
1w
Custom
UTC timezone
Esc
Actions
ip-192-168-25-51.ec2.internal
ip-192-168-56-141.ec2.internal
Log group: /aws/containerinsights/3-tier-cluster/performance

```
#    CountOfNodeFailures
1    0
```

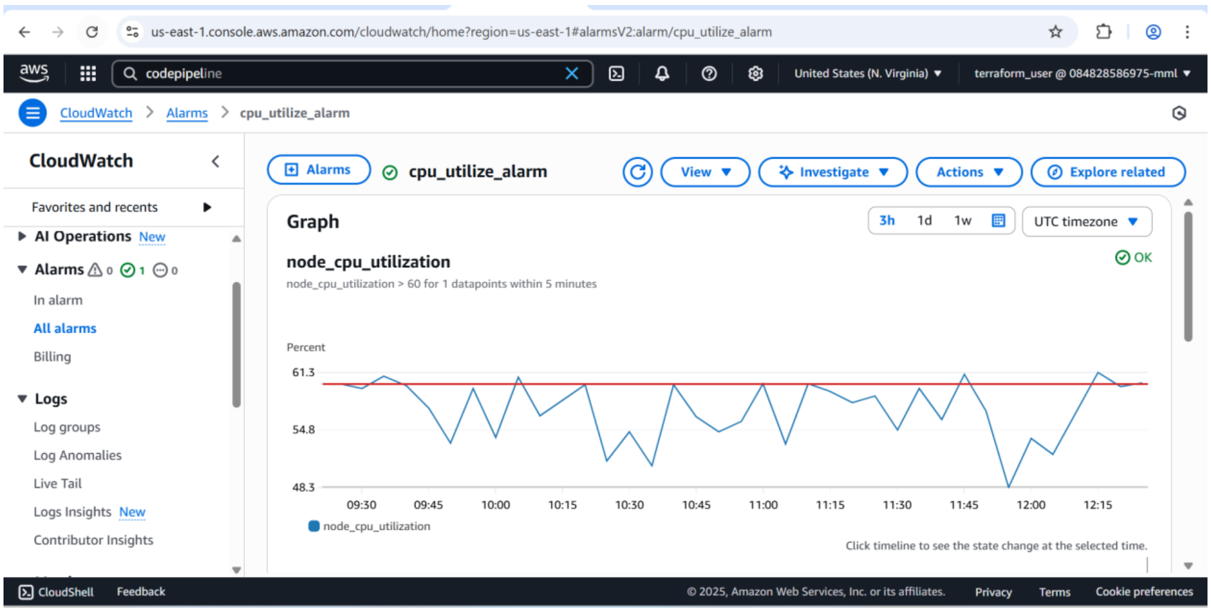
Log group: /aws/containerinsights/3-tier-cluster/performance
avg_node_cpu_utilization
1. ip-192-168-25-51.ec2.internal
2. ip-192-168-56-141.ec2.internal

6.2 SNS Setup

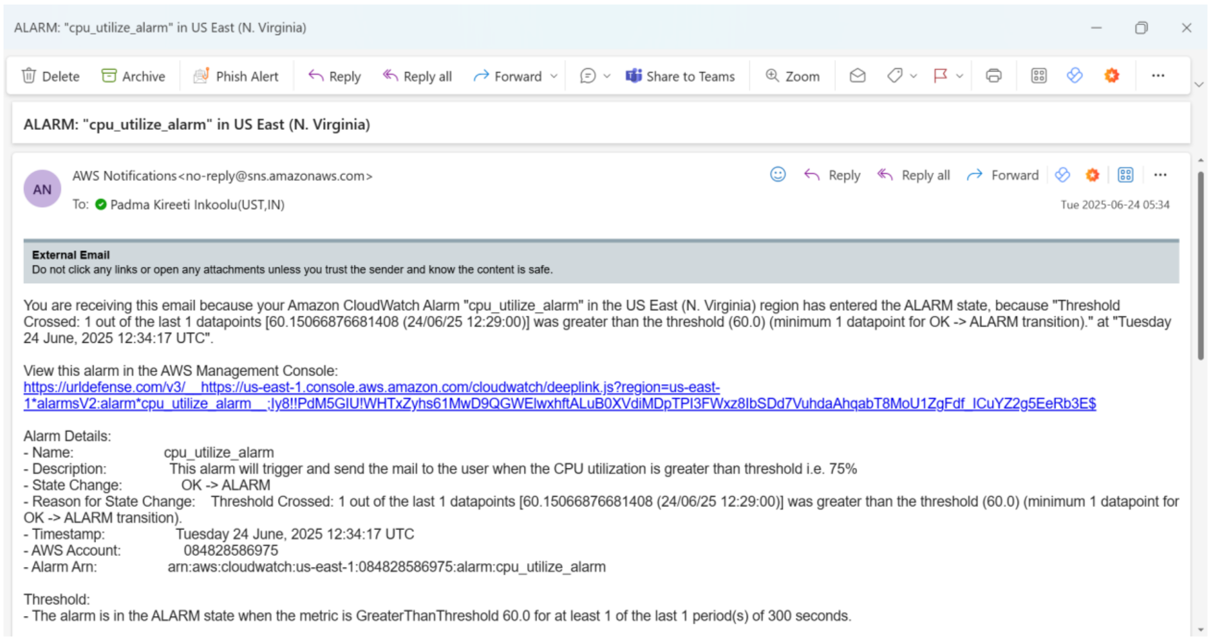
- Go to SNS → Create Topic.
- Add email subscription and confirm the subscription.
- Attach SNS topic to CloudWatch alarms.
- Creating Alarm and notifying through Email from SNS Topic:



After Applying a load:


us-east-1.console.aws.amazon.com/cloudwatch/home?region=us-east-1#alarmsV2:alarm/cpu_utilize_alarm
codepipeline
United States (N. Virginia)
terraform_user @ 084828586975-mml
CloudWatch > Alarms > cpu_utilize_alarm
cpu_utilize_alarm
View
Investigate
Actions
Explore related
Graph
3h 1d 1w
UTC timezone
node_cpu_utilization
node_cpu_utilization > 60 for 1 datapoints within 5 minutes
OK
Percent
61.3
54.8
48.3
09:30 09:45 10:00 10:15 10:30 10:45 11:00 11:15 11:30 11:45 12:00 12:15
node_cpu_utilization
Click timeline to see the state change at the selected time.

Mail came from SNS Notification service:



ALARM: "cpu_utilize_alarm" in US East (N. Virginia)

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

To: Padma Kireeti Inkoolu(UST,IN)

Tue 2025-06-24 05:34

External Email
Do not click any links or open any attachments unless you trust the sender and know the content is safe.

You are receiving this email because your Amazon CloudWatch Alarm "cpu_utilize_alarm" in the US East (N. Virginia) region has entered the ALARM state, because "Threshold Crossed: 1 out of the last 1 datapoints [60.15066876681408 (24/06/25 12:29:00)] was greater than the threshold (60.0) (minimum 1 datapoint for OK -> ALARM transition)." at "Tuesday 24 June, 2025 12:34:17 UTC".

View this alarm in the AWS Management Console:
https://urldefense.com/v3/__https://us-east-1.console.aws.amazon.com/cloudwatch/deeplink.js?region=us-east-1*alarmsV2:alarm*cpu_utilize_alarm__;Iy8!!PdM5GIU!WHTxZyhs61MwD9QGWElwxhftALuB0XVdiMDpTPI3FWxz8lbSDd7VuhdaAhqabT8MoU1ZgFdf_ICuYZ2g5EeRb3E$

Alarm Details:
- Name: cpu_utilize_alarm
- Description: This alarm will trigger and send the mail to the user when the CPU utilization is greater than threshold i.e. 75%
- State Change: OK -> ALARM
- Reason for State Change: Threshold Crossed: 1 out of the last 1 datapoints [60.15066876681408 (24/06/25 12:29:00)] was greater than the threshold (60.0) (minimum 1 datapoint for OK -> ALARM transition).
- Timestamp: Tuesday 24 June, 2025 12:34:17 UTC
- AWS Account: 084828586975
- Alarm Arn: arn:aws:cloudwatch:us-east-1:084828586975:alarm:cpu_utilize_alarm

Threshold:
- The alarm is in the ALARM state when the metric is GreaterThanThreshold 60.0 for at least 1 of the last 1 period(s) of 300 seconds.

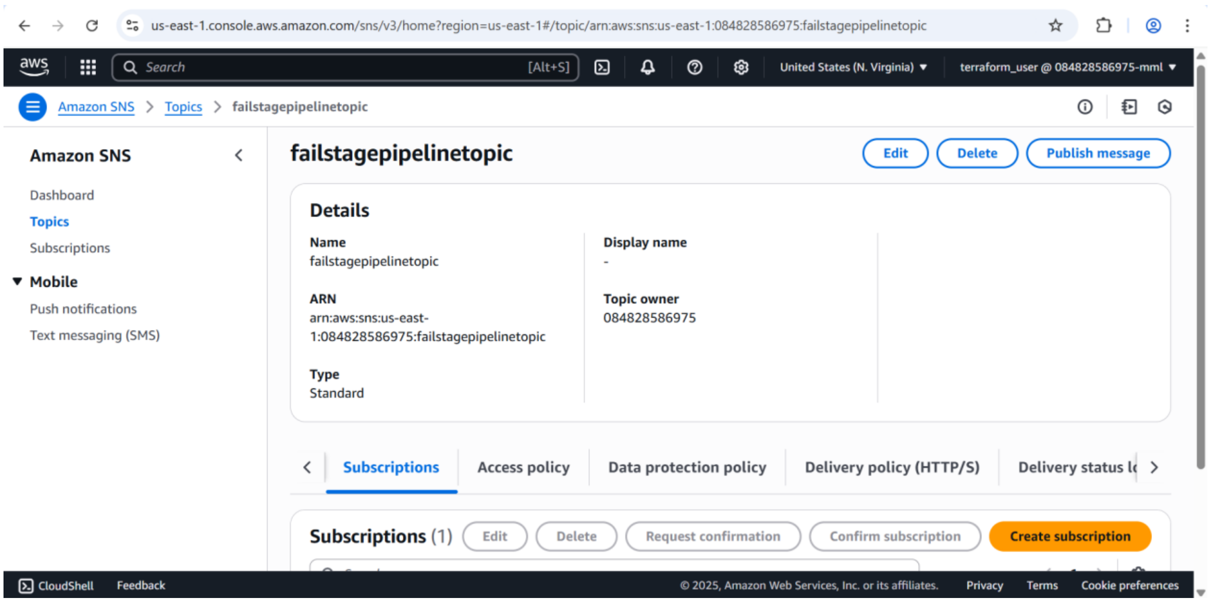
6.3. Process Communication: Pipeline Failure Notification

To ensure immediate awareness of CI/CD pipeline failures, an automated notification system is implemented:

6.3.1. EventBridge Rule for CodePipeline Events

An Amazon EventBridge rule is configured to listen for specific events from AWS CodePipeline.

- **Event Pattern:** CodePipeline Pipeline Execution State Change
- **Detail Type:** CodePipeline Pipeline Execution State Change
- **Status:** FAILED



us-east-1.console.aws.amazon.com/sns/v3/home?region=us-east-1#/topic/arn:aws:sns:us-east-1:084828586975:failstagepipelinetopic

United States (N. Virginia) ▾ terraform_user @ 084828586975-mml ▾

Amazon SNS > Topics > failstagepipelinetopic

Amazon SNS: Dashboard, Topics, Subscriptions, Mobile: Push notifications, Text messaging (SMS)

failstagepipelinetopic — Edit | Delete | Publish message

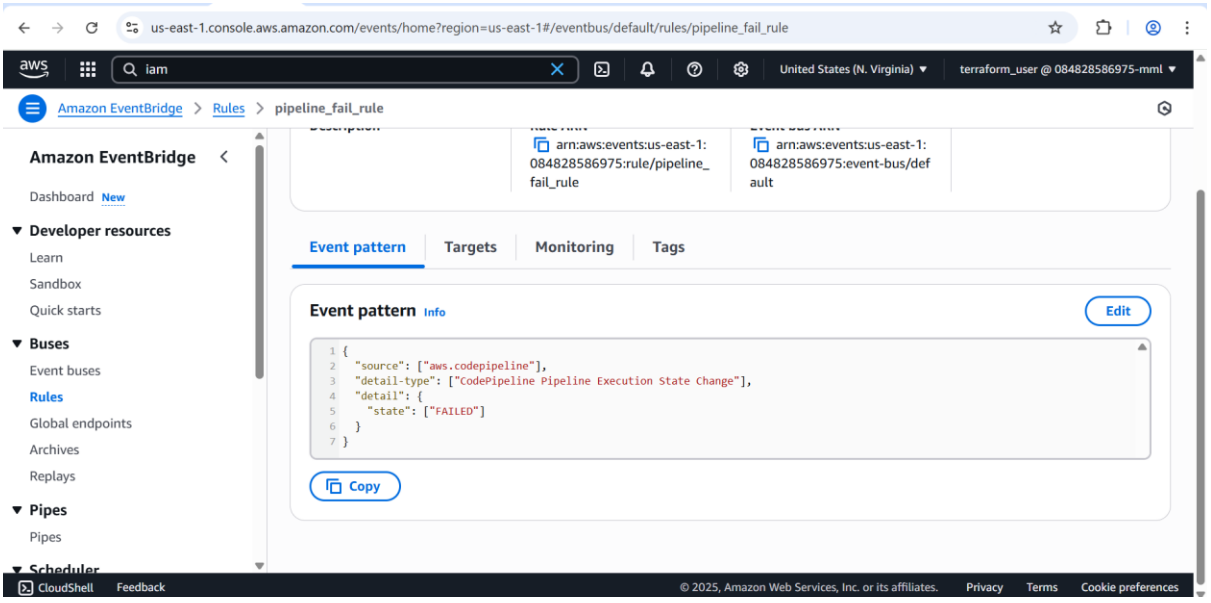
Details

Name	failstagepipelinetopic
Display name	-
ARN	arn:aws:sns:us-east-1:084828586975:failstagepipelinetopic
Topic owner	084828586975
Type	Standard

Subscriptions | Access policy | Data protection policy | Delivery policy (HTTP/S) | Delivery status l

Subscriptions (1) — Edit | Delete | Request confirmation | Confirm subscription | Create subscription

© 2025, Amazon Web Services, Inc. or its affiliates. Privacy Terms Cookie preferences



us-east-1.console.aws.amazon.com/events/home?region=us-east-1#/eventbus/default/rules/pipeline_fail_rule

United States (N. Virginia) ▾ terraform_user @ 084828586975-mml ▾

Amazon EventBridge > Rules > pipeline_fail_rule

Rule ARN: arn:aws:events:us-east-1:084828586975:rule/pipeline_fail_rule

Event bus ARN: arn:aws:events:us-east-1:084828586975:event-bus/default

Event pattern | Targets | Monitoring | Tags

Event pattern Info

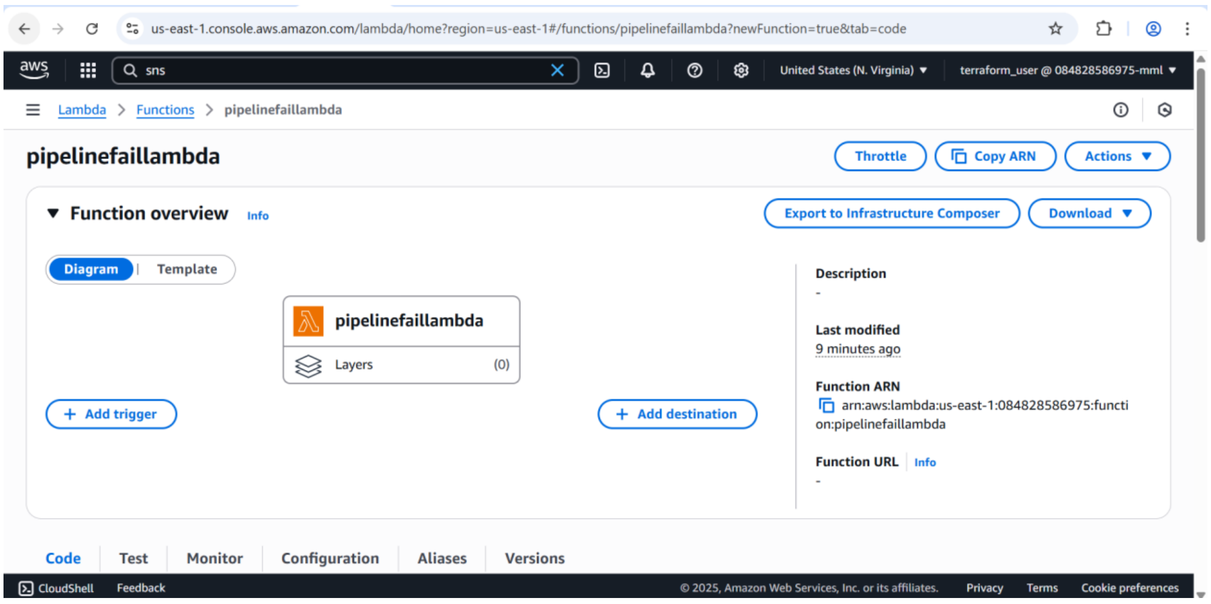
```
{
  "source": ["aws.codepipeline"],
  "detail-type": ["CodePipeline Pipeline Execution State Change"],
  "detail": {
    "state": ["FAILED"]
  }
}
```

© 2025, Amazon Web Services, Inc. or its affiliates. Privacy Terms Cookie preferences

6.3.2. AWS Lambda Function for Notification Logic

When the EventBridge rule detects a pipeline failure, it invokes an AWS Lambda function.

- The Lambda function receives the detailed event payload from CodePipeline.
- It extracts relevant information (pipeline name, execution ID, failed stage/action, error message).
- This information is formatted into a user-friendly message.


us-east-1.console.aws.amazon.com/lambda/home?region=us-east-1#/functions/pipelinefaillambda?newFunction=true&tab=code
United States (N. Virginia)
terraform_user @ 084828586975-mml
Lambda > Functions > pipelinefaillambda
Throttle
Copy ARN
Actions
Export to Infrastructure Composer
Download
Diagram
Template
Layers (0)
Add trigger
Add destination
Code
Test
Monitor
Configuration
Aliases
Versions

pipelinefaillambda

Function overview

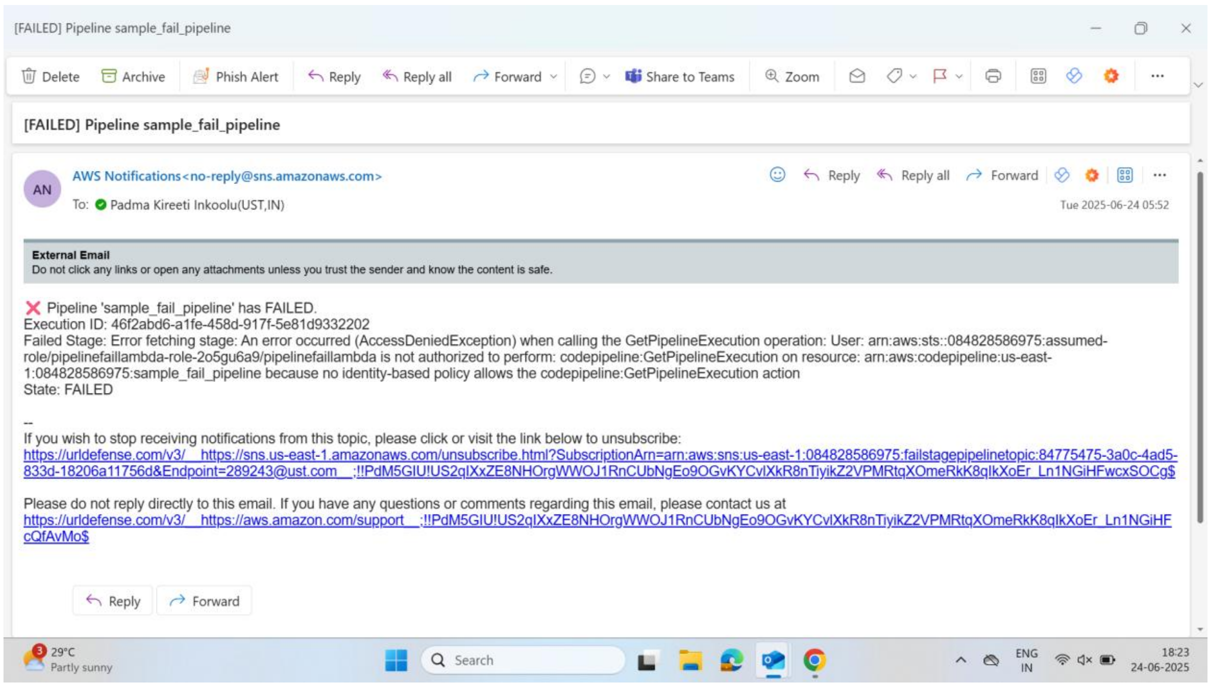
Description
-

Last modified
9 minutes ago

Function ARN
arn:aws:lambda:us-east-1:084828586975:function:pipelinefaillambda

Function URL
-

Mail From SNS:



[FAILED] Pipeline sample_fail_pipeline

AWS Notifications<no-reply@sns.amazonaws.com>
To: Padma Kireeti Inkoolu(UST,IN)
Tue 2025-06-24 05:52

External Email
Do not click any links or open any attachments unless you trust the sender and know the content is safe.

❌ Pipeline 'sample_fail_pipeline' has FAILED.
Execution ID: 46f2abd6-a1fe-458d-917f-5e81d9332202
Failed Stage: Error fetching stage: An error occurred (AccessDeniedException) when calling the GetPipelineExecution operation: User: arn:aws:sts::084828586975:assumed-role/pipelinefaillambda-role-2o5gu6a9/pipelinefaillambda is not authorized to perform: codepipeline:GetPipelineExecution on resource: arn:aws:codepipeline:us-east-1:084828586975:sample_fail_pipeline because no identity-based policy allows the codepipeline:GetPipelineExecution action
State: FAILED

--
If you wish to stop receiving notifications from this topic, please click or visit the link below to unsubscribe:
https://urldefense.com/v3/__https://sns.us-east-1.amazonaws.com/unsubscribe.html?SubscriptionArn=arn:aws:sns:us-east-1:084828586975:failstagepipelinetopic:84775475-3a0c-4ad5-833d-18206a11756d&Endpoint=289243@ust.com__;!!PdM5GIU!US2qIXxZE8NHOrgWWOJ1RnCUbNgEo9OGvKYCvlXkR8nTiyikZ2VPMRtqXOmeRkK8qIkXoEr_Ln1NGiHFwcxSOCg$

Please do not reply directly to this email. If you have any questions or comments regarding this email, please contact us at
https://urldefense.com/v3/__https://aws.amazon.com/support__;!!PdM5GIU!US2qIXxZE8NHOrgWWOJ1RnCUbNgEo9OGvKYCvlXkR8nTiyikZ2VPMRtqXOmeRkK8qIkXoEr_Ln1NGiHFcQfAvMo$

7. Backup and Recovery

7.1. AWS Backup Service Configuration

AWS Backup is utilized to centralize and automate backup operations across AWS services. This ensures that critical data and resources can be recovered in case of accidental deletion, corruption, or disaster.

7.2. Backup Plan Details

A comprehensive backup plan is configured within AWS Backup:

7.2.1. Snapshot Frequency (Every 30 Days)

The backup plan specifies a backup rule to take a snapshot of the protected resources every 30 days. This provides regular recovery points.

7.2.2. Retention Period (40 Days)

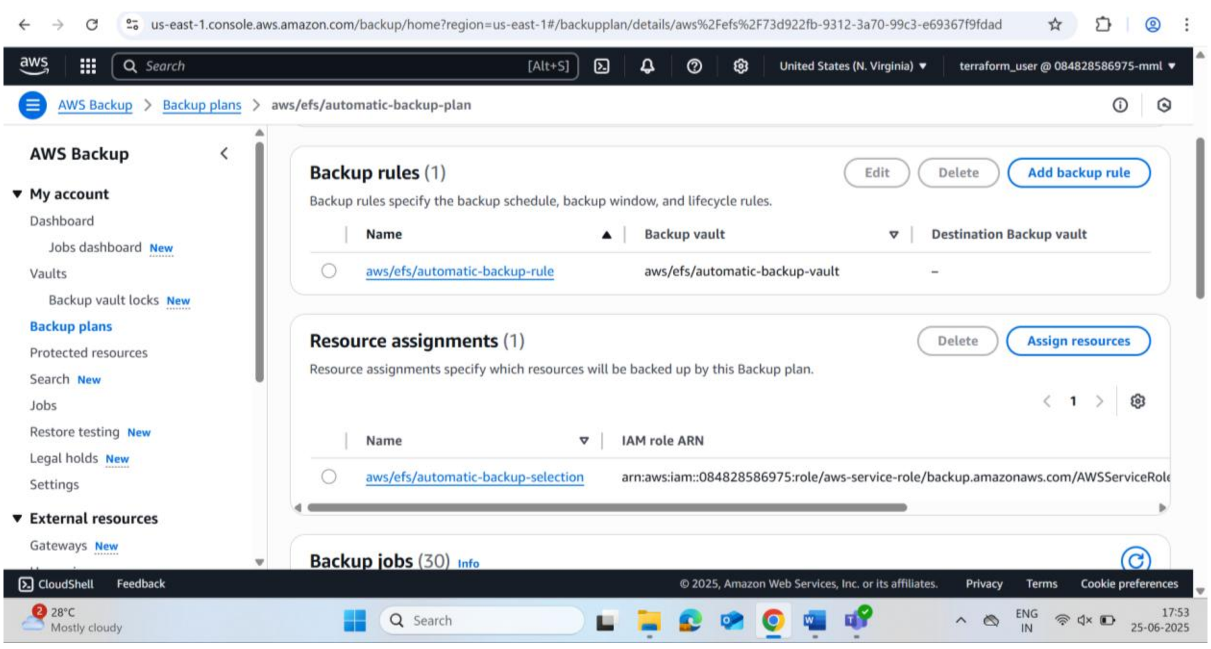
The taken snapshots are retained for a period of 40 days. After 40 days, AWS Backup automatically deletes the recovery point, optimizing storage costs.

7.3. Scope of Backup

The backup plan includes:

- **Amazon RDS:** The primary application database is a key protected resource. Snapshots capture the entire database state.

Created an RDS Vault and Assigned the RDS resource to the Backup Rule.



Backup rules (1)

Backup rules specify the backup schedule, backup window, and lifecycle rules.

Name	Backup vault	Destination Backup vault
aws/efs/automatic-backup-rule	aws/efs/automatic-backup-vault	–

Resource assignments (1)

Resource assignments specify which resources will be backed up by this Backup plan.

Name	IAM role ARN
aws/efs/automatic-backup-selection	arn:aws:iam::084828586975:role/aws-service-role/backup.amazonaws.com/AWSServiceRol

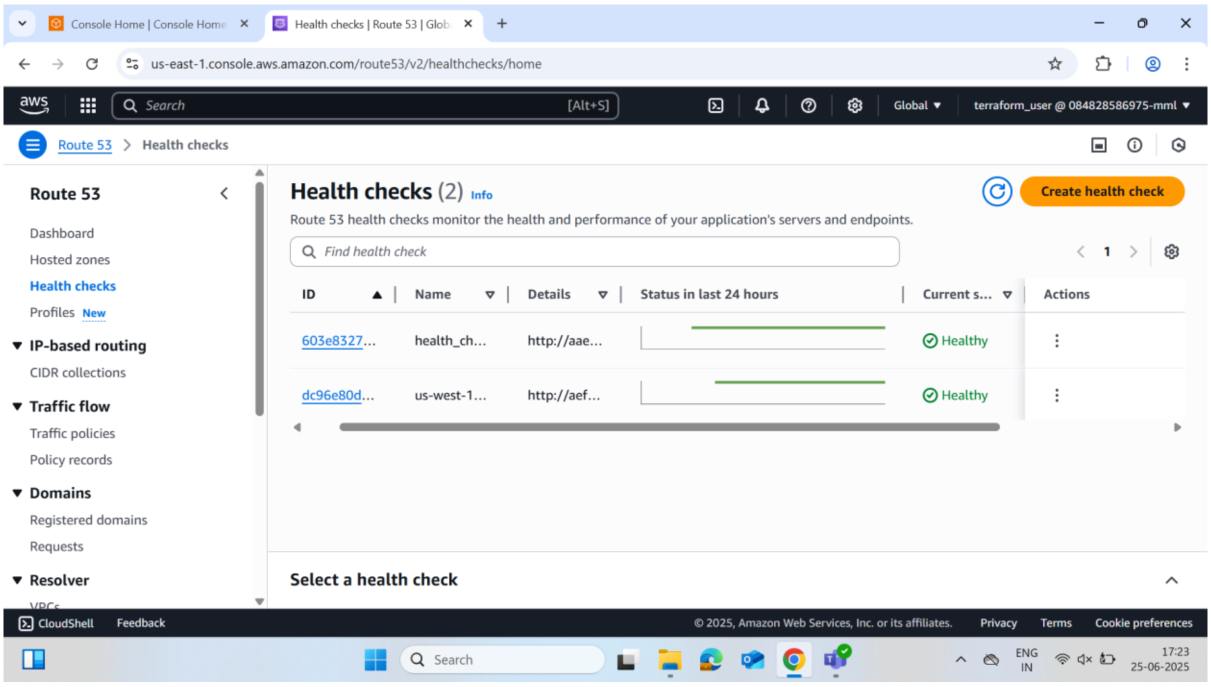
Backup jobs (30) Info

8. Route53 Setup:

Hosted Zone with failover routing policy

ALB endpoints from both regions added as failover targets

Health checks configured to determine regional availability


Health checks (2) Info

Route 53 health checks monitor the health and performance of your application's servers and endpoints.

ID	Name	Details	Status in last 24 hours	Current s...	Actions
603e8327...	health_ch...	http://aae...		Healthy	⋮
dc96e80d...	us-west-1...	http://aef...		Healthy	⋮

9. Security Considerations

9.1. IAM Roles and Policies (Least Privilege)

All AWS services and components are configured with specific IAM roles and policies adhering to the principle of least privilege. This minimizes the blast radius in case of a security compromise. Examples include:

- EKS Service Role
- EKS Node Group Instance Profile Role
- CodePipeline Service Role
- CodeBuild Service Role
- Lambda Execution Role
- ECR Repository Policies

9.2. Security Groups and Network ACLs

Network security is enforced using:

- **Security Groups:** Act as virtual firewalls at the instance or ENI level, controlling inbound and outbound traffic to EKS worker nodes, RDS instances, and Load Balancers.
- **Network ACLs (NACLs):** Optional, stateless firewalls at the subnet level, providing an additional layer of defense.

Final Notes

- The application runs in two AWS regions (us-west-1 primary, us-west-2 secondary) for high availability and disaster recovery.
- CloudFormation and Terraform are used for fully automated infrastructure setup, ensuring consistency and rapid provisioning.
- CI/CD is implemented using CodePipeline, with SonarQube for static code analysis, streamlining the development and deployment workflow.
- Route 53 provides DNS-based failover between regions, offering seamless traffic redirection in case of primary region failure.
- CloudWatch and SNS handle comprehensive monitoring and alerts, notifying teams of performance issues, resource thresholds, and pipeline failures.
- Security is enforced through IAM roles with the principle of least privilege, AWS Secrets Manager for sensitive data, and stringent network security measures.
- AWS Backup ensures automated and policy-driven data protection for the RDS database.
- The entire setup is automated, secure, scalable, and resilient, making it suitable for production-grade web applications.