



DEVOPS & CLOUD PROJECTS

BEGINNER TO ADVANCED

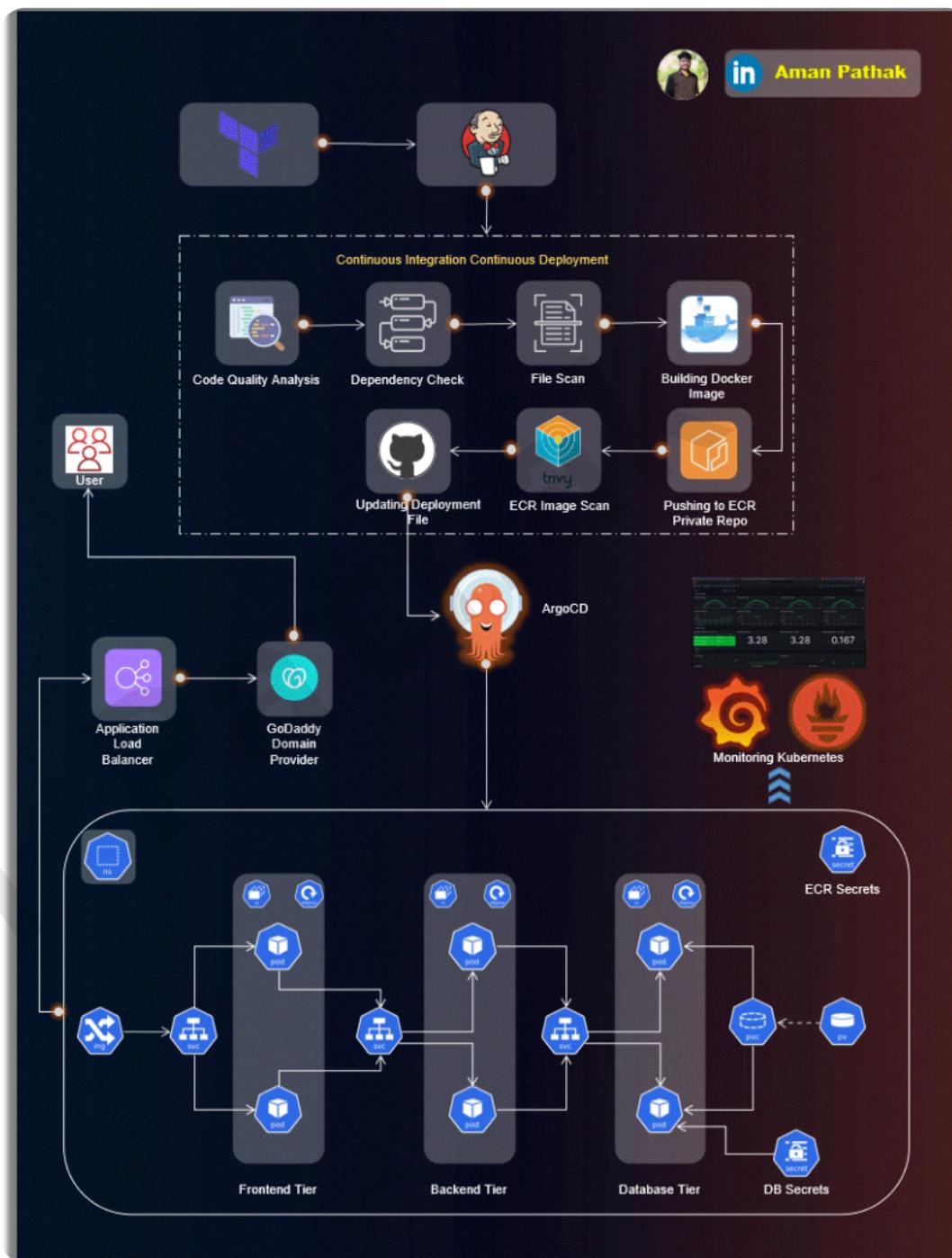
-BY AMAN PATHAK



Featuring 28 Projects across
DevSecOps, Kubernetes, AWS,
IaC, and Monitoring

Advanced Three-Tier K8S Project on AWS (EKS, ArgoCD, Prometheus, Jenkins)

Deploy and manage a three-tier app using microservices on EKS. Features GitOps, CI/CD, observability stack, and scaling strategies – a blueprint for enterprise-grade K8s design.



↗ [BLOG LINK](#)

 [GITHUB REPO](#)

KUBERNETES

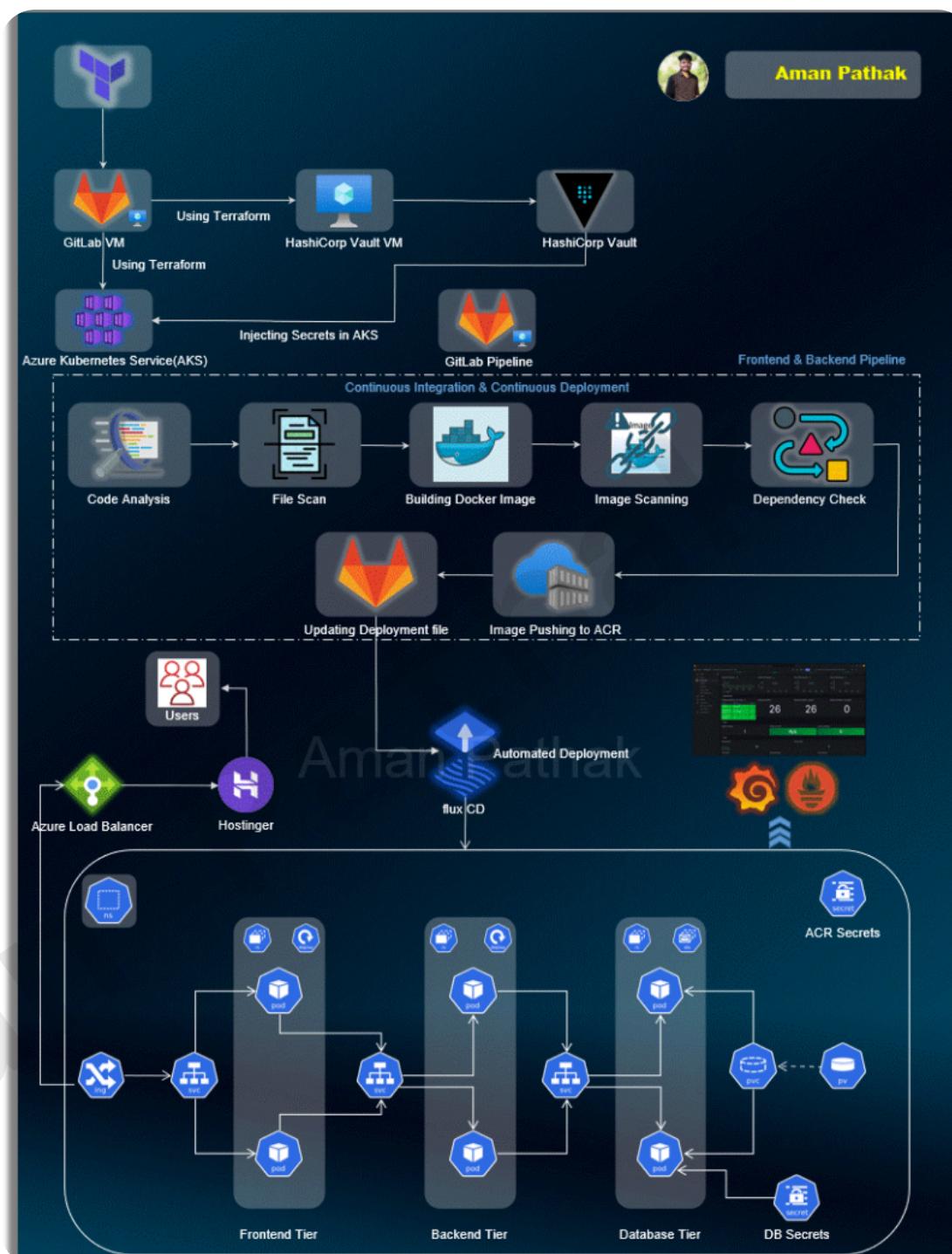
OBSERVABILITY

MICROSERVICES

EKS

Advanced Three-Tier K8s Project on Azure (AKS, FluxCD, GitLab, Vault)

Deploy a secure and scalable three-tier app on Azure Kubernetes Service (AKS) with GitLab CI, GitOps via FluxCD, and secret management using Vault. Cloud-agnostic and production-tested.



↗ [BLOG LINK](#)

[GITHUB REPO](#)

AKS

FLUXCD

GITLAB

VAULT

Deploy a Go App on EKS with Terraform, GitHub Actions & Helm

Deploy a scalable Go-based web app to AWS EKS using IaC with Terraform, CI/CD with GitHub Actions, and Helm for manifest templating—clean, modular, and production-ready setup.



↗ [BLOG LINK](#)

 [GITHUB REPO](#)

GO

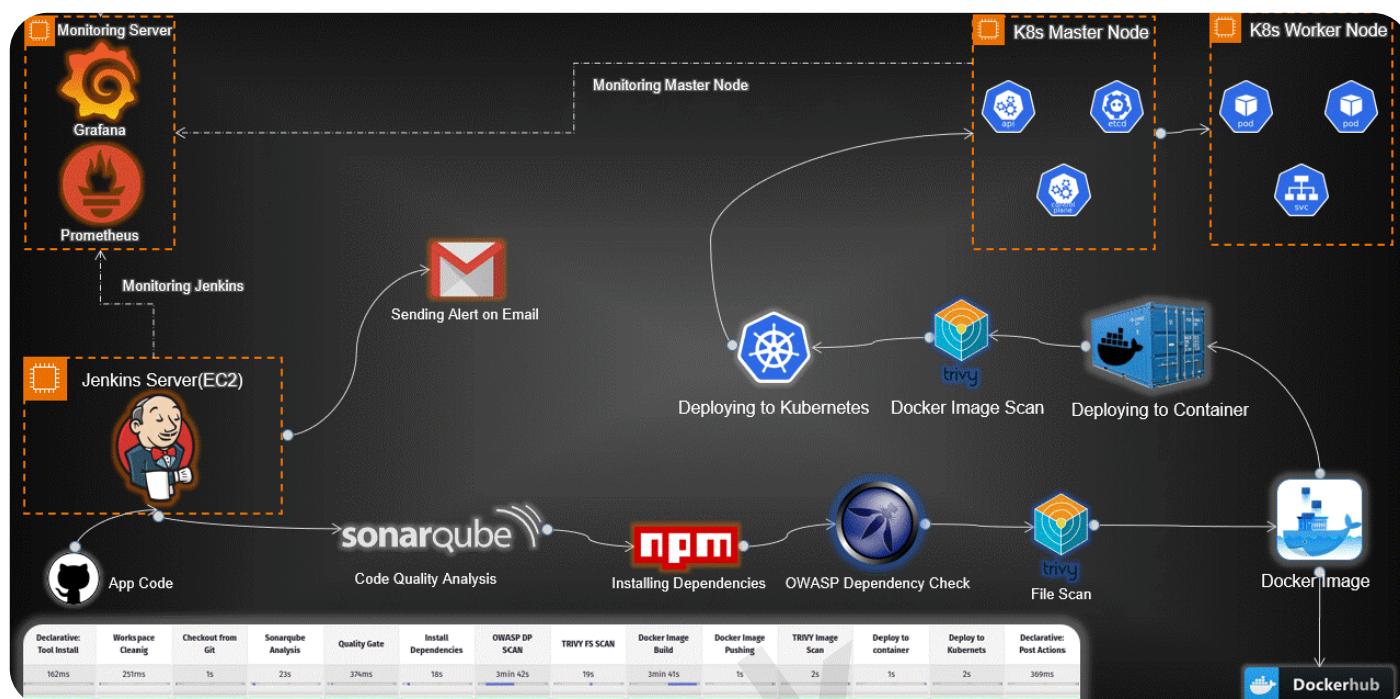
HELM

TERRAFORM

GITHUBACTIONS

DevSecOps EKS Project: Jenkins, Terraform & ArgoCD- Deploy Netflix Clone

Build a secure DevOps pipeline with Jenkins, ArgoCD, and Terraform that includes image scanning, IAM roles, and GitOps on AWS EKS. A full security-focused delivery setup.



[BLOG LINK](#)

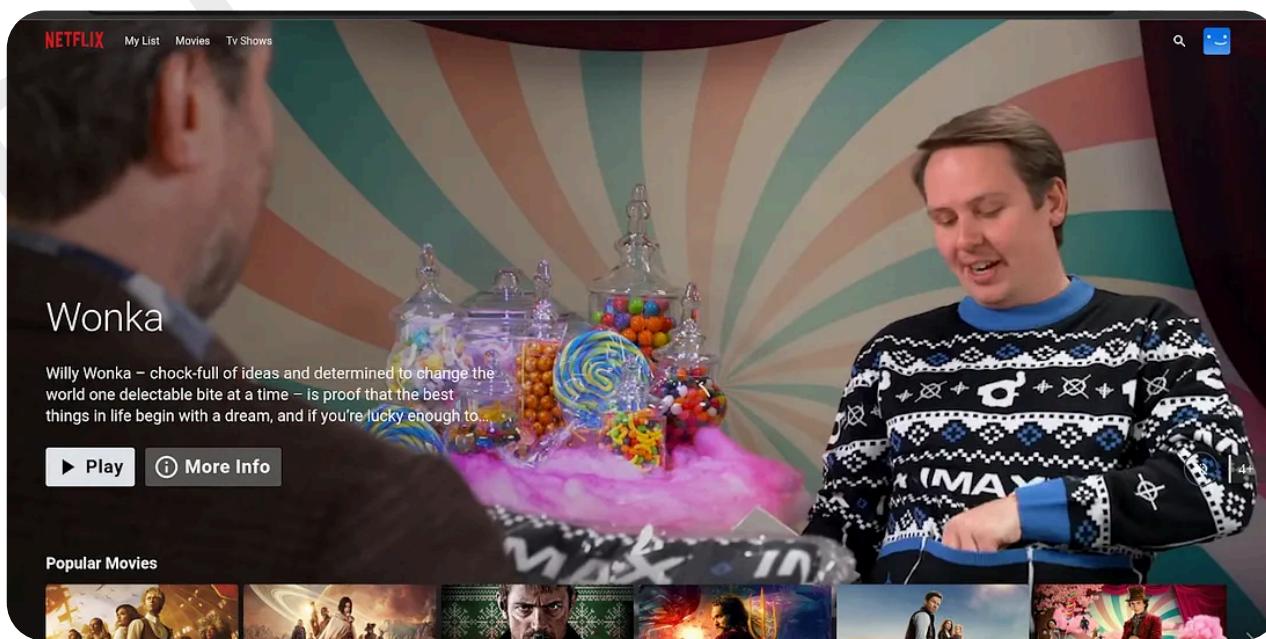
[GITHUB REPO](#)

DEVSECOPS

EKS

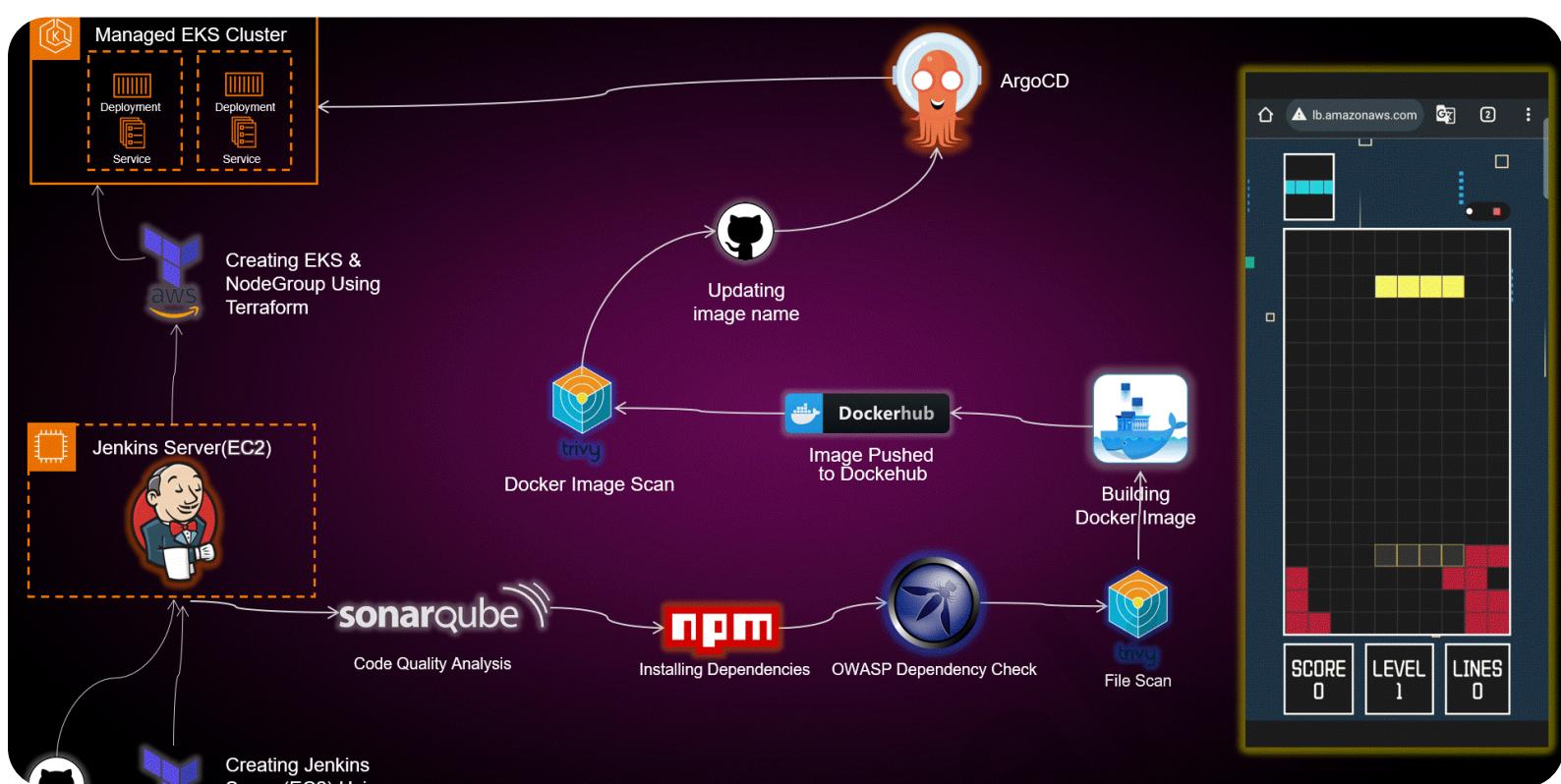
GITOPS

JENKINS



End-to-End DevSecOps Project (Jenkins, K8S, Trivy, Docker, AWS)- Deploy Game

A complete DevSecOps pipeline with container scanning, SAST checks, Jenkins pipeline automation, and secured deployments to Kubernetes on AWS using Trivy and IAM best practices.



↗ [BLOG LINK](#)

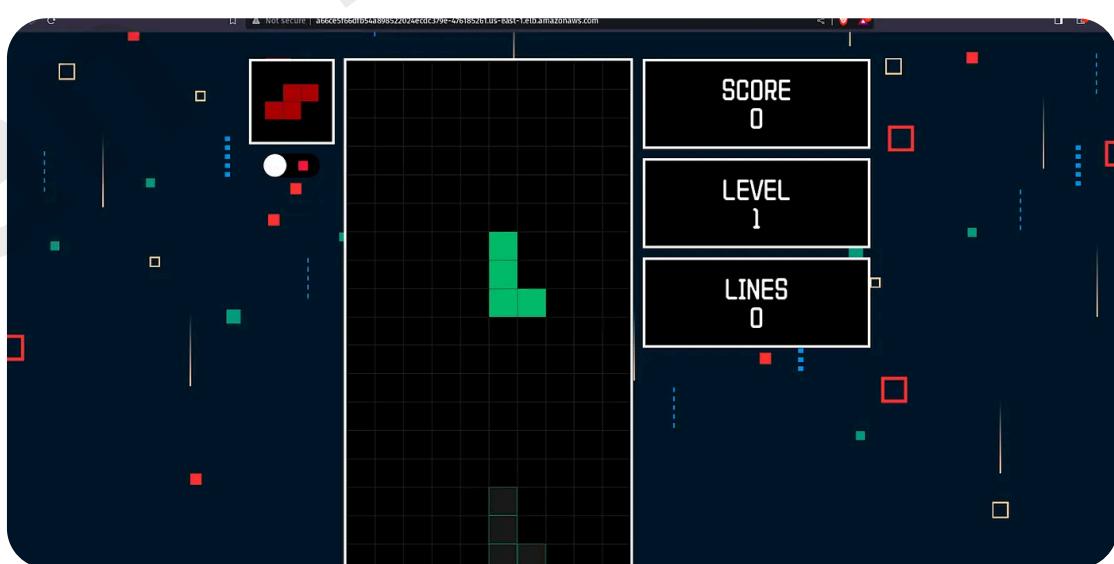
[GITHUB REPO](#)

DEVSECOPS

DOCKER

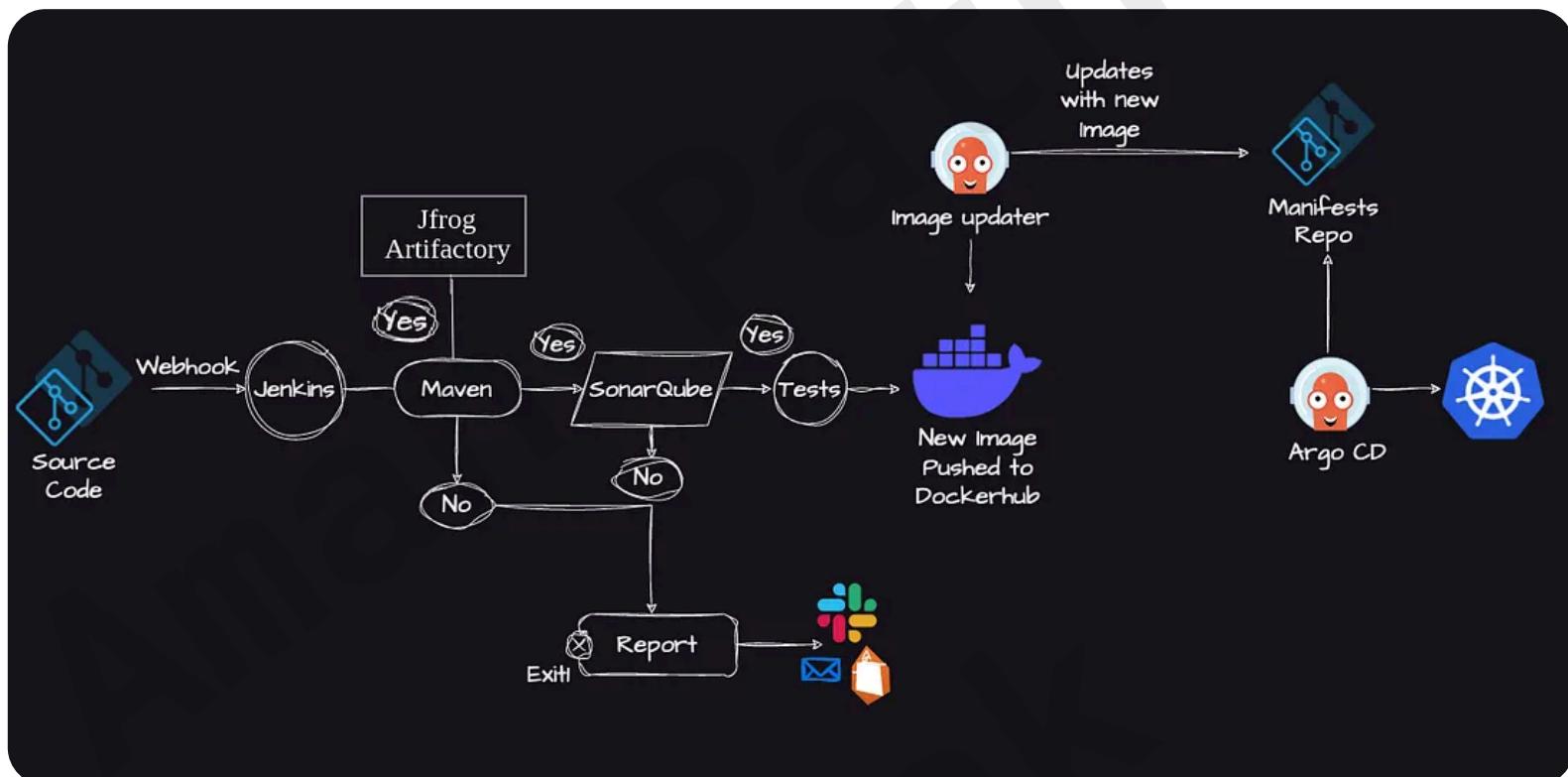
SECURITY

CI/CD



Java CI/CD to K8S with Maven, Jenkins, Sonar, ArgoCD & JFrog

Implement an enterprise-grade CI/CD pipeline for Java apps using Maven, Jenkins, SonarQube, ArgoCD and JFrog Artifactory. Quality checks, binary storage, and auto-deploys to Kubernetes.



↗ [BLOG LINK](#)

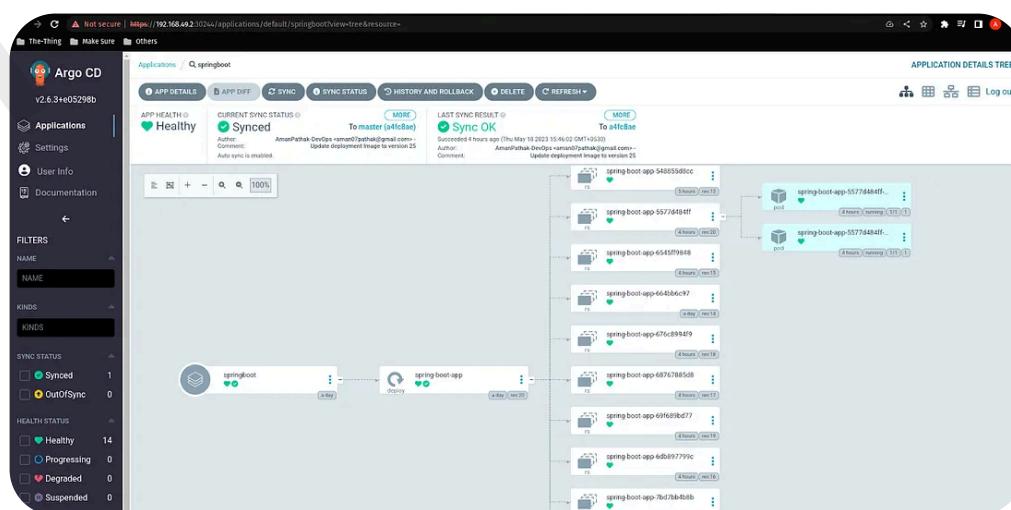
 [GITHUB REPO](#)

[JAVA](#)

[CI/CD](#)

[SONARQUBE](#)

[JFROG](#)



How to Containerise and Deploy a Three-Tier App on AWS EKS

Learn to containerize and deploy a MERN stack three-tier application on EKS with best practices like horizontal scaling, Liveness probes, Ingress, and centralized logging.



↗ [BLOG LINK](#)

[GITHUB REPO](#)

KUBERNETES

EKS

DOCKER

MERN

Production-Ready EKS Clusters with Terraform and GitHub Actions

Automate the provisioning of scalable, secure EKS clusters with Terraform and GitHub Actions CI/CD pipeline. This setup ensures repeatable, consistent, and production-ready infrastructure delivery.



↗ [BLOG LINK](#)

 [GITHUB REPO](#)

EKS

TERRAFORM

CI/CD

GITHUB ACTIONS

A screenshot of the GitHub Actions UI for a "Terraform-Action" job. The "Summary" section shows a list of steps: "CheckOut-Repo", "Setting-Up-Terraform", "Terraform-Initializing", "Terraform-Formatting-Validating", and "Terraform-Action". The "Terraform-Action" step is currently running. The "Terraform-Plan" tab is selected, displaying the Terraform plan code:

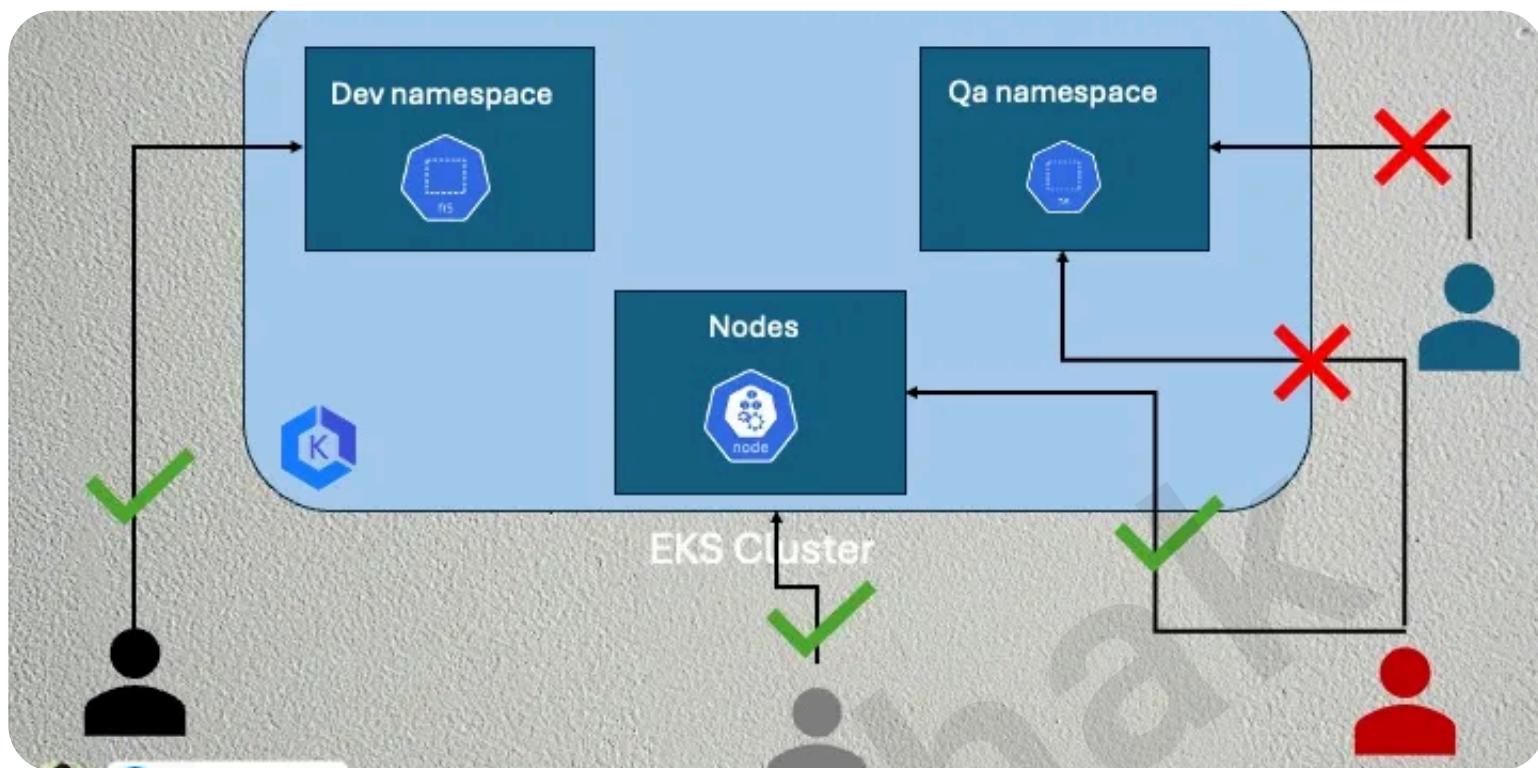
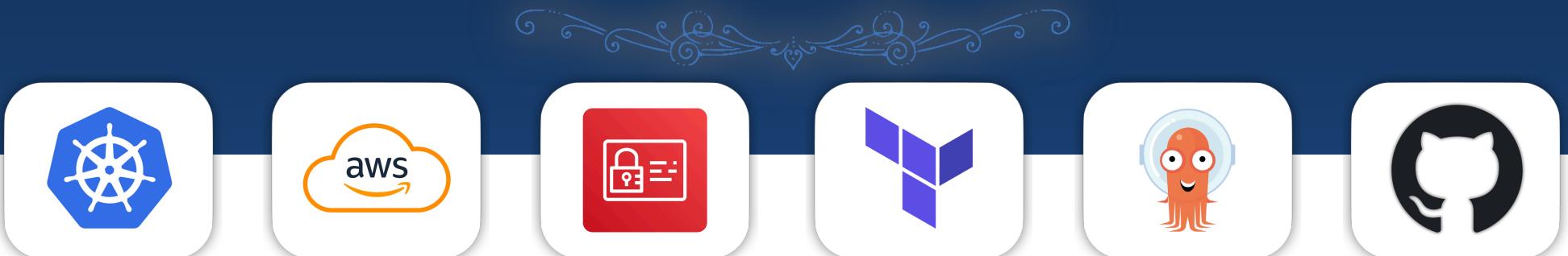
```
resource "aws_vpc" "dev_medium_vpc" {
    cidr_block = "10.0.0.0/16"
    enable_dns_hostnames = true
    enable_dns_support = true
    enable_network_address_usage_metrics = true
    id = "dev-medium-vpc"
    instance_tenancy = "default"
    ipv6_association_id = "dev-medium-association"
    ipv6_cidr_block = "10.0.1.0/24"
    main_route_table_id = "dev-main-route-table"
    owner_id = "123456789012"
    tags = [
        { "Env" = "dev" },
        { "Name" = "dev-medium-vpc" }
    ]
}

resource "random_integer" "random_suffix" {
    max = 9999
    min = 1000
    result = "dev-medium-suffix"
}
```

The plan indicates 37 resources to add, 0 to change, and 0 to destroy.

Configure RBAC on Production Ready EKS Cluster

Secure access to Kubernetes workloads with AWS IAM roles, service accounts, and Kubernetes RBAC. This setup ensures tight access control over your EKS cluster, following least privilege principles for DevOps and app teams in production-grade environments.



↗ [BLOG LINK](#)

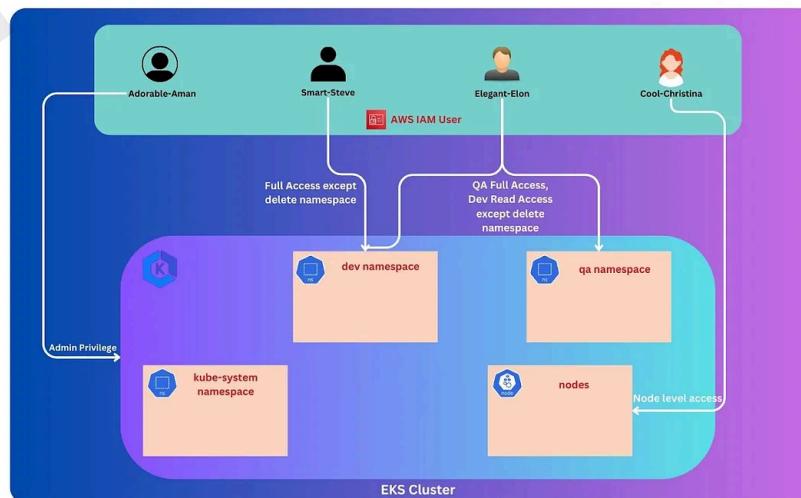
 [GITHUB REPO](#)

EKS

RBAC

SECURITY

IAM



Custom Helm Chart Publishing

Build, package, and publish your own Helm charts for Kubernetes apps. Host them via Helm repo and automate chart versioning with CI tools. A must-do for DevOps engineers creating reusable deployments.



A screenshot of a LinkedIn profile for Aman Pathak. The profile picture shows a person with brown hair. Below the profile picture, there is a blue circular icon containing a Helm logo, followed by a plus sign, another blue hexagonal icon with a ship's wheel, another plus sign, and a GitHub logo. Below this diagram, the text "Create your custom Helm Chart and Publish it" is displayed. At the bottom of the profile, there is a command-line interface (CLI) terminal window showing the command: "helm repo add aman-devops https://amanpathak-devops.github.io/Helm-Charts/".

↗ [BLOG LINK](#)



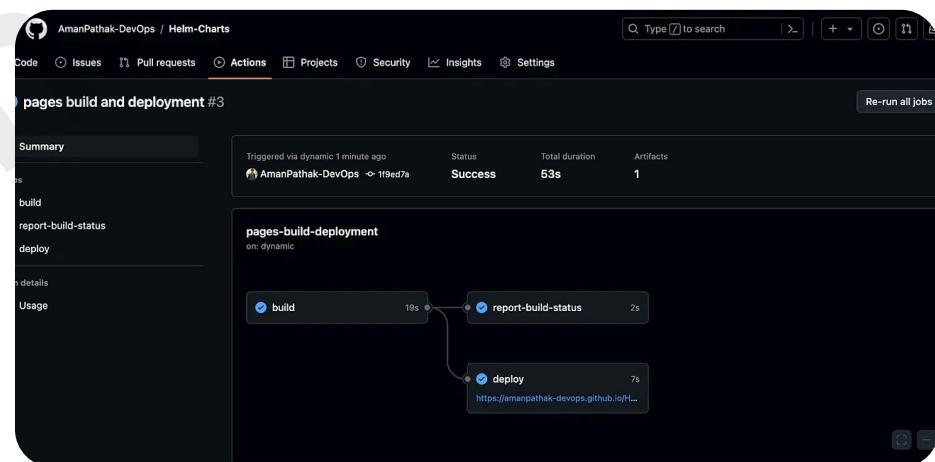
[GITHUB REPO](#)

HELM

CI/CD

KUBERNETES

AUTOMATION



Configure ArgoCD, Prometheus, Grafana & AWS Load Balancer Controller on EKS using Terraform

Set up ArgoCD for GitOps, monitor workloads with Prometheus and Grafana, and use AWS Load Balancer Controller for high availability. A complete observability + deployment stack on EKS.



[BLOG LINK](#)



[GITHUB REPO](#)

[GITOPS](#)

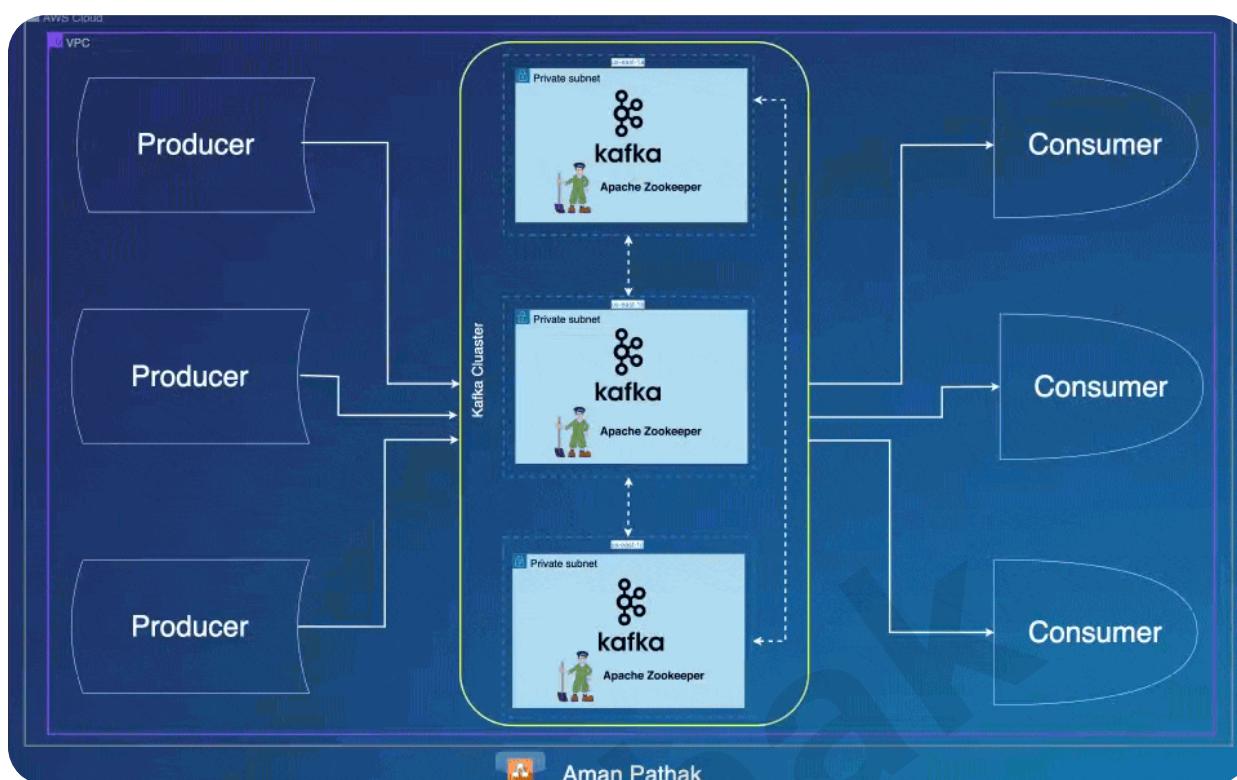
[MONITORING](#)

[EKS](#)

[ARGOCD](#)

Kafka Cluster on AWS EC2

Deploy a scalable, multi-node Kafka cluster using AWS EC2 and Zookeeper. Learn to configure topic partitions, replication, and network isolation. Ideal for data engineers working on production streaming architectures.



↗ [BLOG LINK](#)

 [GITHUB REPO](#)

KAFKA

EC2

AWS

STREAMING

```
fka Producer", "topic": "test-topic", "1  
xceeded 90% on server node-42. Immediate  
176.165:9092 --topic test-topic
```

```
.165:9092 --topic test-topic --from-  
pic", "location": "Data Center - EU", "i  
mmediate action required to prevent syst
```

K8sGPT with Ollama on EKS

Integrate K8sGPT and Ollama for AI-powered diagnostics in Kubernetes. Automatically detect and explain cluster issues in real time on AWS EKS. A futuristic observability and troubleshooting solution for production environments.



↗ [BLOG LINK](#)

 [GITHUB REPO](#)

K8SGPT

EKS

OLLAMA

AWS

```
athak@Devops POC % k8sgpt analyze --filter Service,Pod,Node --backend ollama --explain
provider: ollama

service default/nginx-svc()
cor: Service has no endpoints, expected label name=nginx
cor: service has no endpoints, expected label name=nginx
cor: 
cor: check if you have applied the correct label to your nginx deployment.
cor: `kubectl get svc -l name=nginx` to verify the service exists and is labeled correctly.
cor: if it doesn't exist, apply the correct label using `kubectl label svc <service-name> name=nginx`.
pod default/broken-app-788fffb4c99-g5zhb(Deployment/broken-app)
cor: the last termination reason is Error container=broken-app-container pod=broken-app-788fffb4c99-g5zhb
is the simplified error message:

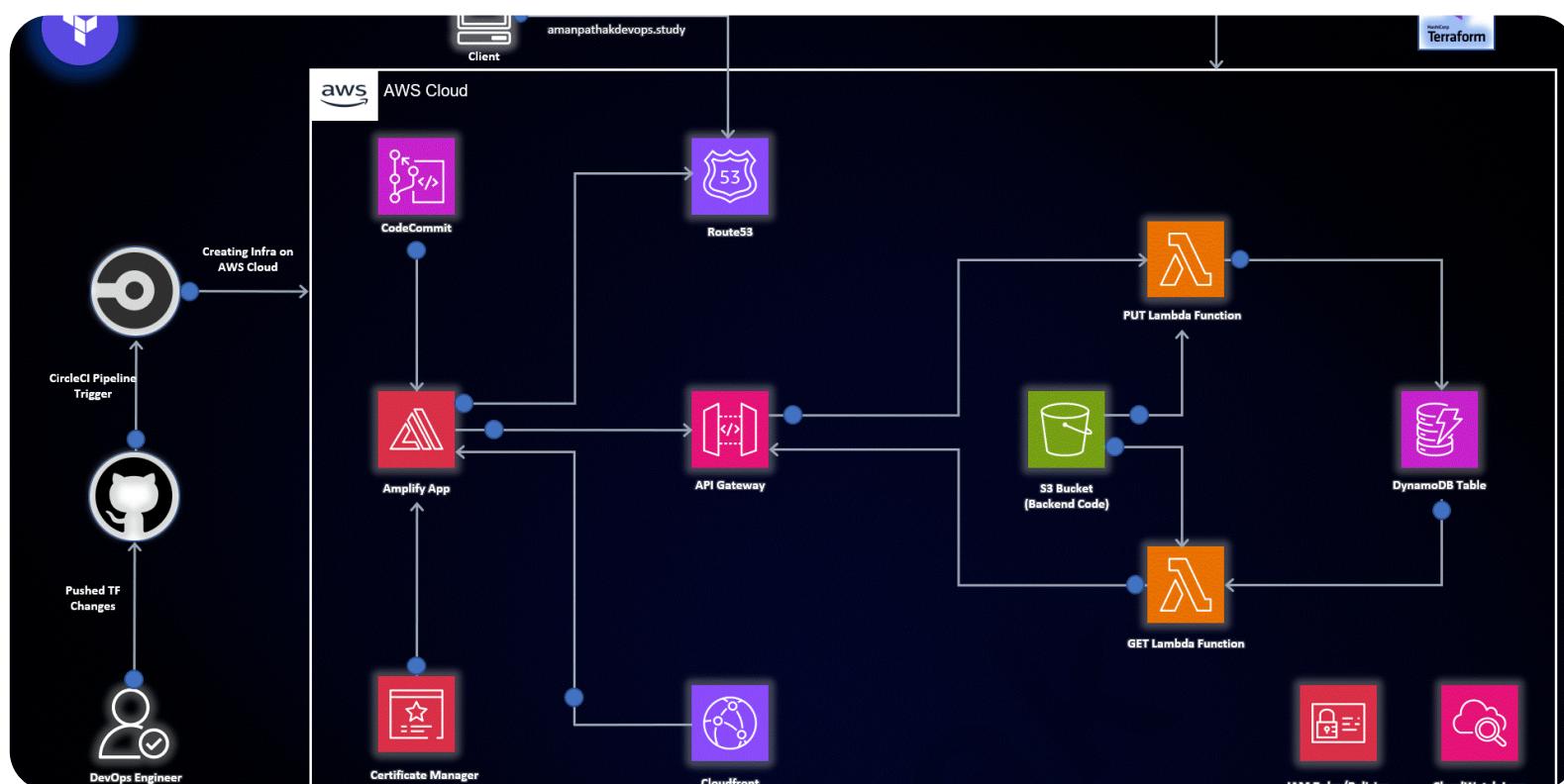
cor: A container in your pod, 'broken-app-container', has crashed due to an unexpected error.

action:
check the container logs for more information.
identify the cause of the crash (e.g., code issue or resource constraint).
fix the issue and restart the container.
verify that the pod is running successfully.
pod default/pod-resources-demo()
cor: 0/2 nodes are available: 2 Insufficient cpu, 2 Insufficient memory. preemption: 0/2 nodes are available: 2 No preemption victims found for incoming pod.
cor: 0/2 nodes are available: 2 Insufficient cpu, 2 Insufficient memory.
cor: 
cor: check resource requests and limits in pod spec.
cor: increase resources (cpu, memory) for pods.
cor: verify node availability and capacity.
pod dev/runtime-failure-demo-5646c9569c-kbmv8(Deployment/runtime-failure-demo)
cor: couldn't find key APP_MODD in ConfigMap dev/app-config
cor: Couldn't find key APP_MODD in ConfigMap dev/app-config.

action:
check if the ConfigMap is correctly created and deployed.
make sure that the key 'APP_MODD' exists in the ConfigMap.
if not, create a new ConfigMap with the required key-value pair.
update the existing ConfigMap to include the missing key.
```

AWS Serverless App (Manual)

Manually deploy a full-stack serverless web app on AWS using services like API Gateway, S3, Lambda, DynamoDB, Amplify, and more. Connect domains with Route53 and add SSL via ACM. Perfect for mastering AWS service integrations end-to-end.



↗ [BLOG LINK](#)

 [GITHUB REPO](#)

SERVERLESS

AWS

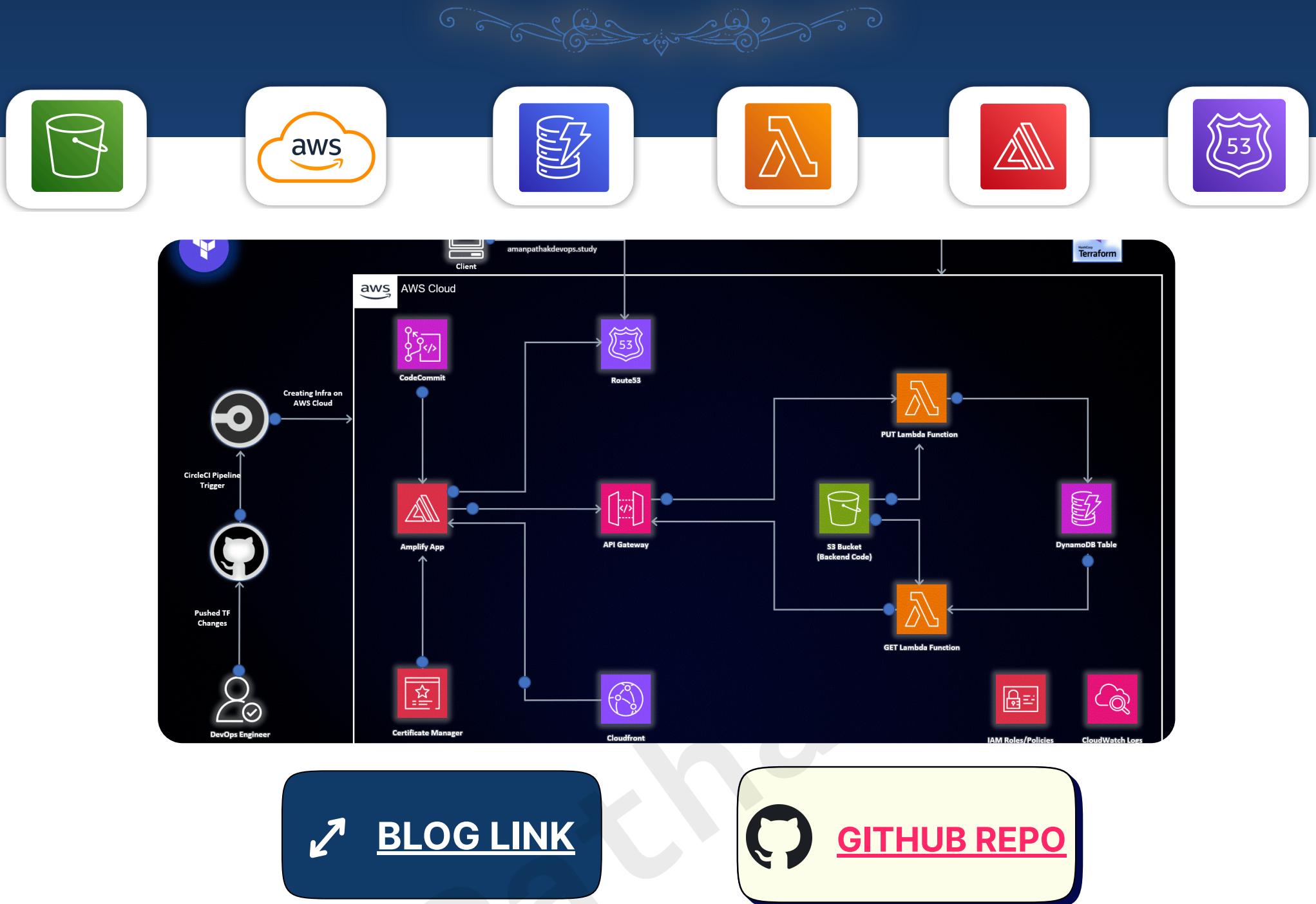
NOTERRAFORM

FULLSTACK



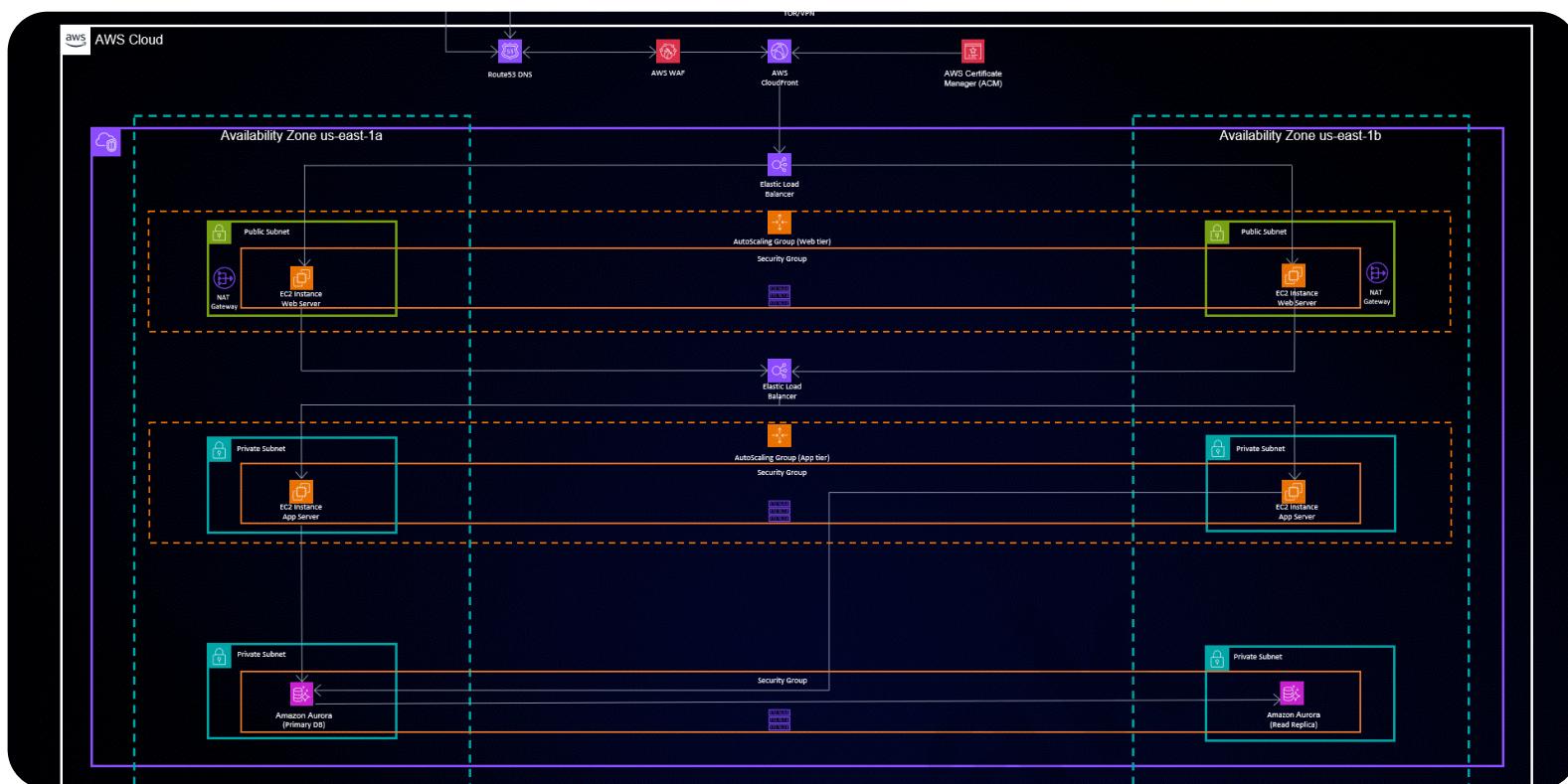
Terraform Serverless App Deployment

Provision and deploy a complete serverless architecture on AWS using Terraform. The stack includes API Gateway, Lambda, DynamoDB, Route53, S3, Amplify, and CircleCI – all orchestrated for automated CI/CD and infrastructure management.



Mastering AWS Three-Tier Infra with Terraform

Build a production-grade three-tier architecture with Auto Scaling EC2 instances, Aurora RDS, Application Load Balancer, WAF, and ACM. Terraform automates it all, bringing infra-as-code to the enterprise level.



↗ [BLOG LINK](#)

 [GITHUB REPO](#)

[TERRAFORM](#)

[AWS](#)

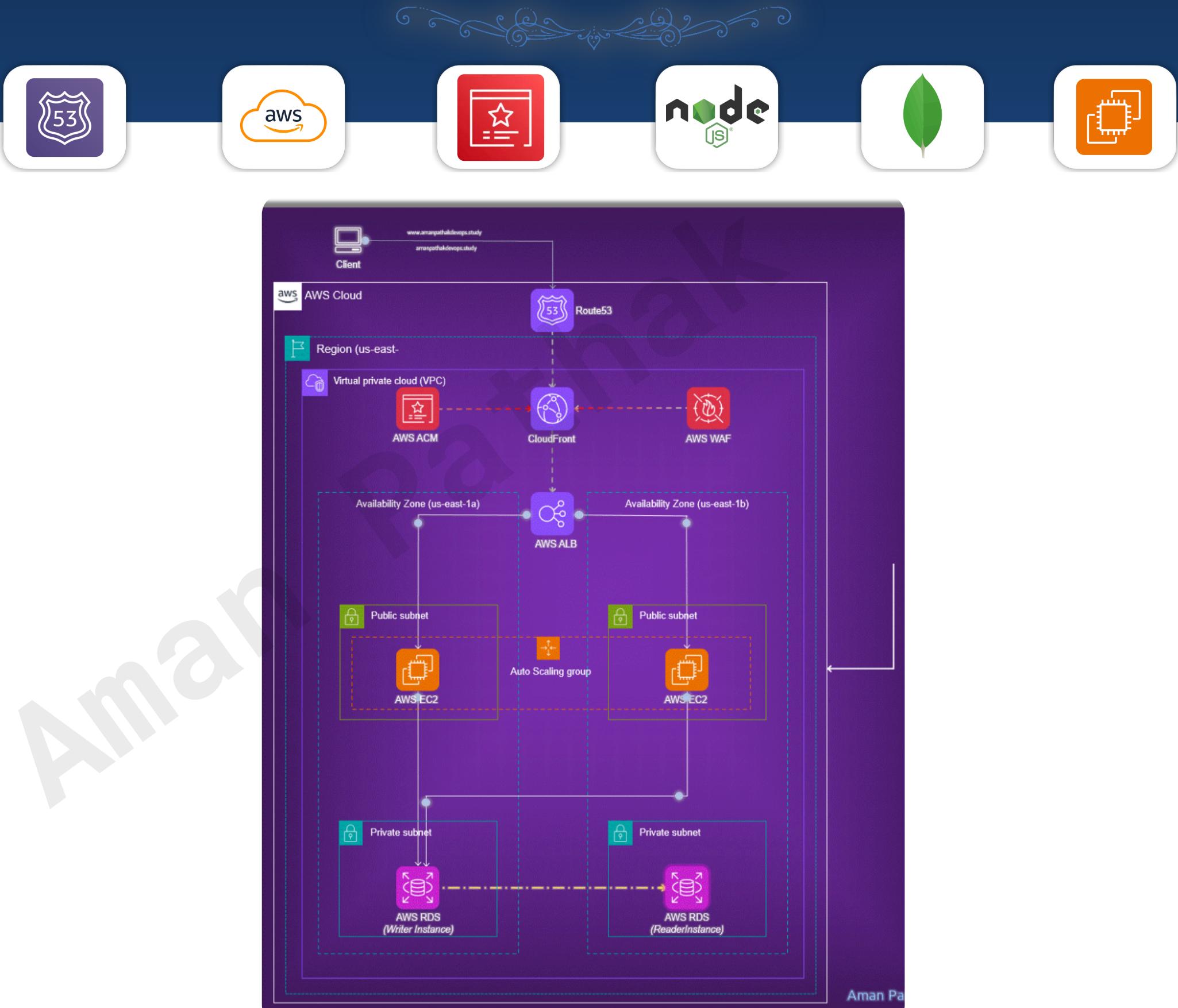
[RDS](#)

[THREE-TIER](#)



Two-Tier App on AWS with Terraform

Deploy a two-tier architecture on AWS: EC2 for web, RDS for data, and set up ACM, CDN, and Load Balancer. All infra provisioned using Terraform with best security practices.



↗ [BLOG LINK](#)

 [GITHUB REPO](#)

AWS

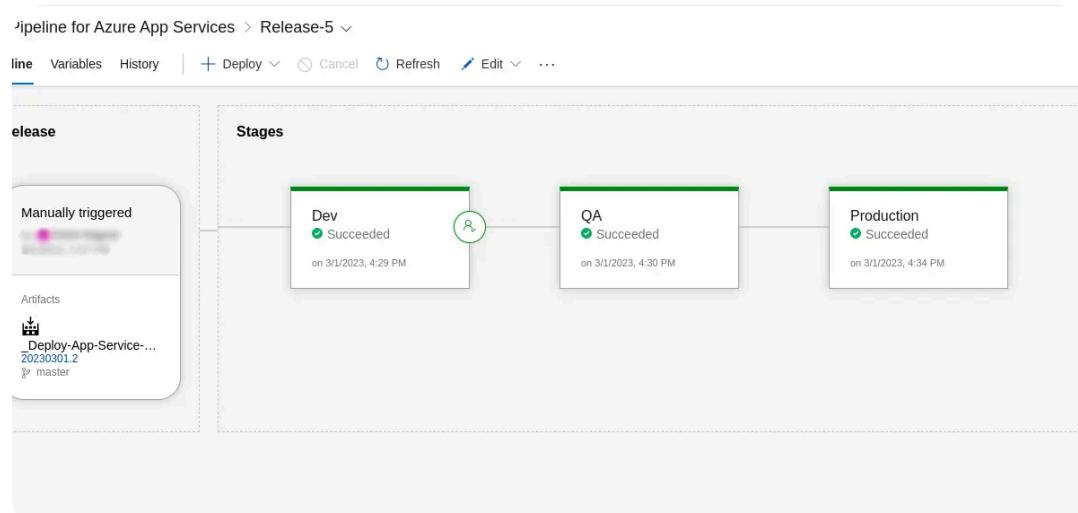
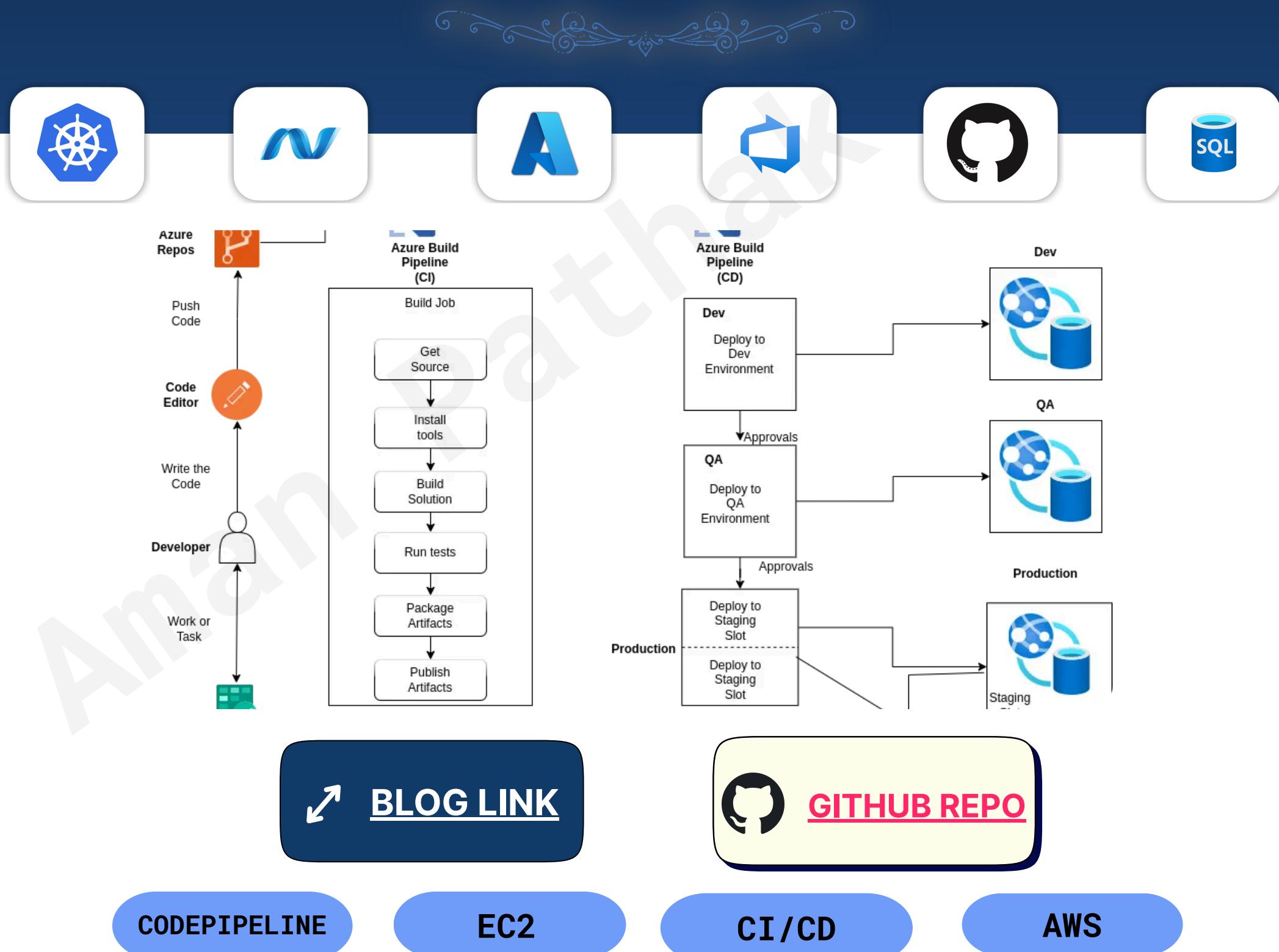
TERRAFORM

TWO-TIER

IAC

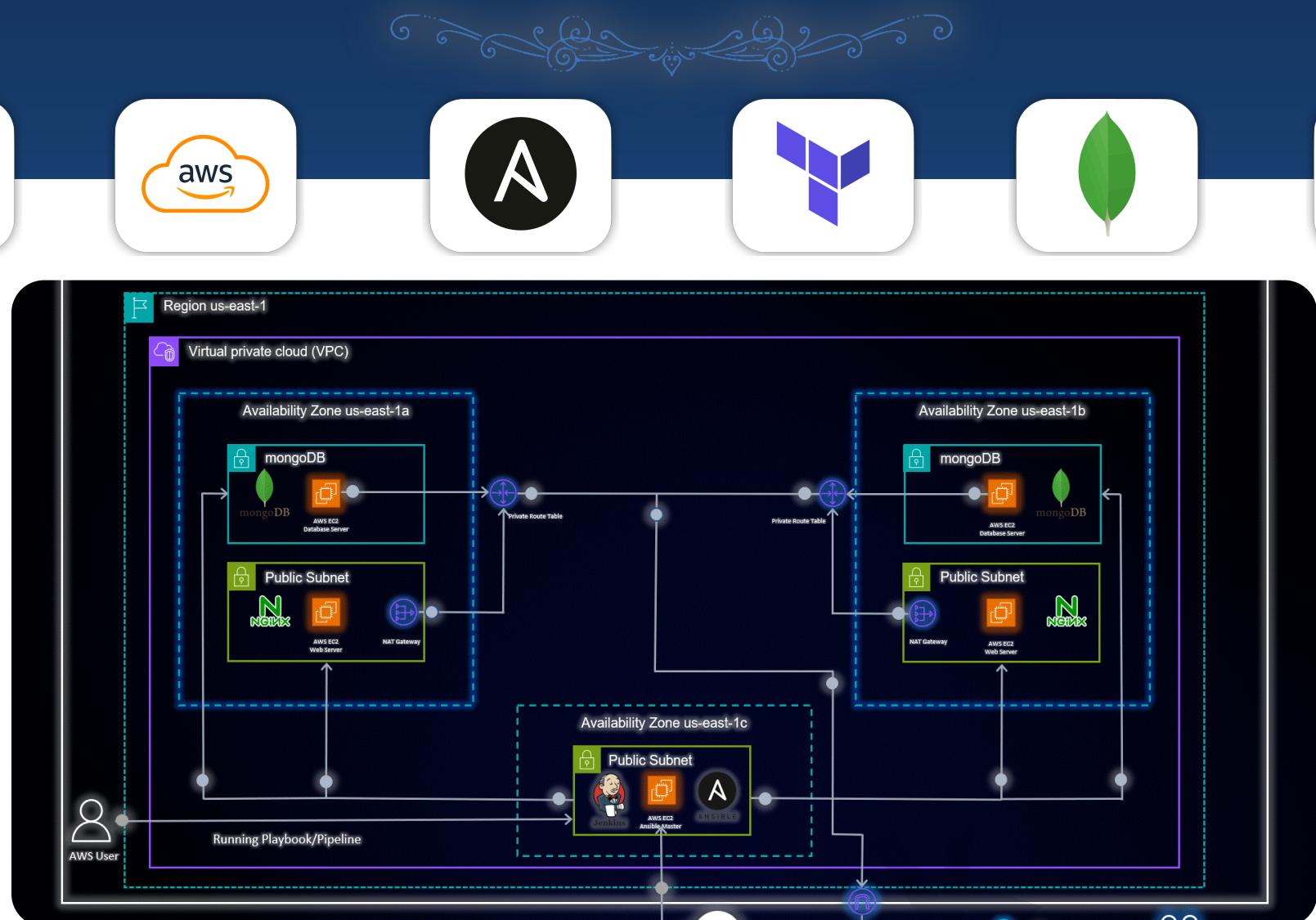
Azure CI/CD with Azure DevOps Pipelines

Set up a CI/CD pipeline using Azure DevOps to build, test, and deploy applications on Azure App Service. Leverage Azure Artifacts and Boards to manage end-to-end development and delivery.



Ansible Infra Automation Chapter 1

Use Ansible Playbooks to configure Nginx and MongoDB across multiple AWS EC2 instances. Automate setup, security, and service validation. Ideal for those stepping into infrastructure-as-code.



↗ [BLOG LINK](#)

[GITHUB REPO](#)

ANSIBLE

AUTOMATION

NGINX

MONGODB

```
11-12-19@: $ mongosh "mongodb://localhost:27017/aman_db"
sh Log ID: 64c7fbfb3ae8b58b95656d6f
To: mongodbs://localhost:27017/aman_db?directConnection=true&serverSelectionTimeoutMS=2000&appName=mongosh+1.10.3
DB: 6.0.8
sh: 1.10.3
info see: https://docs.mongodb.com/mongodb-shell/

Warning: generated these startup warnings when booting
2021-10-13T13:57:50.316+00:00: Using the XFS filesystem is strongly recommended with the WiredTiger storage engine. See http://dochub.mongodb.org/core/prodnotes-filesystem
2021-10-13T13:57:51.383+00:00: Access control is not enabled for the database. Read and write access to data and configuration is unrestricted
2021-10-13T13:57:51.383+00:00: vm.max_map_count is too low

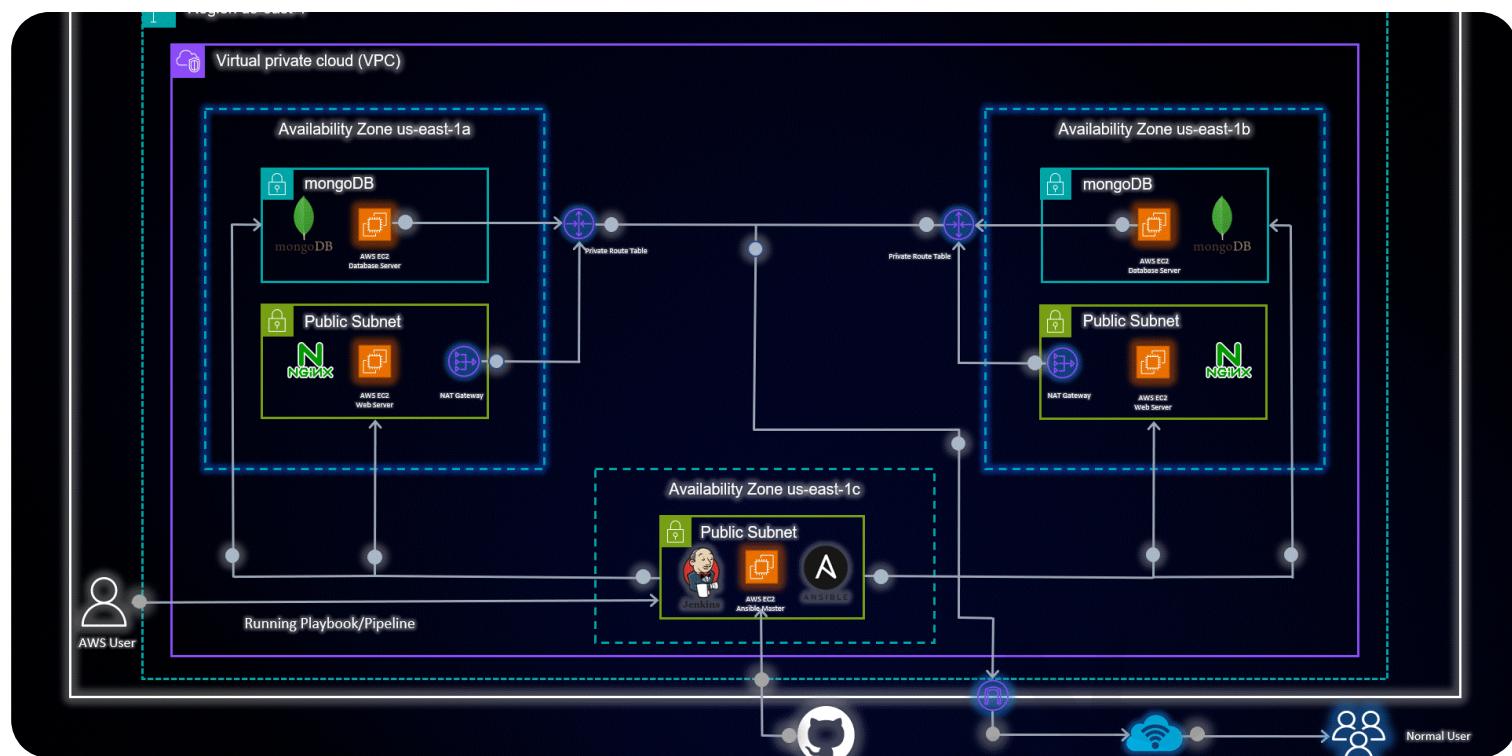
.getUsers()

aman_db.admin
: new UUID("c4f2435b-c66e-488c-b310-diba03c61620"),
admin,
aman_db
:[
  { role: 'readWrite', db: 'admin' },
  { role: 'readWrite', db: 'aman_db' }
]
nisms: [ 'SCRAM-SHA-1', 'SCRAM-SHA-256' ]

aman_db.backup/usr
: new UUID("1467b79e-23be-471e-9930-7aafbb37dded"),
'backup/usr',
aman_db
:[
  { role: 'backup', db: 'admin' },
  { role: 'backup', db: 'aman_db' }
]
nisms: [ 'SCRAM-SHA-1', 'SCRAM-SHA-256' ]
```

Ansible Infra Automation Chapter 2

Extend Chapter 1 to handle advanced configs, backup, user management, and remote log integration. Implement Ansible roles for reusable, scalable playbook architecture.



↗ [BLOG LINK](#)

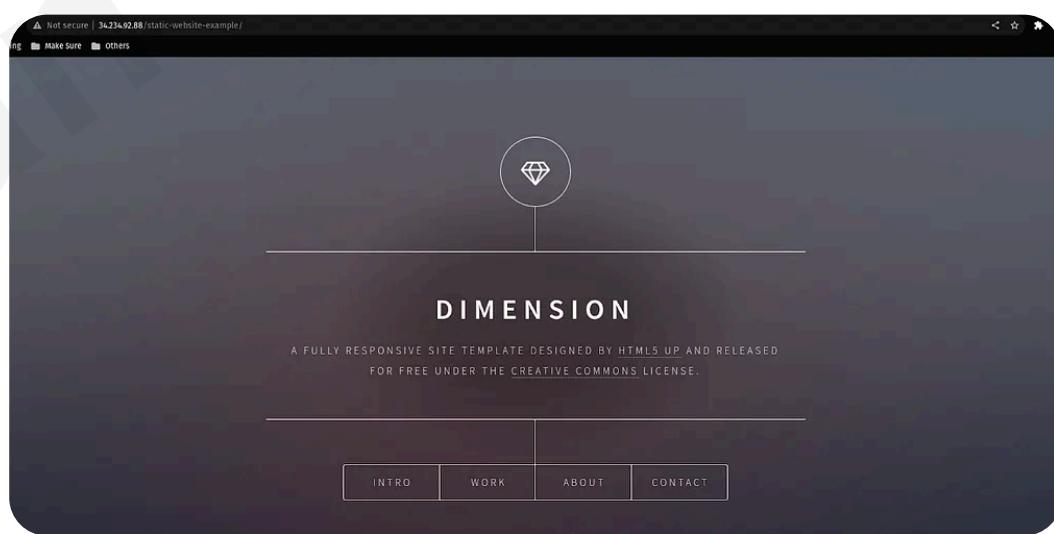
 [GITHUB REPO](#)

ANSIBLE

AUTOMATION

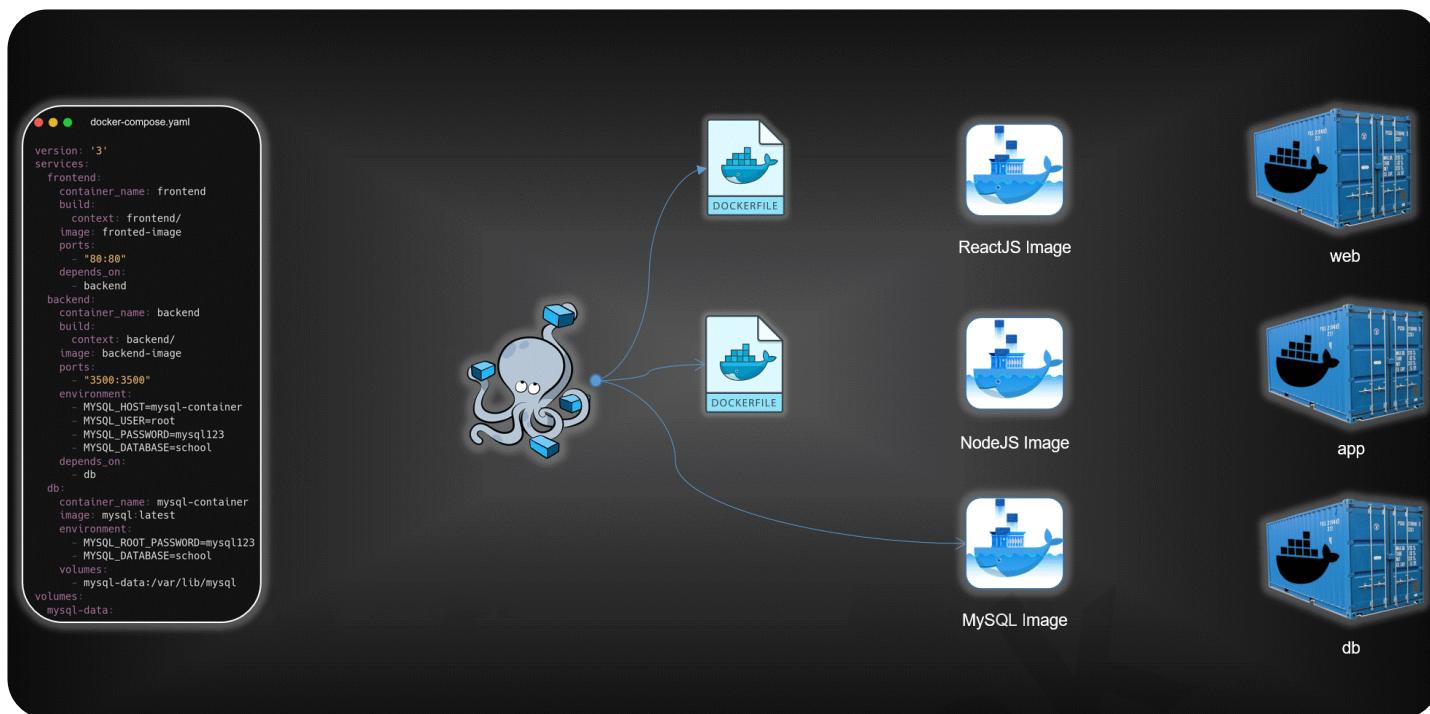
NGINX

MONGODB



Deploy MERN Stack with Docker Compose

Package and deploy a complete MERN stack using Docker and Docker Compose. Learn service linking, volumes, and network configs to run frontend, backend, and DB in harmony.



[BLOG LINK](#)



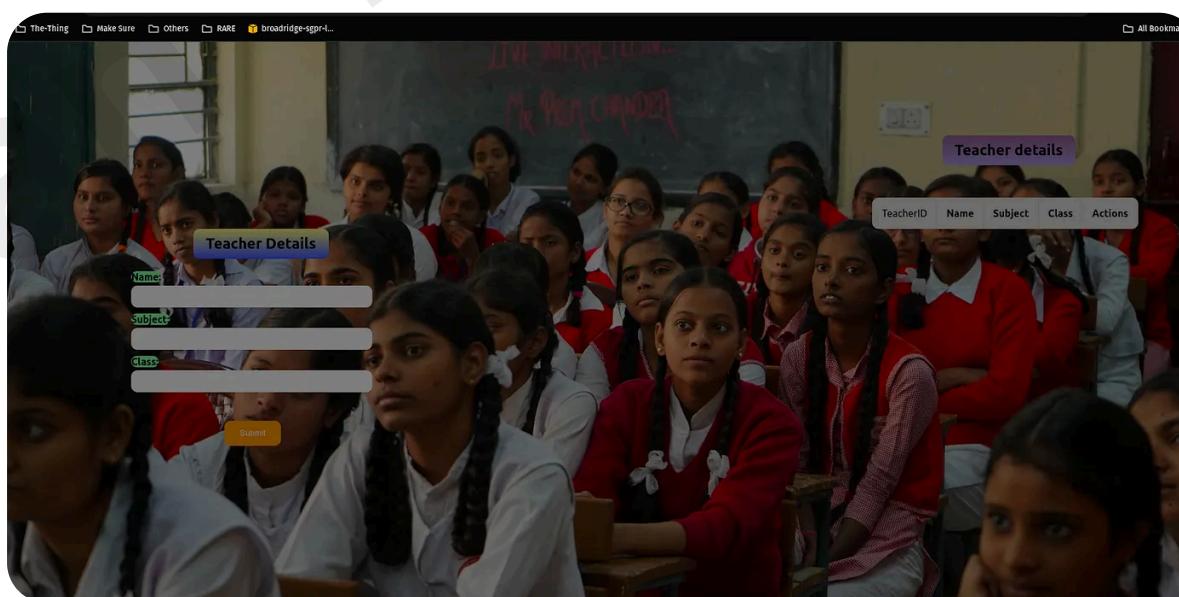
[GITHUB REPO](#)

KUBERNETES

EKS

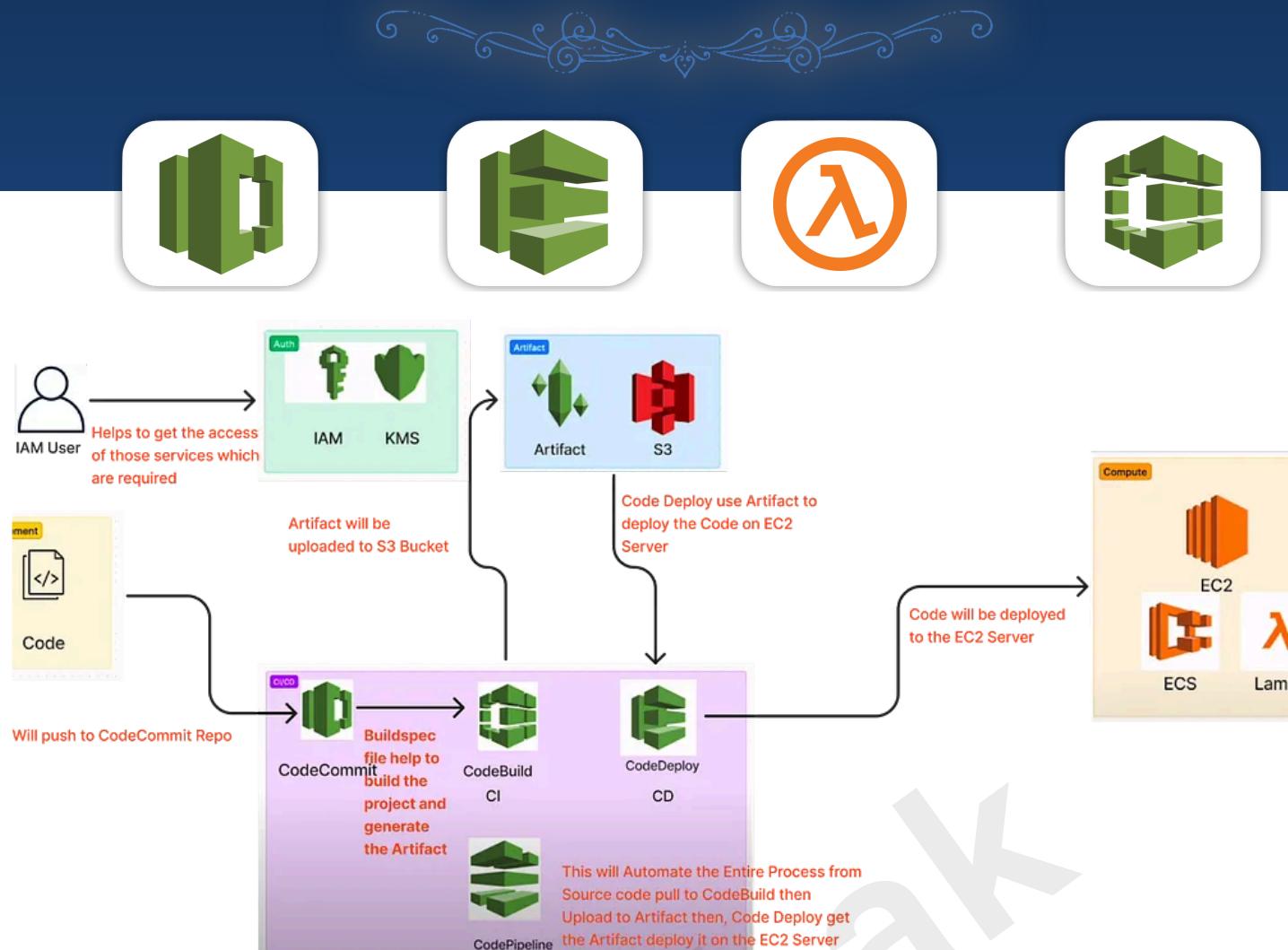
DOCKER

MERN



AWS CodePipeline for EC2 App

Build a robust CI/CD pipeline using AWS-native tools: CodeCommit, CodeBuild, CodeDeploy, and CodePipeline to deploy apps on EC2. Integrate with S3 and Lambda for automation.



[BLOG LINK](#)

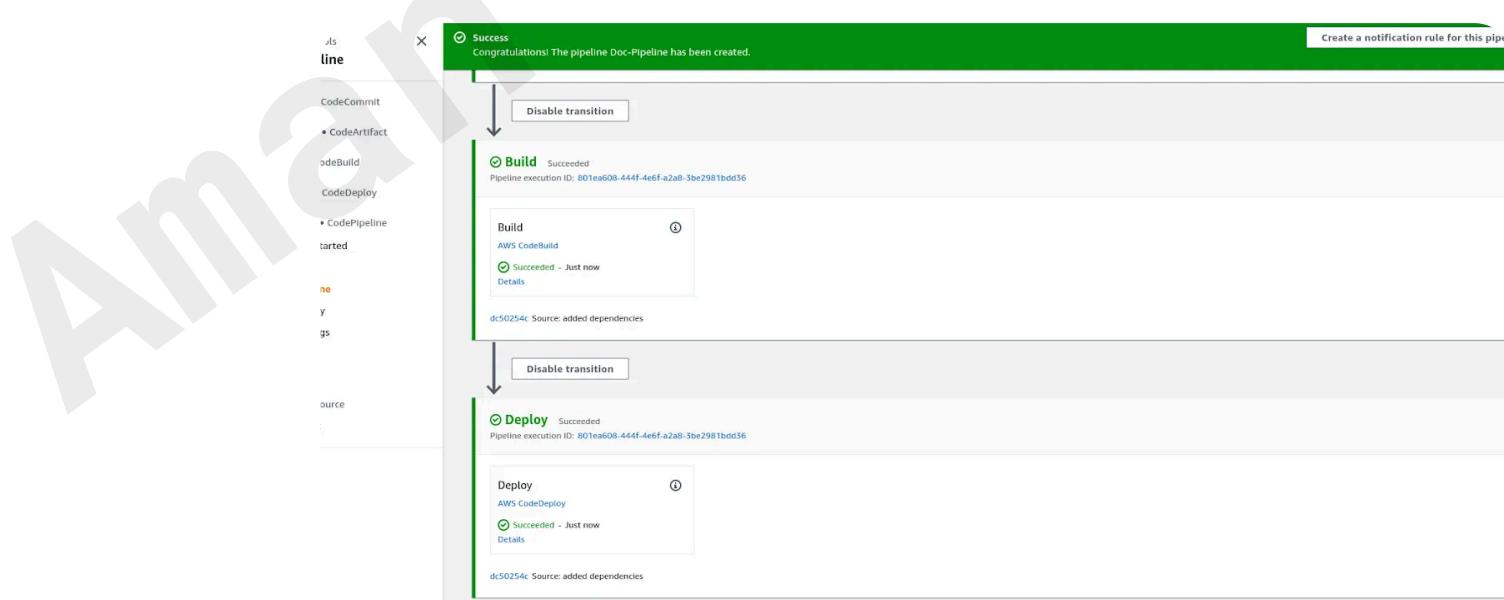
[GITHUB REPO](#)

CODEPIPELINE

CI/CD

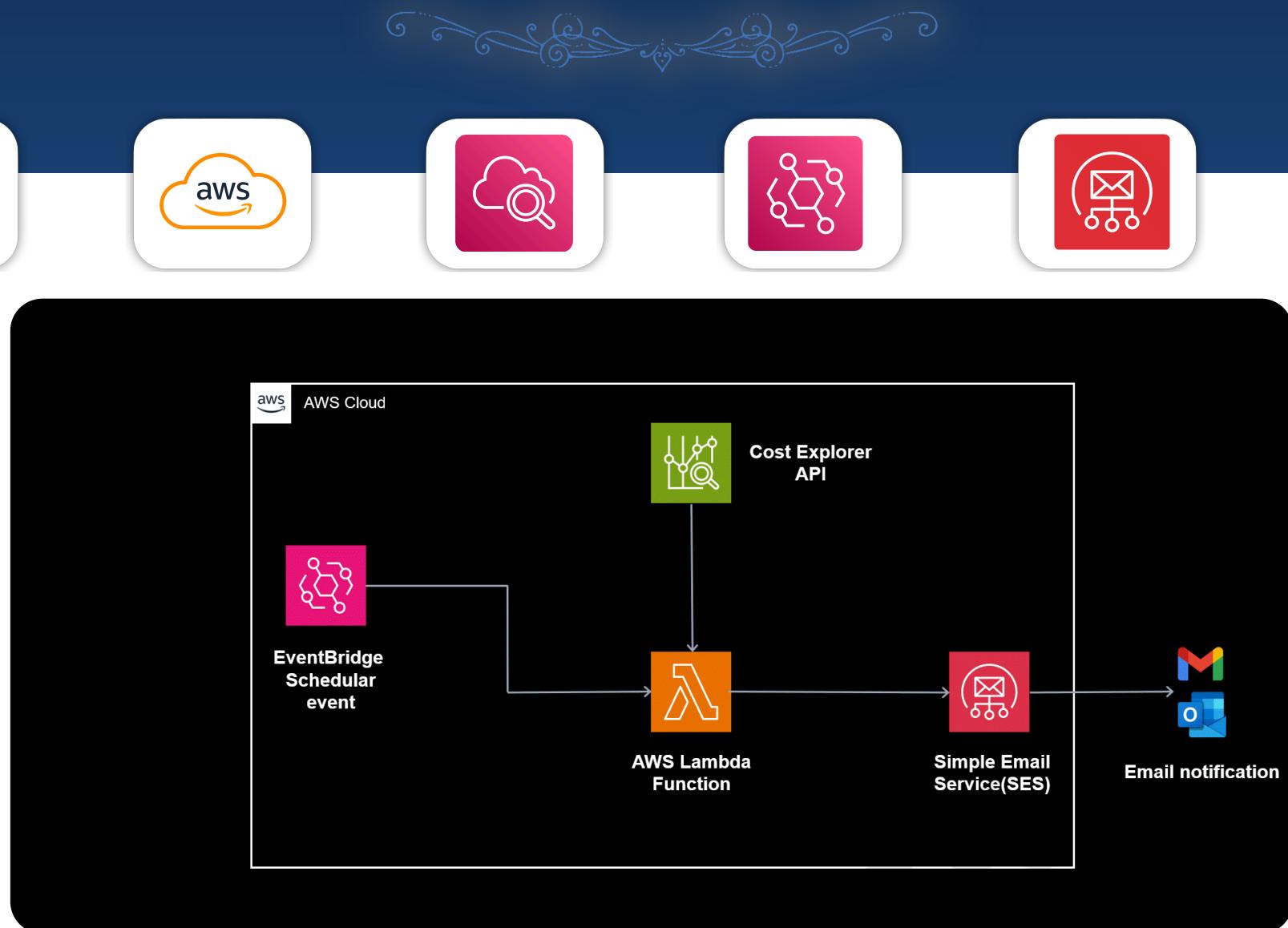
AWS

EC2



Automated AWS Cost Reports

Generate automated AWS billing reports using EventBridge, Cost Explorer API, and SES. Receive daily/monthly emails with detailed usage insights. Cost transparency + alerting in a single serverless pipeline.



↗ [BLOG LINK](#)

 [GITHUB REPO](#)

[AWS BILLING](#)

[AUTOMATION](#)

[SES](#)

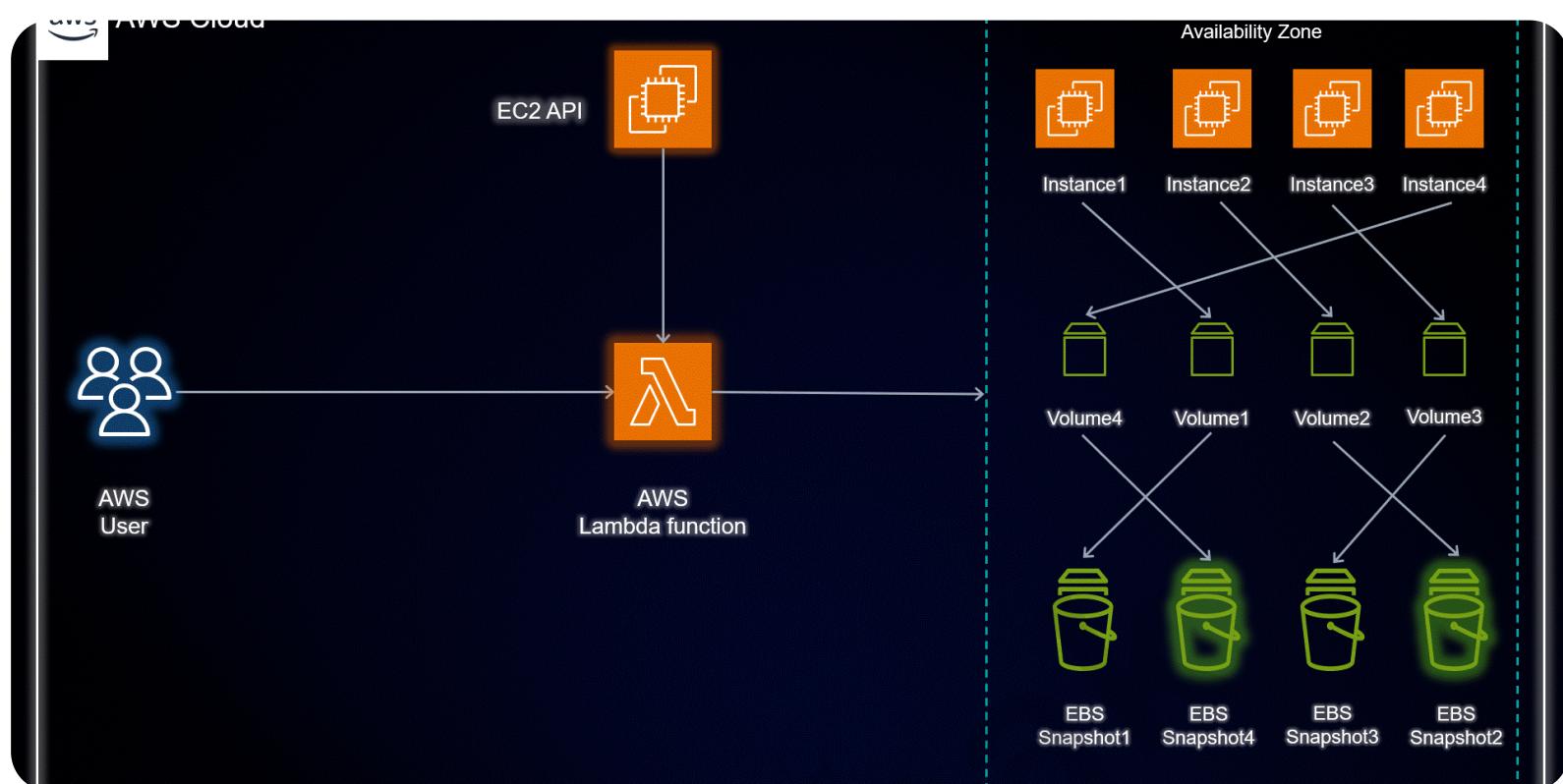
[EVENTBRIDGE](#)

\$0.002

Usage Type	Cost
medStorage-ByteHrs	\$0.002

EBS Snapshot Optimiser Lambda

Save costs on unused snapshots by building a Lambda function to automate retention and cleanup. Tag-aware snapshot rotation and notifications with zero manual effort.



[BLOG LINK](#)

[GITHUB REPO](#)

EBS

LAMBDA

COST SAVINGS

AUTOMATION

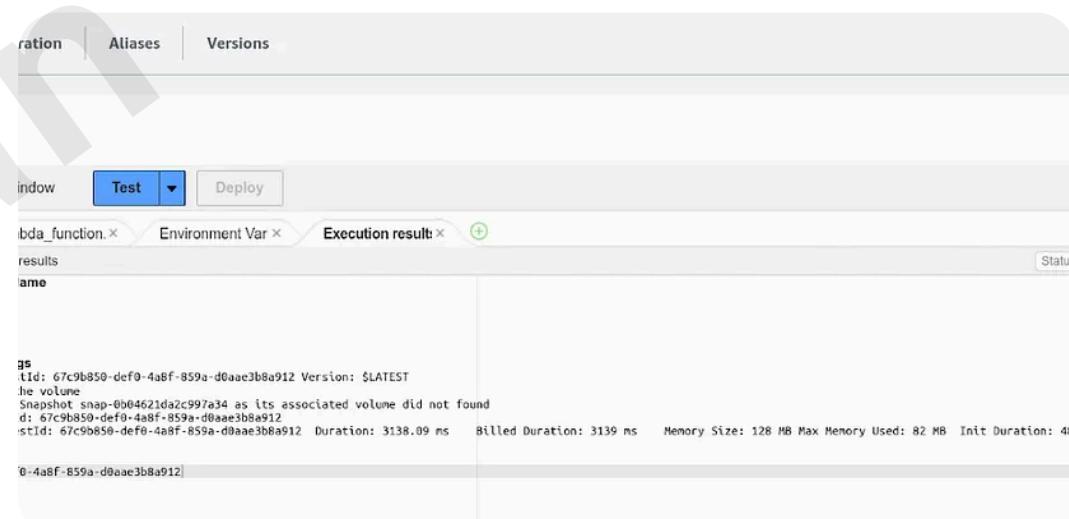
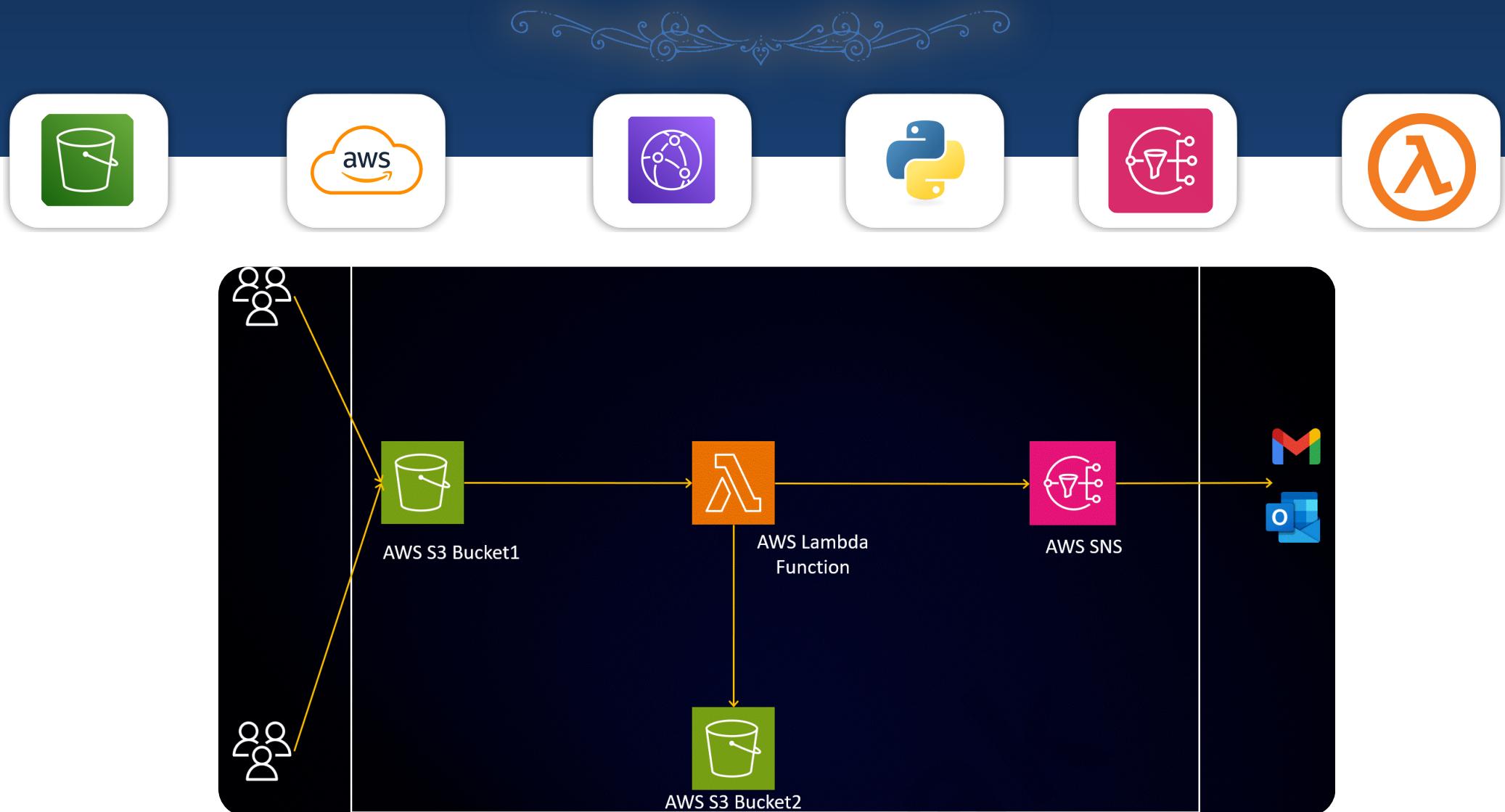


Image Resizing Pipeline (S3 + Lambda)

Create an event-driven image resizing pipeline triggered via S3 uploads. Lambda resizes the image, SNS notifies downstream systems. Fully serverless and scalable for media-heavy apps.



↗ [BLOG LINK](#)



[GITHUB REPO](#)

LAMBDA

S3

SNS

IMAGE PROCESSING

AWS Notifications

to me ▾

The image /tmp/4a9068d0-e0b4-4b32-8f31-7a2dd7999686roshan-mohammed-O_jdx6EeZRA-unsplash.jpg

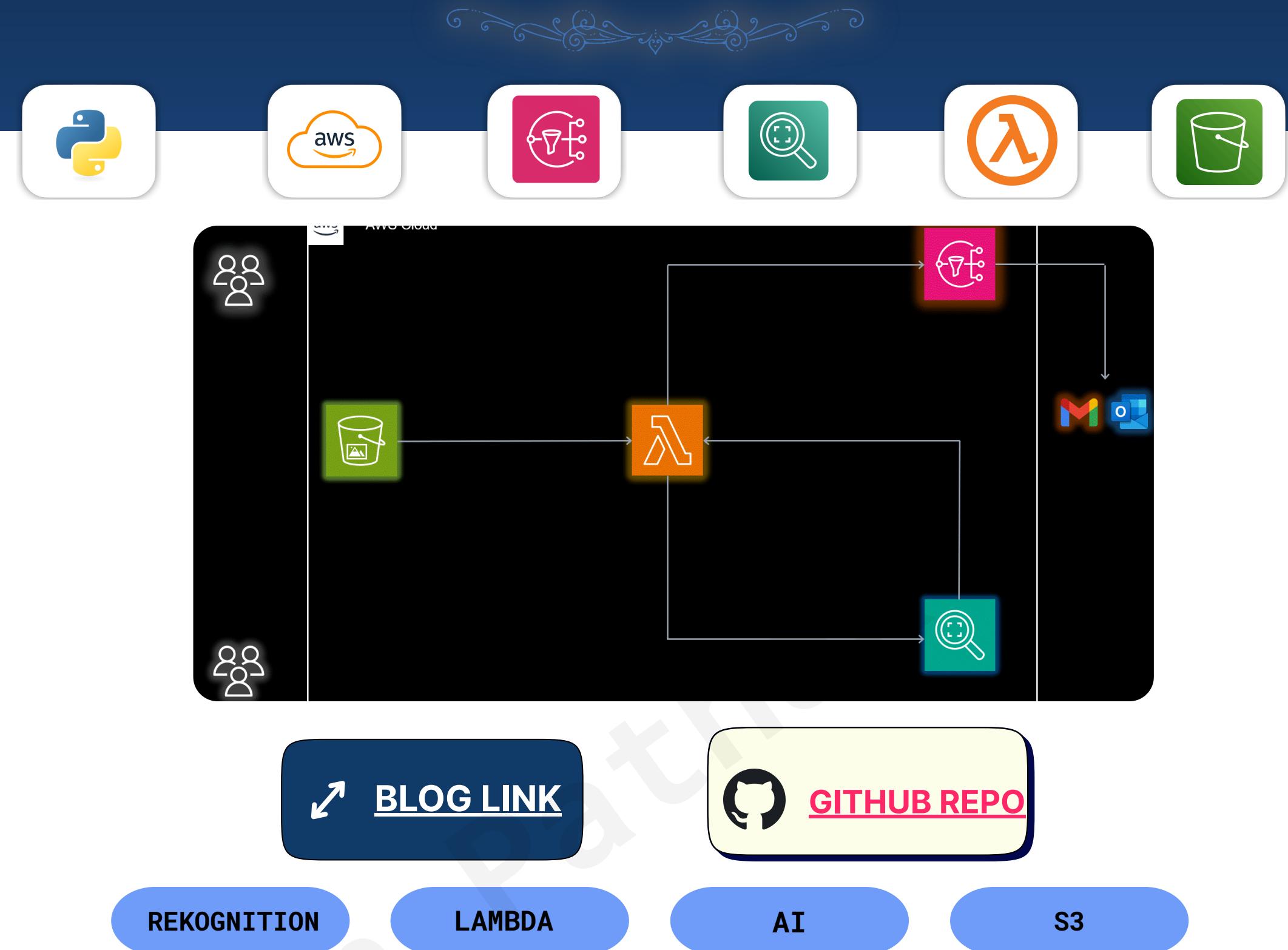
...

If you wish to stop receiving notifications from this topic, please click or visit the link below to unsubscribe:
<https://sns.us-east-1.amazonaws.com/unsubscribe.html?SubscriptionArn=arn:aws:sns:us-east-1:...>

Please do not reply directly to this email. If you have any questions or comments regarding this email, please

Image Recognition with Rekognition

Automatically analyze images uploaded to S3 using AWS Rekognition, notify users via SNS, and orchestrate flows using Lambda. Adds machine vision to your serverless stack



↗ [BLOG LINK](#)

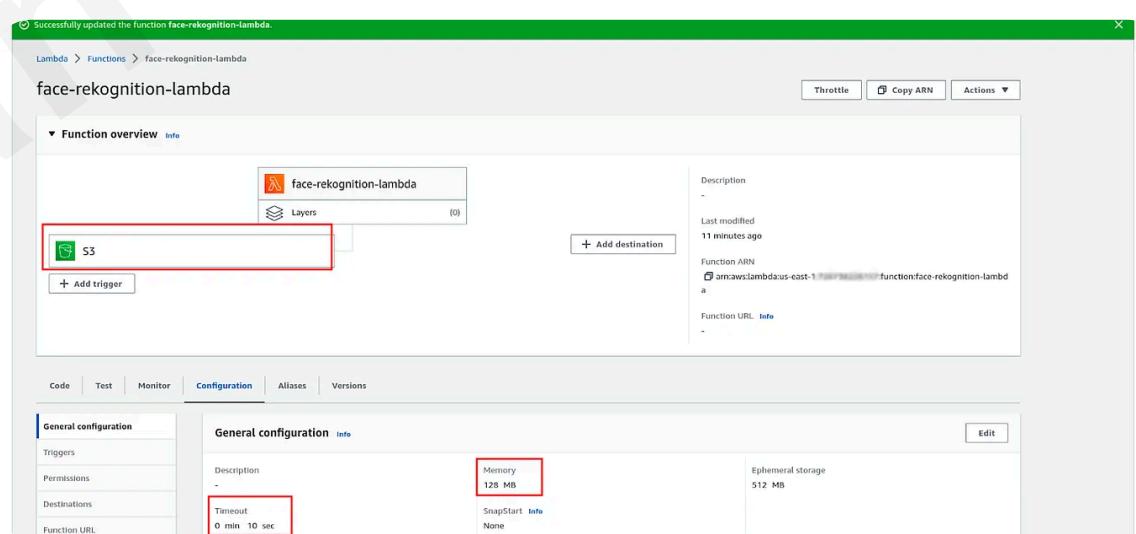
[GITHUB REPO](#)

RECOGNITION

LAMBDA

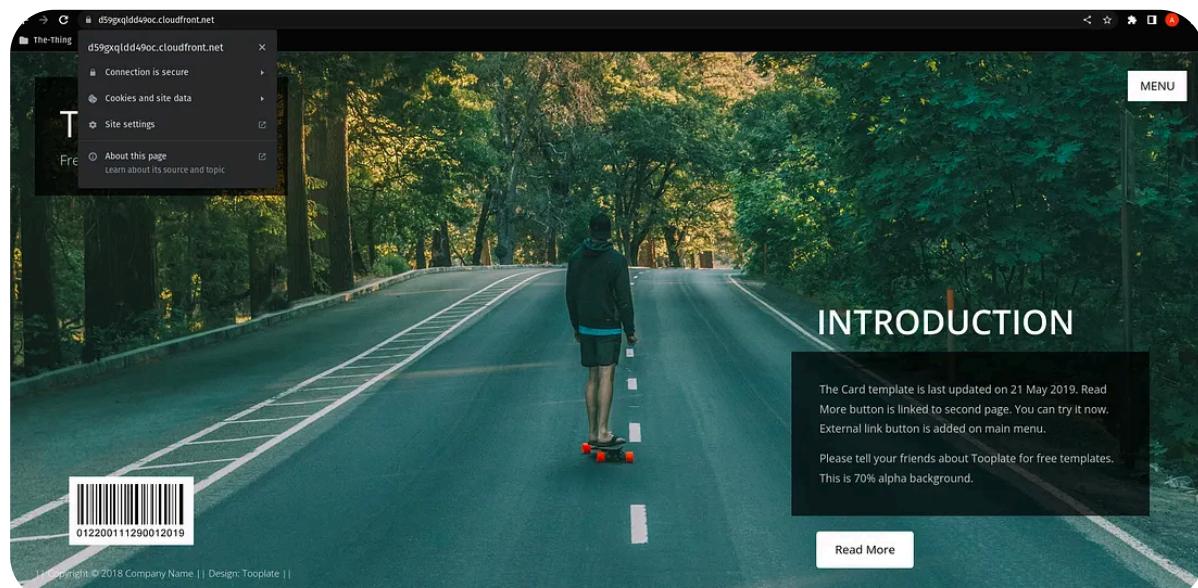
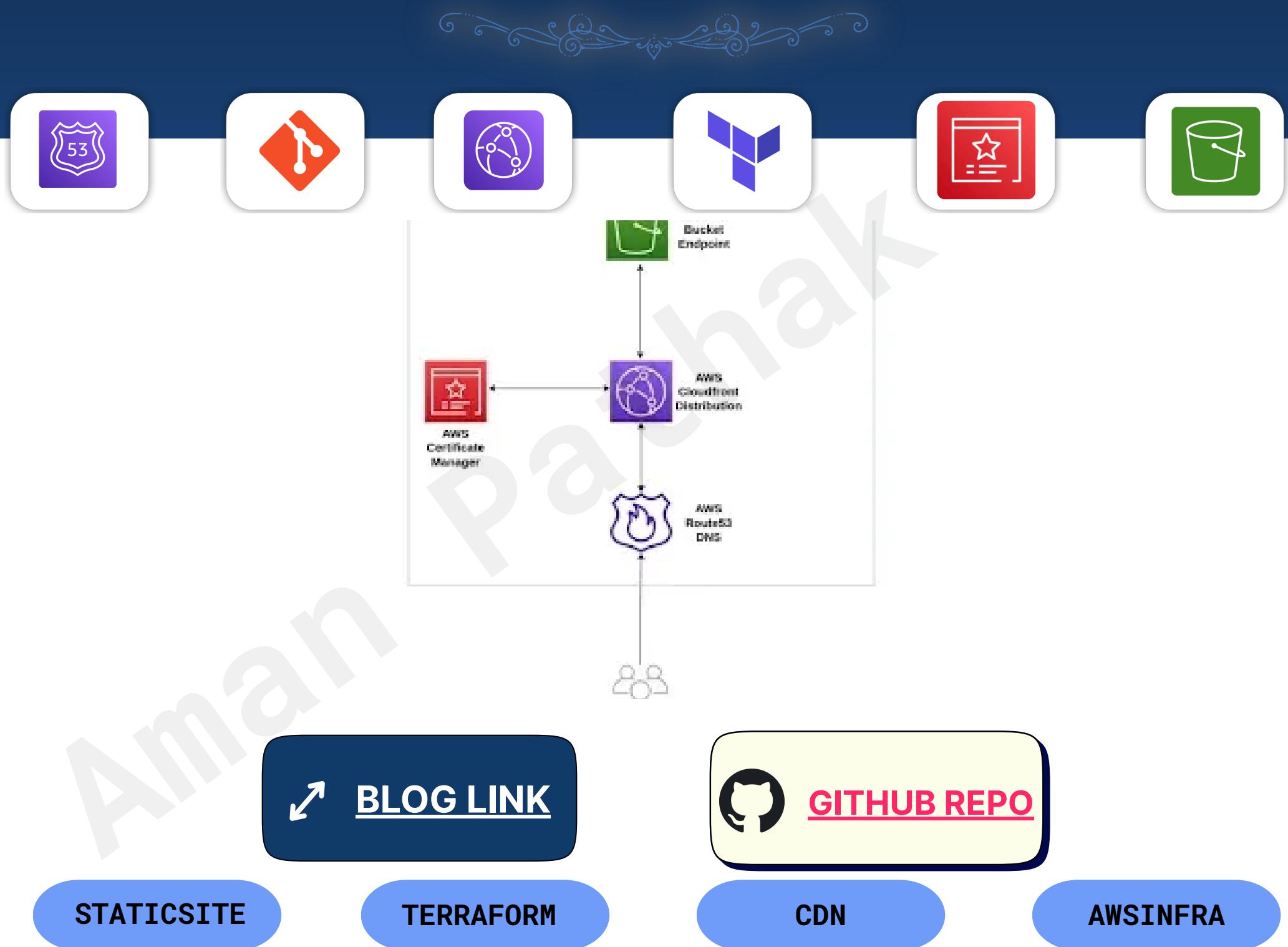
AI

S3



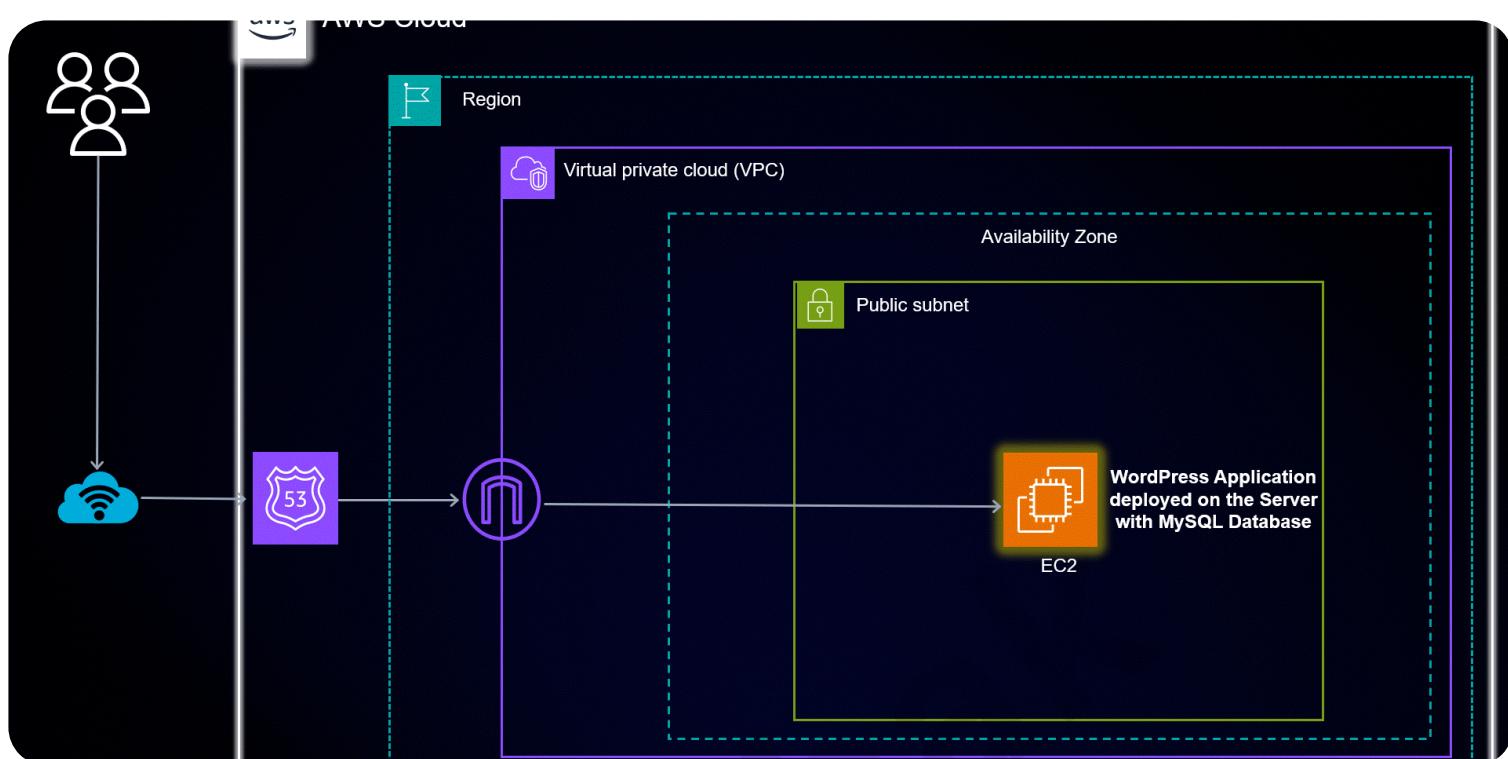
Static Website on AWS with Terraform

Deploy a modern static site using AWS S3, CloudFront, ACM, and Route53 – fully automated via Terraform. Fast, global, and secure content delivery made easy.



Deploy WordPress Application on EC2

Launch a WordPress blog using AWS EC2, configure web server and DB, and map domain via Route53. Perfect starter cloud deployment to learn AWS basics.



↗ [BLOG LINK](#)

WORDPRESS

AWS

EC2

ROUTE53



Mindblown: a blog about philosophy.

Hello world!

Welcome to WordPress. This is your first post.
Edit or delete it, then start writing!

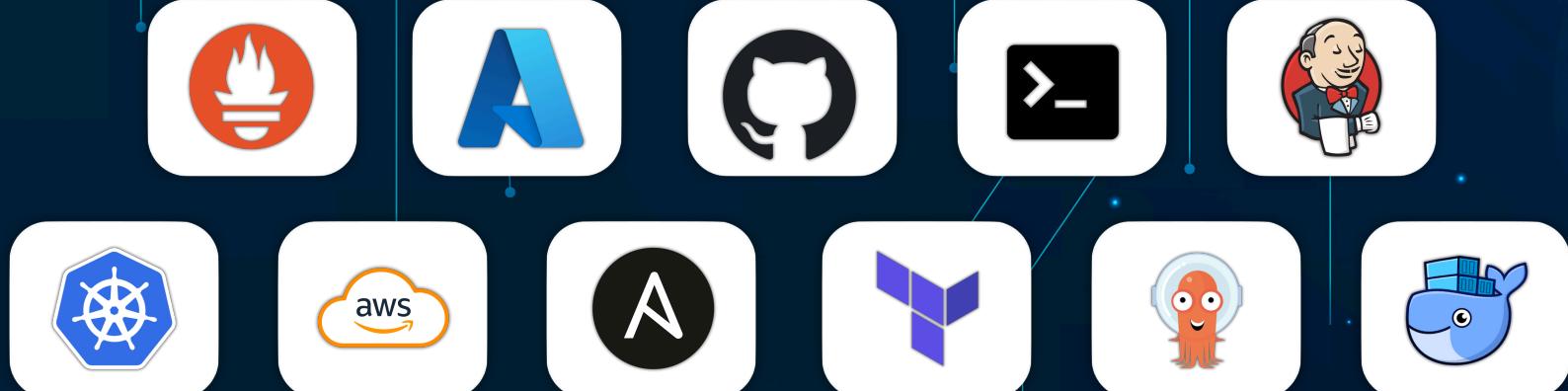
July 14, 2023

Got any book recommendations?

[Get In Touch](#)

Touched the Cloud, Never Held It

*Some architectures aren't meant to last
— but the love of building them
always will.*



Connect with Me



LinkedIn



Medium



GitHub



Discord Server