## **What is your experience with AWS services such as EC2, S3, RDS, and Lambda?**

## I am a junior skilled engineer with boot camp experience in AWS DevOps. On the projects I successfully designed, deployed, and maintained various AWS- Based systems, including EC2, S3, RDS, and Lambda. I am familiar with automation tools such as CloudFormation, Terraform, Docker, Kubernetes and Ansible.

## I have done Capstone project which was including;

## Blog Page Application (Django) deployed on AWS Application Load Balancer with Auto Scaling, S3, Relational Database Service (RDS), VPC's Components, Lambda, DynamoDB and CloudFront with Route 53

## EC2 (Elastic Compute Cloud) :

## EC2 is a scalable virtual server provided by AWS.

## It allows you to launch and manage virtual machines in the cloud.

## S3 (Simple Storage Service) :

## S3 is a scalable object storage service offered by AWS.

## It provides secure and durable storage for various types of data,such as files, images, and videos.

## RDS (Relational Database Service):

## RDS is a managed database service that makes it easy to set up,operate, and scale relational databases in the cloud.

## It supports various database engines like MySQL, PostgreSQL, Oracle, and SQL Server

## Lambda: AWS Lambda is a serverless compute service.

## It lets you run your code without provisioning or managing servers. You can run functions in response to events or on a scheduled basis.

## **How do you manage infrastructure as code using tools like CloudFormation, Terraform, and AWS CDK?**

## We can manage Infrastructure as Code (IaC) Tools:

## CloudFormation: AWS CloudFormation is a service that allows you to define and provision AWS infrastructure resources using YAML or JSON templates. It helps you create and manage a collection of related AWS resources in a predictable and automated way.

## Terraform: Terraform is an open-source IaC tool that supports multiple cloud providers, including AWS. It uses a declarative language to define and provision infrastructure resources. Terraform enables you to manage infrastructure as code and provides features like resource dependencies, versioning, and state management.

## AWS CDK (Cloud Development Kit): AWS CDK is a software development framework that allows you to define cloud infrastructure using familiar programming languages such as TypeScript, Python, and Java. It provides higher-level abstractions and constructs to define AWS resources and helps generate CloudFormation templates.

## **What is your experience with configuring and managing CI/CD pipelines using tools like Jenkins, AWS Code Pipeline, and Code Deploy?**

## I have done Jenkins Pipeline project for Web Page Application (PostgreSQL-Nodejs-React) deployed on EC2's with Ansible and Docker.

## If I were to summarize this project:

## This project aims to create a Jenkins pipeline to deploy web-page written Nodejs and React Frameworks on AWS Cloud Infrastructure using Ansible and Jenkins.

## Building infrastructure process is managing with control node utilizing Ansible. This infrastructure has 1 Jenkins server (`Amazon Linux 2023 AMI`) as ansible control node and 3 EC2's as worker node (`Red Hat Enterprise Linux 9 with High Availability`). These EC2's launches on AWS console. Webpage has 3 main components which are PostgreSQL, NodeJS, and react. Each component is serving in Docker container on EC2s dedicated for them. PostgreSQL is serving as Database of webpage. Nodejs controls backend part of web-side and react controls frontend side of webpage.

## In summary:

## CI/CD Pipeline Tools

## Jenkins: Jenkins is an open-source automation server that facilitates building, testing, and deploying software projects. It supports various plugins and integrations, allowing you to configure custom CI/CD pipelines for different use cases.

## AWS Code Pipeline: AWS Code Pipeline is a fully managed CI/CD service. It helps you create and orchestrate end-to-end release pipelines that automate software delivery processes. Code Pipeline integrates with other AWS services and supports source code repositories, build providers, and deployment actions.

## Code Deploy: AWS Code Deploy is a deployment service that automates the application deployment process to EC2 instances, Lambda functions, or on premises servers. It simplifies the release management of applications and enables you to automate deployment strategies and rollbacks.

## **What is your experience with Docker and container orchestration tools like Kubernetes and Amazon ECS?**

## In the projects I have completed, one of them was the Bookstore Web API Application. Its aim was to create a bookstore web service using Docker in order to help me understand how to dockerize an application. The application code was deployed as a RESTful web service with Flask using Dockerfile and Docker Compose on an AWS Elastic Compute Cloud (EC2) Instance. This was done using Terraform.

## Another project I have completed is the Phonebook Microservice Web Application which aimed to create a web application with a MySQL database using Docker and Kubernetes to give me an understanding of Microservice architecture. This application consists of a frontend service and a backend service which interact with the database service. Each service is managed by a Kubernetes deployment. The backend service has a gateway for the application and serves the necessary web pages for creating, deleting, and updating operations, while the frontend service serves a search page for conducting read operations.

## If I were to summarize:

## Docker and Container Orchestration:

## Docker: Docker is an open-source platform that enables you to automate the deployment and management of applications within containers. It provides a lightweight and isolated environment for running applications and their dependencies.

## Kubernetes: Kubernetes is an open-source container orchestration platform that automates the deployment, scaling, and management of containerized applications. It provides features like service discovery, load balancing, and self-healing capabilities.

## Amazon ECS (Elastic Container Service): Amazon ECS is a fully managed container orchestration service provided by AWS. It supports running Docker containers at scale and integrates well with other AWS services.

## **5. How do you ensure the security of AWS resources and applications, and what tools and practices do you use?**

## If I explain briefly:

## Security of AWS Resources and Applications:

## Ensuring the security of AWS resources and applications is essential.

## Some common practices and tools used for security in AWS include:

## Identity and Access Management (IAM): IAM allows you to manage user identities and their access to AWS resources. It includes features like user authentication, access control policies, and multi-factor authentication (MFA).

## Encryption: AWS offers various encryption mechanisms to secure data at rest and in transit. This includes AWS Key Management Service (KMS) for managing encryption keys and SSL/TLS for secure communication.

## Network Security: AWS provides services like Virtual Private Cloud (VPC) to isolate and secure network resources. Network Access Control Lists (ACLs) and Security Groups help control inbound and outbound traffic.

## Monitoring and Logging: Services like AWS CloudTrail and AWS Config allow you to monitor and audit API activity and configuration changes in your AWS account. CloudWatch Logs provide centralized logging.

## **Describe your experience with monitoring and logging tools like**

## **CloudWatch, ELK stack, and Splunk.**

## **7. How do you manage secrets and credentials in a secure and scalable manner?**

## I think that managing secrets and credentials in a secure and scalable manner is crucial for protecting sensitive information in today's digital landscape. Here are some best practices and techniques we can employ:

## **Centralized Secret Management**: Use a dedicated secret management system or service, such as HashiCorp Vault, AWS Secrets Manager. This tool provides a secure and centralized location for storing and managing secrets.

## **Secure Access Controls**: Implement strict access controls to limit who can view and modify secrets. Follow the principle of least privilege, granting access only to the individuals or systems that require it. Use strong authentication mechanisms, such as multi-factor authentication (MFA), to secure access to the secret management system.

## **Secrets in Containers and Orchestration**: If you're using containerization technologies like Docker or orchestration frameworks like Kubernetes, avoid embedding secrets directly in container images or configuration files. Instead, utilize secrets management features provided by these platforms, such as Kubernetes Secrets or Docker Secrets.

## **8. What is your experience with automating infrastructure and application deployments using tools like Ansible, Chef, and Puppet?**

## I have several hands-on and Project experiences for Ansible.

## **9****. What is your experience with scaling and optimizing AWS resources for high availability, performance, and cost efficiency?**

## I have hands-on and project experience and I am always eager to explore new AWS services and techniques to further enhance high availability, performance, and cost efficiency for AWS resources in terms of performance optimization.

## I have studied with various AWS services to improve application performance. For example, I have utilized Amazon CloudFront to deliver content globally with low latency and implemented caching using Amazon ElastiCache. I have also optimized database performance by leveraging features such as read replicas and database caching with Amazon RDS.

## To ensure cost efficiency, I have focused on rightsizing AWS resources to match workload requirements. I have utilized AWS Cost Explorer and AWS Budgets to monitor and manage costs effectively. Additionally, I have made use of Reserved Instances for predictable workloads and spot instances for cost savings during non-critical periods.

## **10. How do you stay up to date with the latest AWS services and DevOps trends and best practices?**

## To stay up to date with the latest AWS services and DevOps trends and best practices, we can follow these strategies:

## AWS Official Resources: We can visit the AWS website regularly to explore the latest services, features, and updates.

## AWS Online Resources: Take advantage of AWS online resources like webinars, videos, and online training courses.

## Continuous Learning: Cultivate a mindset of continuous learning.

## Staying curious, exploring new technologies, and keeping ourselves updated with industry news and trends beyond just.

## Staying up to date with AWS services and DevOps practices is an ongoing process. By combining these strategies and actively engaging with the AWS community, we can remain current with the latest advancements and best practices in the AWS ecosystem.