S. Z.

DevOps Engineer

5+ years of experience

Core competencies

Highly skilled Senior DevOps Engineer leveraging 5+ years of hands-on experience in designing, implementing, and maintaining large-scale, highly available systems. Proficient in a variety of technologies, including containerization **(Docker, Kubernetes)**, cloud infrastructure **(AWS, GCP)**, automation (**Python, Bash**), CI/CD (**Github Actions, GitlabCI**), and infrastructure as a code and configuration management tools **(Ansible, Terraform)**. Strong background in developing applications, **Linux/Unix** administration, **networking**, and **security best practices**. Excellent problem-solving skills and ability to work in fast-paced, dynamic environments. Proven track record of delivering projects on time and within budget, while ensuring high availability and performance. Exceptional leadership abilities, fostering collaboration and mentoring junior team members to maximize productivity.

Technical capabilities

Cloud	**AWS** (5+ years), **GCP** (3+ year),
Containerization	**Kubernetes** (5+ years), **Docker** (5+years)
CI/CD	**GitHub Actions/GitlabCI** (4+ year), **Jenkins** (3+ year)
IaC	**Terraform** (4+ year), **CloudFormation** (4+ years)
Configuration Management	**Ansible** (5+years)
Monitoring	**Prometheus/Grafana** (3+ year), **New Relic**(3+year)
Database	**PostgreSQL**, **MySQL**(5+ years), **MongoDB**, **DynamoDB**(4+ years)
Scripting & Automation	**Python**(5+ years), **Js/Nodejs**(5+ years), **Bash**(3+ years)
Operating Systems	**RHEL/CentOS, Ubuntu**

Certifications

Certified Kubernetes Administrator🔗

Certified Kubernetes Application Developer🔗

Google Cloud Certified Associate Cloud Engineer🔗

HashiCorp Certified: Terraform Associate🔗

Education

Ala-Too International University **Bh. of Computer Science**

English: Advanced

Selected Projects

Cost Optimization: Efficiency Through Automation

Role FinOps / DevOps Engineer

Technologies AWS (EKS, RDS, OpenSearch, ElastiCache, Amazon MQ, CloudWatch CloudFormation), Python

Description:

Led a comprehensive cost optimization initiative within the E-commerce sector, with a primary focus on non-production environments. Conducted an in-depth analysis that uncovered instances of overprovisioned resources, including RDS, ElasticSearch, ElastiCache, Amazon MQ, and EKS clusters. Employed precise calculations to strategically resize these resources to smaller instance types, ensuring consistent performance while achieving substantial cost reductions.

Implemented a sophisticated Python-based automation system deployed on AWS Lambdas to manage RDS instances intelligently. This automation efficiently halts and restarts RDS instances during weekends and after working hours, optimizing resource utilization during periods of inactivity. Addressed inefficiencies in ElasticSearch and ElastiCache clusters through specialized automation, dynamically adjusting instance types to more cost-effective sizes during non-operational periods.

Achieved a noteworthy 61.3% reduction in RDS expenses and a commendable 20.1% savings for ElastiCache and ElasticSearch clusters. Developed an automation solution to scale down EKS node groups to a minimum during weekends, optimizing cloud infrastructure and minimizing operational costs.

Responsibilities:

- Led the entire project lifecycle, overseeing planning and execution for successful outcomes.
- Identified instances of overprovisioning, conducting detailed cost analyses for critical resources.
- Implemented strategic changes, resizing instance types for RDS, ElasticSearch, ElastiCache, Rabbit MQ, and EKS clusters.
- Introduced a Python-based automation system on AWS Lambdas for efficient RDS instance management.
- Developed innovative automation optimizing resource utilization during weekends and after working hours.
- Addressed ElasticSearch and ElastiCache inefficiencies through specialized automation.
- Monitored system performance to ensure cost optimization did not compromise overall efficiency.
- Evaluated results to ensure significant cost reduction without compromising system performance.
- Achieved a remarkable 61.3% reduction in RDS expenses and a commendable 20.1% savings for ElastiCache and ElasticSearch clusters.

- Implemented automation to scale down EKS node groups during weekends, optimizing cloud infrastructure and minimizing operational costs.
- Demonstrated a consistent track record of implementing cost-effective solutions and leveraging automation for enhanced cloud efficiency.

Automated IAM User Key Management with Ansible Integration

Role DevOps Engineer

Technologies AWS (IAM), Ansible, SMTP, Python

Description:

Developed a tailored Ansible module to automate the rotation of IAM User access keys, creating a scalable and efficient solution for managing access keys in an AWS IAM environment. The primary goal was to eliminate manual monitoring and rotation processes, enhancing security and reducing operational overhead.

The custom Ansible module, crafted to receive input parameters, primarily the user and group name, utilized the AWS SDK for Python (Boto3). It retrieved users within the specified group, assessed the expiration status of their access keys, and seamlessly generated new keys when nearing expiration. The automation also managed key deletion, and a secure email mechanism was implemented to share the new keys, ensuring a smooth transition without disrupting user access.

This solution significantly streamlined key rotation, eliminating manual intervention and reducing the risk of access key misuse or compromise. By automating this critical security task, the project not only enhanced the IAM environment's overall security posture but also improved operational efficiency.

Responsibilities:

- Designed and implemented a custom Ansible module for automated IAM User key rotation, adhering to best practices and security standards.
- Developed the Ansible module using Python and the Boto3 AWS SDK, leveraging IAM APIs for interaction with IAM Users and access keys.
- Incorporated error handling and logging mechanisms for module robustness and traceability.
- Conducted comprehensive testing and validation, simulating diverse scenarios to verify the key rotation process.
- Integrated the custom module into Ansible playbooks, facilitating seamless IAM User key rotation as part of broader infrastructure management workflows.
- Documented module functionality, usage instructions, and troubleshooting guidelines for easy adoption and knowledge transfer within the team.
- Collaborated with a team to deploy and configure the Ansible automation, ensuring proper integration with existing infrastructure and workflows.
- Monitored and maintained the automation solution, promptly addressing issues or enhancements.
- Actively participated in knowledge-sharing sessions, providing guidance on IAM best practices, automation techniques, and Ansible module development to junior team members.

CloudFormation Stack Refactoring and Segmentation for Minimizing Resource Interdependencies and Improved Control

Role DevOps Engineer

Technologies AWS (CloudFormation, AWS Services)

Description:

Led a comprehensive refactoring and segmentation initiative of a complex AWS CloudFormation stack, aimed at minimizing resource interdependencies and improving control over infrastructure changes. The project involved dissecting the existing monolithic CloudFormation stack into multiple logical stack groups, implementing best practices, and ensuring proper resource isolation and management.

The primary motivation for this undertaking was to address the challenge of unwanted effects caused by changes made to a specific resource within the monolithic stack. By breaking down the stack into smaller, logically separated components, each with its own CloudFormation stack, we were able to mitigate the risk of unintended impacts on other resources during infrastructure modifications.

The refactoring process encompassed a thorough analysis of the existing stack's resource dependencies, mapping out relationships, and identifying opportunities for segmentation. By carefully considering resource boundaries, we created smaller, more manageable stack groups, reducing interdependencies and improving the granularity of control over changes.

Throughout the project, industry best practices and AWS guidelines were followed, ensuring the new stack segmentation aligned with security, scalability, and reliability requirements. Automation techniques and infrastructure-as-code principles were leveraged to ensure consistent and reproducible deployments of the segmented stack groups.

The resulting architecture provided several advantages, including increased flexibility for making changes to individual components, faster stack deployments, simplified troubleshooting and debugging, and improved resource isolation, leading to enhanced resilience and stability.

Responsibilities:

- Analyzed the existing monolithic CloudFormation stack and assessed resource interdependencies.
- Developed a comprehensive refactoring plan, outlining the steps required for dividing the stack into smaller, more manageable stack groups.
- Implemented the refactoring process, ensuring the smooth transition of resources and maintaining consistency and integrity throughout the segmentation.
- Checked and resolved dependencies between the newly created stack groups, ensuring they were properly linked and integrated.
- Implemented best practices for resource naming, tagging, and configuration to adhere to industry standards and improve manageability.
- Leveraged infrastructure-as-code principles to automate the deployment and update processes of the segmented stack groups.
- Conducted rigorous testing and validation of the refactored stack groups, ensuring proper functionality and alignment with business requirements.
- Collaborated with the team to deploy and monitor the segmented stack groups, providing necessary support and guidance.
- Documented the refactoring process, including architectural diagrams, deployment guidelines, and troubleshooting documentation.

- Actively participated in knowledge-sharing sessions, sharing lessons learned, and providing guidance on best practices for CloudFormation stack segmentation and resource isolation.

Infrastructure Security Evaluation, Improvement Planning and Implementation

Role	DevSecOps Engineer
Technologies	AWS (CloudFront, ALB, S3, EC2, Lambda, DynamoDB, WAF, AWS Services) Security

Description:

Led a comprehensive infrastructure security evaluation, improvement planning, and implementation project focused on identifying and rectifying potential vulnerabilities and misconfigurations within the existing AWS infrastructure. The initiative included a thorough assessment of various AWS resources and the implementation of robust security measures to elevate the overall security posture.

The primary objective was to evaluate the existing infrastructure for security weaknesses, misconfigurations, and unauthorized access points. Key areas of scrutiny encompassed publicly exposed CloudFront and ALB endpoints, S3 bucket public access, EC2 security group permissions, and Lambda access to DynamoDB databases.

To mitigate these security risks, a series of remediation actions were executed. For publicly exposed CloudFront and ALB endpoints, Web Application Firewalls (WAFs) were implemented to protect against common web-based attacks and provide an additional layer of defense. Measures were taken to restrict public access to S3 buckets, enforcing proper access controls to prevent unauthorized access and data exposure.

Additionally, the security group permissions for EC2 instances underwent a thorough review and adjustment to ensure only necessary and appropriate inbound and outbound traffic was permitted. Access permissions for Lambda functions accessing DynamoDB databases were tightened, following the principle of least privilege, and tailored to the specific needs of each environment.

Throughout the project, adherence to security best practices and compliance standards was prioritized. Regular audits and vulnerability scans were conducted to identify any remaining security gaps, and prompt measures were taken to address them.

Responsibilities:

- Conducted a comprehensive evaluation of the existing infrastructure, assessing security vulnerabilities and misconfigurations.
- Collaborated with stakeholders, including security teams and infrastructure owners, to gather requirements and align security measures with business goals.
- Developed a detailed improvement plan, outlining necessary actions to enhance the security of various AWS resources.
- Implemented security measures, including WAF deployment for publicly exposed CloudFront and ALB endpoints, restriction of S3 bucket public access, and adjustment of EC2 security group permissions.
- Audited and adjusted Lambda access permissions to DynamoDB databases, ensuring adherence to the principle of least privilege.
- Collaborated with cross-functional teams to coordinate and execute remediation actions, minimizing disruptions to ongoing operations.

- Conducted regular vulnerability scans and audits to identify and address any remaining security gaps.
- Documented security configurations, policies, and procedures, providing comprehensive documentation for reference and future audits.

Kubernetes Application Migration from AWS (EKS) to GCP (GKE)

Role DevOps Engineer

Technologies AWS (EKS, Secret Manager, other AWS Services), Helm, RedHat Quay

GCP (GKE, Secret Manager, other GCP Services), Docker, Kubernetes

Description:

A complex and critical project involving the migration of existing applications running on Kubernetes clusters in Amazon Web Services (AWS) to Google Cloud Platform (GCP). The objective was to seamlessly transfer the Kubernetes infrastructure, including associated resources, to GCP's managed Kubernetes service (GKE), ensuring uninterrupted service delivery and maximizing the benefits of the GCP ecosystem.

The migration encompassed various aspects, starting with the planning and assessment phase, where existing applications, their dependencies, and associated resources were analyzed. Detailed mapping and documentation of the architecture, networking, and service discovery components were crucial in ensuring a smooth and accurate migration process.

One of the key tasks involved rewriting existing Kubernetes manifest files into Helm Charts, enabling easier deployment, versioning, and management of the applications in the GKE environment. Additionally, the migration project involved transferring other resources, such as persistent volumes, secrets, load balancers, and DNS records, from AWS to GCP equivalents, including Route53 to CloudDNS and AWS Secret Manager to Google Secret Manager.

Testing and validation played a vital role in this project to ensure that the migrated applications functioned correctly in the new environment. Extensive testing procedures were implemented to verify application performance, scalability, and reliability, mitigating any potential issues and ensuring a seamless transition for end-users.

Throughout the project, a strong focus was placed on security, compliance, and best practices for containerization and orchestration. Docker containers were utilized to package the applications, while Kubernetes and Helm provided the orchestration and deployment frameworks. Additionally, RedHat Quay was leveraged as a container registry to securely store and distribute container images.

The successful migration from AWS EKS to GCP GKE allowed the organization to take advantage of GCP's advanced features, scalability, and robust ecosystem while ensuring minimal disruption to operations and a smooth user experience.

Responsibilities:

- Conducted a thorough assessment of the existing AWS EKS environment, identifying dependencies, resources, and potential challenges.
- Designed and implemented the migration strategy, outlining the step-by-step approach for transferring the applications, networking components, and associated resources to GCP GKE.
- Rewrote Kubernetes manifest files into Helm Charts, providing a more streamlined and efficient method for deploying and managing applications in the GKE environment.

- Transferred resources such as persistent volumes, secrets, load balancers, and DNS records from AWS to GCP equivalents, ensuring seamless connectivity and functionality.
- Collaborated with security teams to ensure compliance with security and access control policies during the migration process.
- Conducted rigorous testing and validation of the migrated applications, addressing any issues and optimizing performance for the GCP environment.
- Worked closely with other teams to facilitate a smooth transition, ensuring minimal impact on end-users and continuous service delivery.
- Documented the migration process, including architectural diagrams, deployment guidelines, and troubleshooting documentation, ensuring knowledge transfer and future reference.
- Provided guidance and support to the team, sharing best practices and lessons learned from the migration project.

Implementing IAM Roles for Seamless S3 Access for Magento Applications running in EKS

Role DevOps Engineer

Technologies AWS (EKS, S3, IAM), Kubernetes, Ansible, Security

Description:

Implemented IAM Roles for seamless S3 access for Magento applications running in an Amazon Elastic Kubernetes Service (EKS) cluster. Before this implementation, an IAM user with S3 permissions was created, and access keys were generated for this user. These access keys were then provided to the Magento application pod within the Kubernetes environment, enabling the application to interact with S3. However, this approach posed challenges in terms of key management and rotation, requiring manual intervention for key creation and updates.

To address this problem, a more secure and scalable solution was devised. A service account was created and bound with a dedicated IAM Role that had the necessary S3 permissions. By specifying this service account in the deployment configuration, the Magento application gained seamless access to S3 without the need for key rotations or manual user creations.

Responsibilities:

- Designed and implemented the IAM Role-based access control solution for S3 access for Magento applications.
- Collaborated with the development team to understand the application's requirements and integrate the IAM Role into the Kubernetes deployment process.
- Configured the IAM Role with the appropriate S3 permissions to ensure the application had the necessary access without unnecessary privileges.
- Created and managed the service account in Kubernetes, establishing the link between the IAM Role and the application deployment.
- Conducted testing and validation to verify the seamless S3 access from the Magento application.
- Documented the IAM Role implementation process and provided guidance to the team for future reference.
- Worked closely with security and compliance teams to ensure the solution adhered to best practices and met the organization's security requirements.

- Proactively monitored and addressed any issues related to IAM Role permissions, S3 access, or Kubernetes deployment to ensure the availability and reliability of the application.

Improving Operational Efficiency with Ansible-based IAM User Provisioning

Role DevOps Engineer

Technologies AWS (IAM), Ansible, SMTP, Python

Description:

Implemented an Ansible-based solution to enhance operational efficiency through automated IAM user provisioning. The project aimed to streamline the process of creating IAM users within an AWS environment, reducing manual effort and ensuring consistent user provisioning across the organization.

A custom Ansible module was developed to facilitate the automation of IAM user creation. The module accepted user input in the form of an email address and an optional user group name. Leveraging the capabilities of the AWS SDK for Python (Boto3), the module programmatically created an IAM user using the provided email address. Furthermore, the automation seamlessly generated access keys for the newly created user, enabling secure programmatic access to AWS resources. Additionally, if a user group name was specified, the module added the user to the respective IAM user group to enforce access control policies.

To ensure a smooth onboarding experience, the module generated the access keys and shared them with the user via email, along with any pertinent instructions or policies. By automating these provisioning tasks, the project significantly reduced the administrative overhead and potential errors associated with manual user creation and access key generation.

Responsibilities:

- Designed and developed a custom Ansible module for automated IAM User provisioning, aligning with best practices and security standards.
- Implemented the Ansible module using Python and Boto3, leveraging the IAM APIs to create IAM Users, generate access keys, assign users to groups, and send access keys via email.
- Conducted extensive testing and validation of the module, ensuring its robustness and compatibility across different IAM configurations and environments.
- Integrated the custom module into Ansible playbooks and workflows, allowing for seamless and consistent user provisioning across the infrastructure.
- Created comprehensive documentation, including usage instructions, troubleshooting guidelines, and best practices for IAM User provisioning.
- Monitored and maintained the IAM User provisioning automation, proactively addressing any issues, optimizing performance, and incorporating necessary enhancements.
- Actively participated in knowledge-sharing sessions and guided team members on IAM provisioning automation, Ansible usage, and AWS IAM best practices.

Enhancing Infrastructure Management Efficiency with Terraform Modules

Role DevOps Engineer

Technologies Terraform, AWS (EKS, RDS, VPC, S3, ALB, EC2 and other AWS Services),

GCP (GKE, CloudSQL, VPC and other GCP Services)

Description:

Led a transformative initiative to enhance infrastructure management efficiency through the adoption and implementation of Terraform modules. The project focused on automating the deployment of cloud infrastructure across multiple cloud providers, including AWS and GCP, using reusable and modular Terraform code.

The primary objective of this project was to streamline and standardize the provisioning and management of infrastructure components across different environments. By developing a set of well-defined and reusable Terraform modules, we aimed to eliminate manual intervention and reduce the time and effort required to deploy and maintain infrastructure.

The project involved comprehensive research and analysis of the organization's infrastructure requirements, including common patterns and recurring components. Leveraging Terraform's powerful infrastructure-as-code (IaC) capabilities, we designed and implemented a library of modular Terraform modules that encapsulated infrastructure provisioning logic and best practices.

These modules were carefully crafted to cater to different use cases, encompassing the provisioning of virtual machines, networks, storage resources, security groups, and other essential infrastructure components. They were designed to be easily customizable and adaptable to different environments and cloud providers, promoting consistency and reducing the risk of configuration drift.

Responsibilities:

- Conducted a thorough assessment of the existing infrastructure landscape, identifying patterns and recurring components.
- Researched and evaluated best practices for infrastructure provisioning and management using Terraform.
- Designed and implemented a library of reusable Terraform modules, aligning them with the organization's infrastructure requirements and industry best practices.
- Developed comprehensive documentation for each Terraform module, including usage instructions, configuration parameters, and integration guidelines.
- Collaborated closely with development teams to ensure alignment with business requirements and seamless integration with existing workflows.
- Conducted testing and validation of the Terraform modules, ensuring their compatibility across different environments and cloud providers.
- Integrated the Terraform modules into CI/CD pipelines and deployment workflows, automating the infrastructure provisioning process.
- Implemented version control and release management practices for the Terraform modules, ensuring traceability and accountability.
- Collaborated with security and compliance teams to ensure adherence to security standards and best practices within the Terraform modules.
- Mentored and guided junior team members, fostering knowledge sharing and promoting the adoption of Terraform best practices.

Orchestrating Microservices with ECS and OpenSearch: CI/CD Pipeline and Infrastructure as Code

Role DevSecOps Engineer

Technologies AWS (ECS, ECR, OpenSearch, CloudFormation), GitHub Actions, Docker

Description:

Led the successful deployment and orchestration of a microservice application using AWS Elastic Container Service (ECS) and OpenSearch while implementing a robust CI/CD pipeline and Infrastructure as Code (IaC) practices. The project encompassed the seamless integration of multiple technologies to ensure efficient and reliable application deployment and management.

The project started with designing and implementing a CI/CD pipeline using GitHub Actions. The pipeline was configured to automatically build and push container images to AWS Elastic Container Registry (ECR) whenever changes were pushed to the associated GitHub repository. This streamlined the development process and enabled quick and automated deployments of the microservice application.

The microservice application followed a three-tier architecture, with ECS serving as the container orchestration platform and OpenSearch acting as the database layer. The ECS configurations were written in CloudFormation, enabling Infrastructure as Code practices to provision and manage the necessary resources and infrastructure components. The CloudFormation templates captured the desired state of the ECS clusters, services, tasks, and related networking configurations, ensuring consistent and reproducible deployments.

A critical aspect of the project involved deploying and integrating OpenSearch as the database for the microservice application. The OpenSearch cluster was provisioned and configured within the ECS environment, allowing seamless communication and data storage for the microservices.

Responsibilities:

- Designed and implemented the microservice application deployment strategy using AWS ECS and OpenSearch.
- Developed and managed the CI/CD pipeline using GitHub Actions, enabling automated builds and deployments of container images to AWS ECR.
- Collaborated with the development team to define best practices and standards for containerization, deployment, and version control of microservices.
- Configured the ECS clusters, services, tasks, load balancers, and networking components using CloudFormation templates, ensuring Infrastructure as Code principles.
- Integrated OpenSearch as the database layer, ensuring seamless communication and data storage for the microservice application.
- Conducted thorough testing and validation of the deployment process, addressing any issues or performance bottlenecks.
- Implemented monitoring and alerting mechanisms for the microservices and infrastructure components, ensuring high availability and performance.
- Created and maintained documentation for the deployment process, CI/CD pipeline, and infrastructure configurations, promoting knowledge transfer and best practices.
- Developed and maintained the CloudFormation, ensuring version control and continuous improvement of the infrastructure-as-code approach.

Upgrading EKS and GKE Clusters

Role DevOps Engineer

Technologies AWS (EKS), GCP (GKE), Kubernetes, Python, Helm, Terraform

Description:

Led a critical project involving the seamless upgrade of existing Amazon Elastic Kubernetes Service (EKS) and Google Kubernetes Engine (GKE) clusters, ensuring the adoption of the latest Kubernetes versions and associated resources. The project leveraged the power of Terraform, a leading infrastructure-as-code tool, to automate and streamline the upgrade process, enabling efficient tracking of changes and providing the ability to roll back if necessary.

The upgrade process involved multiple stages, starting with a thorough assessment of the existing EKS and GKE clusters, including their Kubernetes versions, node groups, worker nodes, and networking components. Based on this analysis, an upgrade plan was devised to ensure minimal disruption to services and applications running on the clusters.

Using Terraform, we crafted declarative code to orchestrate the cluster upgrade, allowing for consistent and reproducible deployments. This infrastructure-as-code approach provided traceability, version control, and the ability to roll back to previous configurations in case of any unforeseen issues or incompatibilities.
Additionally, the project addressed the deprecation of Kubernetes APIs by updating all manifest files and Helm charts to use the latest recommended APIs and configurations. This ensured compatibility and smooth operation of the applications running on the upgraded clusters.

Responsibilities:

- Conducted a comprehensive assessment of the existing EKS/GKE clusters, documenting the Kubernetes versions, node groups, worker nodes, and networking components.
- Designed and implemented an upgrade plan, considering potential impacts on applications, services, and dependencies.
- Developed Terraform code to automate the cluster upgrade process, adhering to best practices and infrastructure-as-code principles.
- Conducted rigorous testing and validation of the upgrade process in staging environments, addressing any issues or incompatibilities.
- Collaborated with the team to schedule and execute the production upgrade, minimizing disruption and ensuring the availability of services.
- Monitored the upgrade process, closely tracking metrics, and promptly addressing any issues or performance concerns.
- Ensured proper version control and documentation of the Terraform code, maintaining a clear record of the infrastructure changes made during the upgrade.
- Collaborated with the development and testing teams to verify the compatibility of applications and services with the upgraded clusters.
- Provided guidance and support to the team, sharing best practices for cluster upgrades, Kubernetes versioning, and infrastructure management.

Building a Scalable Serverless Web Application Infrastructure on AWS

Role DevOps Engineer

Technologies AWS (CloudFront, S3, DynamoDB, Lambda, API Gateway), Python
GitHub Actions

Description:

Successfully implemented a scalable serverless web application infrastructure on AWS. This project focused on creating a fully managed and highly scalable infrastructure that could handle the traffic and load of the web application while minimizing operational overhead and costs.

The project commenced by designing and implementing an automated CI/CD pipeline using GitHub Actions. The pipeline was configured to detect source code changes and automatically push the changes to an S3 bucket, enabling continuous integration and deployment of the web application.

The web application was deployed on S3, leveraging its cost-effectiveness and scalability. CloudFront was used to provide a globally distributed content delivery network (CDN) for efficient content delivery and caching. Route53 was used for DNS management, ensuring smooth and reliable access to the web application. To provide dynamic functionality, Lambda functions were implemented as the backend for the serverless web application. Python was used as the primary programming language for developing the Lambda functions, enabling seamless integration with other AWS services.

For persistent data storage, DynamoDB, a fully managed NoSQL database, was utilized. DynamoDB provided a scalable and high-performance data storage solution, accommodating the application's evolving needs.

Responsibilities:

- Designed and implemented the infrastructure for the serverless web application on AWS, ensuring scalability, high availability, and cost-effectiveness.
- Developed and maintained the CI/CD pipeline using GitHub Actions, automating the build and deployment process of the web application.
- Configured S3 for hosting the website, leveraging CloudFront for global content delivery and caching.
- Set up Route53 for DNS management, ensuring reliable and seamless access to the web application.
- Developed Lambda functions in Python to serve as the backend for the serverless web application, integrating with other AWS services as needed.
- Designed and implemented the data storage solution using DynamoDB, ensuring scalability and performance for the application's data needs.
- Conducted rigorous testing and validation of the web application infrastructure, addressing any issues related to scalability, performance, and security.
- Implemented monitoring and logging solutions to ensure optimal performance and identify and resolve any operational issues proactively.
- Documented the infrastructure setup, deployment processes, and best practices, enabling easy adoption and knowledge transfer within the team.
- Collaborated with the team to optimize the application's architecture for serverless deployment, improving performance and cost efficiency.
- Provided guidance and support, fostering collaboration and knowledge sharing on serverless architectures, AWS best practices, and infrastructure management.

Efficient MySQL Database Backup Automation with Jenkins Pipeline

Role DevOps Engineer

Technologies MySQL, Jenkins, Bash scripting

Description:

Implementation efficient MySQL database backup automation project using Jenkins Pipeline. The objective was to automate the creation of database dumps for different environments, ensuring data integrity and easy restoration in case of data loss or corruption.

The project involved leveraging Jenkins to create a pipeline that streamlined the backup process. For each environment, a separate bastion host was configured, with each bastion host having access to its own MySQL database instance.

The Jenkins Pipeline was designed to accept build parameters specifying the target environment. Based on these parameters, the pipeline is connected to the respective bastion host using SSH, utilizing the secure shell protocol for secure remote access.

Using bash scripting within the Jenkins Pipeline, the pipeline executed the necessary MySQL commands to create a database dump, ensuring proper naming and organization. The database dump was then saved within the bastion host, providing a centralized location for data backups.

Responsibilities:

- Designed and implemented the automation process for MySQL database backup using Jenkins Pipeline.
- Configured separate bastion hosts for each environment, ensuring secure and isolated access to the respective MySQL database instances.
- Developed the Jenkins Pipeline, incorporating build parameters to specify the target environment for the backup process.
- Implemented SSH connectivity within the pipeline to securely connect to the bastion hosts for database dump creation.
- Utilized bash scripting to execute MySQL commands within the Jenkins Pipeline, generating proper database dumps with appropriate naming conventions.
- Configured proper file organization and storage within the bastion hosts, ensuring easy accessibility and retrieval of backups.
- Conducted rigorous testing and validation of the backup process, verifying the integrity of the database dumps.
- Implemented monitoring and alerting mechanisms to ensure the successful execution of the backup pipeline and timely resolution of any issues.
- Documented the backup automation process, including step-by-step instructions, configuration details, and troubleshooting guidelines.
- Provided guidance and support to the team, fostering knowledge sharing and best practices for MySQL backup automation, Jenkins Pipeline, and secure access controls.

Custom Integration: Sending RDS Postgres Metrics to New Relic for Advanced Monitoring

Role DevOps Engineer

Technologies AWS (RDS, Cloud Watch), PostgreSQL, New Relic

Description:

Implemented advanced monitoring for RDS Postgres instances by integrating them with New Relic. The goal was to collect and send table-level metrics from the RDS instances to New Relic for enhanced visibility and monitoring capabilities. This project addressed the limitations of default CloudWatch metrics provided by AWS for RDS instances.

To achieve this, the project involved setting up a custom integration using a Bastion host. The bastion hosts were configured with access to the RDS instances, allowing them to establish connections and gather the necessary metrics. The on-host integration on the bastion host facilitated the collection of specific table-level metrics from the RDS instances, which were not available through standard AWS CloudWatch monitoring

Responsibilities:

- Designed and implemented a custom integration to send RDS Postgres metrics to New Relic for advanced monitoring.
- Collaborated with developers to identify the specific table-level metrics required for monitoring and troubleshooting.
- Configured the bastion hosts with the necessary access permissions to connect to the RDS instances securely.
- Set up the on-host integration on the bastion hosts, enabling the collection and forwarding of the desired metrics to New Relic.
- Developed the necessary scripts and configurations to extract the table-level metrics from the RDS instances and send them to New Relic.
- Conducted thorough testing and validation of the integration, ensuring the accuracy and reliability of the collected metrics in New Relic.
- Documented the integration process, including step-by-step instructions, configuration details, and troubleshooting guidelines.

Terraform Automation Application for VM Instance Lifecycle Management

Role Developer / DevOps Engineer

Technologies Terraform, VMware, JavaScript/NodeJs

Description:

The project aimed to revolutionize VM instance lifecycle management for one of the top universities in the USA by implementing a customized solution centered around Terraform automation. The university had a significant number of on-premises instances managed by VMWare, and their Infrastructure as Code (IaC) was scripted using Terraform.

The primary objective was to reduce manual intervention, enhance efficiency, and streamline operations related to the provisioning, modification, and decommissioning of VMs. Designed and developed a

user-friendly web application interface to simplify interactions with VM instances. This included creating intuitive dashboards and controls for provisioning, modifying, and decommissioning VMs.

By leveraging Terraform automation and developing a user-friendly web application interface, the project successfully minimized manual tasks, optimized efficiency in VM instance lifecycle management, and ultimately contributed to streamlined operations and improved operational efficiency for the university.

Responsibilities:

- Conducted thorough analysis and understanding of the university's existing infrastructure setup, including their VMWare environment and Terraform scripts.
- Collaborated with stakeholders to design a tailored solution centered around Terraform automation. This involved architecting a system that seamlessly integrated with the university's infrastructure while meeting their specific requirements.
- Conducted rigorous testing of the Terraform modules, web application, and their integration to ensure reliability, scalability, and security.
- Prepared comprehensive documentation detailing the solution architecture, implementation steps, and usage guidelines to facilitate seamless adoption and future maintenance.
- Provided training sessions to university staff on utilizing the new solution effectively. Offered ongoing support and troubleshooting assistance to address any issues encountered post-implementation.
- Oversaw the project timeline, budget, and resource allocation. Coordinated with cross-functional teams to ensure timely delivery and successful implementation of the solution.

Implementing GitOps with ArgoCD: Automating Application Deployment on Kubernetes

Role DevOps Engineer

Technologies AWS (EKS), GCP (GKE), GitOps, ArgoCD, Kubernetes, Helm, Kustomize

Description:

Implemented GitOps practices using ArgoCD, revolutionizing the way applications were deployed and managed on Kubernetes. This project involved leveraging Git as the single source of truth for declaratively managing the desired state of applications and infrastructure, enabling automated, reliable, and auditable deployment workflows.

The project began by designing and implementing an automated process for creating ArgoCD applications for existing Helm charts and Git repositories. This involved defining the necessary configurations and templates to ensure seamless integration between ArgoCD and the application repositories. By leveraging Helm, the popular package manager for Kubernetes, we achieved consistency and repeatability in application deployments.

Central to this project was the setup of a multi-environment ArgoCD setup, enabling the management of application deployments across different environments. With the use of Kustomize, a native Kubernetes configuration management tool, we were able to manage environment-specific configurations and ensure proper separation of concerns. This allowed for the streamlined deployment of applications across various environments, such as development, staging, and production.

Responsibilities:

- Designed and implemented the GitOps strategy using ArgoCD, enabling declarative management of application deployments on Kubernetes.
- Developed automated processes for creating ArgoCD applications, seamlessly integrating with existing Helm charts and Git repositories.
- Define GitOps workflows, establishing Git as the single source of truth for application deployments.
- Configured and maintained the ArgoCD setup, ensuring high availability, security, and scalability of the platform.
- Implemented multi-environment ArgoCD setups, managing deployments across different environments using Kustomize for environment-specific configurations.
- Establish best practices for managing infrastructure as code within the GitOps framework.
- Conducted extensive testing and validation of the GitOps workflows, ensuring reliability, consistency, and fault tolerance.
- Developed and maintained documentation for the GitOps processes, including configuration guidelines, troubleshooting steps, and best practices.
- Provided guidance and support to the development and operations teams, facilitating the adoption of GitOps practices and ArgoCD.
- Conducted training sessions to educate team members on GitOps principles, ArgoCD usage, and best practices for application deployment on Kubernetes.