DevOps Engineer

5+ years of experience

Core competencies

Experienced Senior DevOps Engineer with over 5 years of hands-on experience in designing, implementing, and maintaining robust CI/CD pipelines, scalable infrastructure, and high-availability systems. Proficient in a wide range of technologies including containerization(Docker), infrastructure as code(Terraform), cloud platforms(AWS, GCP) monitoring tools(Prometheus,Grafana), scripting languages(Bash). Proven leadership qualities, robust problem-solving techniques, and clear communication to promote teamwork and produce significant outcomes.

Technical capabilities

Cloud	AWS (4+ years), GCP (2+ years)
IaC	Terraform (4+ years), CloudFormation(1 year)
Containerization	Docker(4+ years) Kubernetes (4+ years)
CI/CD	Jenkins(4+ years), CircleCi(3+ years), GitHub Actions (4+ years)
Monitoring	CloudWatch, Datadog (4+ year), Prometheus+Grafana(4+ years)
Database	Postgres, MySQL, DynamoDB (4+ years)
Scripting	Python (Boto3), Bash (4+ years)
Operating Systems	Ubuntu, RHEL/CentOS

Education

Ala-Too International University Bh. of Computer Science

Certifications

Certified Jenkins Engineer

HashiCorp Certified: Terraform Associate

Certified Kubernetes Administrator

Selected Projects

Provisioning Infrastructure with Terraform and Automating Application Deployment with GitHub Actions

Role: **DevOps Engineer**

Technologies:

- Kubernetes, Docker,
- AWS(VPC, ALB, WAF, ECS, ECR, CloudTrail, CloudWatch, OpenID, IAM, RDS(postgresql)),
- GitHub Actions

Description:

Implemented Infrastructure as Code (IaC) using Terraform to provision AWS cloud infrastructure, including services like VPC, ECS, ALB, WAF, and RDS. Automated the building, testing, and pushing of Docker images, deploying the application to ECS using a pipeline.

The project began with a need to establish a streamlined and efficient deployment workflow for a complex application stack running on AWS infrastructure. Initial challenges included manual provisioning processes, lack of standardized deployment practices, and limited automation capabilities. The goal was to implement Infrastructure as Code (IaC) principles using Terraform and automate the CI/CD pipeline with GitHub Actions to improve deployment speed, reliability, and security.

Automated infrastructure provisioning and application deployment, resulting in increased operational efficiency and reduced manual intervention. By adopting Infrastructure as Code (IaC) principles with Terraform and leveraging GitHub Actions for automation, we significantly improved the deployment process's speed, reliability, and security. The streamlined workflows and documented processes ensure scalability and maintainability, providing a solid foundation for future development and expansion initiatives.

Responsibilities:

- Designed and implemented the infrastructure architecture to support the application's deployment needs.
- Collaborated with team members to select appropriate technologies and deployment strategies.
- Developed Terraform modules for provisioning AWS networking components, security groups, and database resources.
- Configured AWS WAF to protect the application from common web-based security threats.
- Implemented load balancing using ALB to distribute incoming traffic across multiple ECS instances.
- Integrated RDS (PostgreSQL) with ECS for efficient database management and data persistence.
- Established a secure connection to the ECS cluster using a Bastion Host for administrative tasks.
- Monitored the health and performance of the infrastructure using CloudTrail and CloudWatch.
- Set up an EKS cluster with node pools and autoscaling capabilities to optimize resource allocation based on workload demands.
- Automated the Docker image build, test, and push process to Amazon ECR using GitHub Actions workflows.
- Orchestrated continuous integration of the application codebase using GitHub Actions for streamlined development workflows.
- Implemented Helm charts for managing Kubernetes configurations of individual microservices.
- Configured secure authentication and authorization between GitHub and AWS using OpenID.
- Established a continuous deployment pipeline with human approval options to ensure controlled releases to production.
- Conducted thorough testing and validation of all deployment processes to guarantee reliability and consistency.
- Documented project components, processes, and best practices for knowledge sharing and future reference.

Setting up SonarQube for Infrastructure (GKE)

Role: **DevOps Engineer**

Technologies: **SonarQube, GKE(Kubernetes), VPC, IAM, Kong server, CircleCI**

Description:

Led the initiative to deploy SonarQube across the Kubernetes clusters within our infrastructure, aiming to enhance code quality analysis and automate the deployment process. The project involved meticulous configuration and integration of various technologies to ensure seamless operation within the Google Cloud Platform (GCP) environment.

Setting up SonarQube for Infrastructure was a pivotal endeavor to elevate our code quality assessment capabilities within the Kubernetes ecosystem. The objective was to establish SonarQube as a central tool for conducting comprehensive code analysis and enforcing quality standards across all development projects.

Responsibilities:

- Orchestrating the deployment of SonarQube within the Kubernetes environment, ensuring scalability and availability.
- Fine-tuning Kubernetes components such as Ambassador, External-DNS, and Kong Apache to optimize integration with SonarQube and facilitate seamless communication.
- Implementing automation pipelines with CircleCI to streamline the deployment and configuration process, enabling swift creation of new environments and ensuring consistency across deployments.
- Setting up Cloud DNS to manage domain resolution and facilitate secure communication with SonarQube server instances.
- Implementing SSL certificate configuration to encrypt communications and enhance security posture within the infrastructure.

Deploying a Java Application to AWS Elastic Beanstalk and RDS

Role: DevOps Engineer

Technologies:

- AWS Elastic Beanstalk, IAM (Identity and Access Management), Route 53, RDS (Relational Database Service) S3 (Simple Storage Service)
- GitHub Actions
- Java Spring
- Terraform

Description:

Led the deployment of a Java Spring Boot application to AWS Elastic Beanstalk and RDS. This initiative aimed to establish a scalable and reliable infrastructure for hosting the application within the AWS ecosystem, utilizing managed services to streamline deployment workflows.

The project began with provisioning AWS Elastic Beanstalk environments tailored to the Java Spring Boot application's requirements. Configuration parameters were optimized for scalability, performance, and cost-efficiency. Simultaneously, RDS instances were created to serve as the backend database, ensuring data durability and availability.

Designed and implemented IAM roles to facilitate secure deployment via GitHub Actions. These roles adhered to the principle of least privilege, ensuring strict access control and compliance with security best practices.

A robust deployment pipeline was developed using GitHub Actions to automate the deployment process to AWS Elastic Beanstalk. This pipeline encompassed stages for building, testing, and deploying the Java Spring Boot application, ensuring thorough validation at each step.

Comprehensive testing was conducted to verify the functionality, performance, and compatibility of the application within the AWS environment. This included unit tests, integration tests, and end-to-end tests to identify and address any issues proactively.

Responsibilities:

- Provisioning AWS Elastic Beanstalk environments tailored to the application's requirements.
- Setting up RDS instances to serve as the backend database for the application.
- Designing and implementing IAM roles for secure deployment via GitHub Actions.
- Developing a robust deployment pipeline in GitHub Actions for automated deployment to Elastic Beanstalk.
- Conducting comprehensive testing of the Java application to ensure functionality and performance.

- Authoring documentation outlining deployment procedures and best practices for knowledge transfer and future reference.

Automated Infrastructure Provisioning with Terraform

Role: DevOps Engineer

Technologies:

- Kubernetes
- Terraform
- AWS (VPC, RDS(MySQL), EKS, ECR, S3, Secret Manager, ALB, NAT Gateway, WAF, Internet Gateway, CloudTrail, CloudWatch, ASG)
- AppMesh
- GitHub Actions

Description:

Spearheaded the implementation of Terraform for managing infrastructure, utilizing modules for provisioning various AWS resources including VPC, IAM, RDS, EKS, ECR, and EC2. This initiative aimed to automate the provisioning of infrastructure components, ensuring consistency, scalability, and reliability across environments.

The project involved designing and implementing the full infrastructure stack, configuring essential components such as Bastion Host for secure cluster access, AWS networking modules for VPC, subnets, and route tables, and integrating services like WAF for application protection and ALB for load balancing.

Key responsibilities included setting up RDS (MySQL) to seamlessly integrate with EKS, storing sensitive information securely in Secret Manager, and implementing monitoring solutions using CloudTrail and CloudWatch to track infrastructure activities and performance metrics.

Furthermore, an EKS cluster with node pool and autoscaling capabilities was configured to dynamically adjust resource allocation based on workload demands, ensuring optimal performance and cost-efficiency.

Comprehensive documentation in the form of a README file was authored to provide guidelines for utilizing Terraform modules, configuring infrastructure components, and managing deployments. This documentation served as a reference for team members and facilitated knowledge transfer within the organization.

Responsibilities:

- Designed and implemented the full infrastructure stack using Terraform modules.
- Configured Bastion Host for secure cluster access.
- Developed AWS networking modules including VPC, subnets, route tables, NAT Gateways, and Internet Gateway.
- Configured WAF to protect the application from security threats.
- Implemented load balancing using ALB to distribute traffic to the application.
- Configured RDS (MySQL) to integrate with EKS for seamless database management.
- Stored sensitive information securely in the Secret Manager.
- AppMesh
- Monitored infrastructure activities and performance metrics using CloudTrail and CloudWatch.
- Configured EKS cluster with node pool and autoscaling capabilities for dynamic resource allocation.
- Authored documentation (README file) for Terraform modules to guide users in utilizing and configuring infrastructure components.

Migration of Organization Source Code from BitBucket to GitHub and GitHub Organization Management with Terraform and Jenkins Pipeline

Role: DevOps Engineer

Technologies: BitBucket, GitHub, Terraform, Jenkins

Description:

Migrated the organization's source code from BitBucket to GitHub Enterprise while implementing an automated solution for managing the GitHub organization using Terraform and Jenkins Pipeline. The migration was driven by the need to centralize repository management and standardize processes across the organization. The migration process itself was meticulously planned and executed, with Terraform automating the creation and configuration of GitHub repositories, teams, and access controls. Jenkins Pipeline orchestrated the Terraform deployment process, automating tasks such as repository creation and access control configuration, ensuring accurate and efficient migration

The successful completion of the project resulted in a centralized and standardized approach to GitHub organization management. Manual interventions were minimized, and administrative tasks were automated, leading to increased efficiency and productivity across the organization. Additionally, the use of Terraform and Jenkins facilitated continuous integration and deployment (CI/CD) practices, enabling rapid and consistent delivery of changes to the GitHub organization infrastructure.

Responsibilities:

- Led the planning and design of the migration strategy, considering factors such as repository structure, permissions, and version control history preservation.
- Conducted a thorough analysis of the organization's structure, identifying users, teams, repositories, default branches, and protection rules. Collaborated with stakeholders to understand specific requirements and dependencies.
- Worked closely with coworkers to define granular permissions for each repository, ensuring appropriate access control post-migration. Developed role-based access policies to enforce security standards.
- Managed the end-to-end migration process, ensuring seamless transfer of source code from BitBucket to GitHub Enterprise. Implemented migration scripts and performed data validation to maintain integrity.
- Developed modular Terraform modules to manage existing organization resources and automate the creation of new resources, such as repositories and teams. Ensured code reusability and scalability.
- Imported existing resources from GitHub into Terraform state, providing a centralized configuration management solution. Implemented version control mechanisms to track changes effectively.
- Conducted rigorous testing of Terraform code logic to ensure scalability, flexibility, and consistency across various organizational components. Performed integration testing to validate infrastructure changes and configurations.
- Produced comprehensive documentation detailing Terraform modules, including configuration guidelines, usage instructions, and troubleshooting steps. Facilitated knowledge transfer and supported ongoing maintenance efforts.

Implementing GitOps with ArgoCD: Automating Application Deployment on Kubernetes

Role: DevOps Engineer

Technologies: 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:

- Developed automated processes for creating ArgoCD applications, seamlessly integrating with existing Helm charts and Git repositories.
- Designed and implemented the GitOps strategy using ArgoCD, enabling declarative management of application deployments on Kubernetes.
- DefineGitOpsworkflows,establishingGitasthesinglesourceoftruthforapplicationdeployments.
- 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.

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.

Automation of Building, Testing, and Deployment of Java Application

Role: DevOps Engineer

Technologies: Kubernetes, Docker, Helm, AWS (EKS, Secret Manager, ECR, OpenID), GitHub Actions, Maven

Description:

Led the implementation of a comprehensive automation solution to streamline the deployment process of Java Maven microservices. Our objective was to revolutionize the development lifecycle, ensuring efficiency and reliability in deploying applications to the Kubernetes environment on Amazon EKS. Leveraging cutting-edge technologies such as Kubernetes, Docker, Helm, AWS services, GitHub Actions, and Maven, we automated every aspect of the deployment pipeline, from building new Docker images to deploying them to the microservices environment on EKS.

Automation resulted in a streamlined and efficient deployment pipeline for Java Maven microservices to Amazon EKS. Leveraging automation and collaboration, it ensured consistent delivery of high-quality software with rigorous testing, efficient containerization, and comprehensive documentation. This approach drives continuous

improvement and innovation within the organization, facilitating rapid and reliable deployment of new features and updates.

Responsibilities:

- Spearheaded the design and implementation of automation solutions tailored for deploying Java Maven application microservices to EKS.
- Collaborated closely with team members to define and refine the release process, ensuring alignment with project goals and requirements.
- Developed and fine-tuned pipelines using GitHub Actions to automate the building and testing of Maven packages, enabling rapid iteration and feedback loops.
- Automated the process of building, testing, and pushing Docker images to Amazon ECR, optimizing the containerization workflow and ensuring consistency in image management.
- Orchestrated the continuous integration of the application using GitHub Actions, facilitating seamless integration of code changes and automated testing.
- Implemented Helm charts for all microservices to simplify deployment, configuration, and management on Kubernetes clusters.
- Configured continuous deployment pipelines with the option for human approval, enabling controlled and efficient rollouts of new application versions.
- Secured integration with AWS using OpenID from GitHub, ensuring controlled access to resources.
- Conducted rigorous testing and validation of the entire deployment process, ensuring reliability, scalability, and resilience under various conditions.
- Authored detailed and comprehensive documentation for each component of the automation workflow, providing valuable insights, best practices, and troubleshooting guidelines for team members and stakeholders.

Automation of Monitoring for EKS Clusters and EC2 Instances

Role: DevOps Engineer

Technologies: Kubernetes, Terraform, AWS (VPC, EKS, EC2, S3, Secret Manager, ALB), Prometheus, Grafana, Thanos, GitHub Actions

Description:

Led the provisioning and configuration of monitoring tools to ensure comprehensive visibility and management of EKS clusters and EC2 instances. The primary objective was to establish a centralized monitoring solution using Prometheus, Grafana, and Thanos, facilitating efficient alerting, metric collection, and long-term storage of monitoring data. Leveraging technologies such as Kubernetes, Terraform, AWS services, and GitHub Actions, we aimed to enhance observability and optimize resource utilization across the infrastructure.

Responsibilities:

- Designed and implemented the entire monitoring infrastructure, ensuring scalability and resilience.
- Configured exporters and agents to collect metrics from EKS clusters and EC2 instances, enabling real-time monitoring.
- Developed Terraform code to provision Prometheus, Grafana, and Thanos for all clusters, ensuring consistency and reproducibility.
- Configured Data Sources for Grafana to integrate with Prometheus, enabling visualization of monitoring data.
- Implemented a strategy for storing metrics for long-term analysis in Amazon S3, optimizing resource utilization and cost-efficiency.
- Deployed Service Discovery mechanisms to dynamically monitor EC2 instances, ensuring comprehensive coverage of the infrastructure.
- Established centralized monitoring, enabling unified visibility and management of all clusters and instances.
- Conducted thorough testing of connections between Prometheus masters and agents, ensuring reliability and data integrity.
- Authored comprehensive documentation outlining the setup, configuration, and maintenance procedures for the monitoring infrastructure, providing valuable insights and guidelines for team members and stakeholders.