Abdul Rehman

DevOps Engineer with CKA and AWS

Phone No: +91-8639853393.

Email: abduldevops9@gmail.com

CARRIER OBJECTIVE:

5+ years of experience as a **DevOps Engineer,** managing end to end Cloud Infrastructure Creation, Management, Integration, Automation and Deployment.

Professional Summary:

- Proficient in **Linux** system administration, scripting, and deployment tasks.
- Ability to quickly understand, learn and implement the new system design, data models in professional work environment.
- Worked on feature branches in **Git**, making changes and pushing them to **GitHub**.
- Configuring and managing Apache **Tomcat** for application deployment.
- **Jenkins** monitors the GitHub repository for changes. Upon detecting new commits, it triggers a build to compile the code and run automated tests.
- **SonarQube** analyses the code for quality issues, providing feedback to developers. If issues are detected, we can address them before proceeding.
- **Nexus** as Artifactory to resolve and publish project artifacts.
- **Jenkins** facilitates deployment, whether it's deploying to a test environment for further testing or releasing the application to production.
- Leveraged **ArgoCD's GitOps** principles to promote declarative configuration and version control of Kubernetes resources, ensuring consistency across environments.
- Cloud computing expertise, particularly on **AWS**, for deploying, managing and optimizing applications.
- **Terraform** scripts define and provision the **AWS** infrastructure.
- Jenkins orchestrates the build process, including **Docker** image creation and pushing to DockerHub.
- **Kubernetes** manifests are updated to reference the new Docker image version.
- Created **Helm** charts for various applications.
- Proven ability to design and implement scalable, reliable, and highly available systems.
- Experience building **micro-services** and deploying them into Kubernetes cluster.
- Kubernetes automatically deploys and manages containers based on the updated manifests.
- **Ansible** playbooks automate the configuration of Linux machines.
- **Prometheus** scrapes metrics from the application, and **Grafana** provides visualizations.
- Logs are sent to **Elasticsearch**, and **Kibana** facilitates log visualization and analysis.

- Worked closely with our development team to create an automated continuous integration (CI) and continuous delivery (CD) system.
- Strong emphasis on **Automation** to streamline tasks, improve efficiency, and enhance reliability.
- Dedicated to staying updated on industry trends and emerging Technologies.
- Passionate about adopting best practices to enhance the software development life cycle.
- Collaborative team player with excellent communication skills.

Professional Certifications:

- Certified Kubernetes Administrator (CKA) – ***Linux Foundation, 2025.***

Educational Qualification:

Bachelor of Technology from **JNTUH, Hyderabad in 2019.**

Technical Skills:

Operating Systems	Linux, windows
Version Control Systems	Git, GitHub
Scripting Languages	Shell Script and Python
Application Web Server	Tomcat
Build Tools	Maven
Continuous Integration	Jenkins, GitHub Actions
Continuous Deployment	Jenkins, ArgoCD
Code Quality	SonarQube
Containerization	Docker
Orchestration Tool	Kubernetes
Artifactory Repository Manager	Nexus
Cloud	AWS
Infrastructure As a Code	Terraform
Configuration Management	Ansible
Monitoring And Visualisation	Prometheus, Grafana and ELK Stack

Professional Experience:

Worked as a DevOps Engineer at Gaman Software Solutions Since July 2019.

Project :- 1
Responsibilities:

- In our project, **Git** serves as the version control system, providing a reliable history of code changes. **GitHub** acts as the central repository, facilitating collaboration among developers and ensuring a centralized location for source code management. Git repositories, adds collaboration features such as **pull requests, code reviews,** and **issue tracking.**

- In the context of our application, **Maven** automates the **compilation, testing, and packaging processes**. It manages project dependencies and plug-in, ensuring consistency across development environments. The integration of Maven with Git enables seamless dependency resolution and ensures that all developers are using the same project configuration.
- **SonarQube** is incorporated into the build process in our system, automatically analysing code changes. It gives feedback, allowing them to fix issues early in the development process. This integration guarantees that coding standards and best practices are followed in the code.
- In our Java-based project, Maven seamlessly integrates with **Nexus** as Artifactory. These tools manage artifices, allowing to publish and retrieve dependencies efficiently. The integration reduces the risk of using outdated or conflicting libraries and enhances build speed by caching artifices locally.
- In our application, **Jenkins** the entire development work flow. It monitors GitHub repositories for changes, triggers Maven builds, conducts code quality analysis with SonarQube, and manages artifices with Nexus as Artifactory. Troubleshooting of build errors in Jenkins for both the software configuration management team and the development team. Jenkins seamlessly integrates with the tools in our ecosystem, ensuring a smooth and automated CI/CD pipeline.
- **Shell scripts** are utilized in our set-up for operations such as environment setup, configuration, and deployment on Linux servers. **Python scripts** used to automate more sophisticated scenarios, such as connecting with AWS services or orchestrating several processes. These scripts increase the overall automation process's flexibility and adaptability.
- In our scenario, Jenkins is configured to deploy the Java application to an Amazon EC2 instance, leverage the scalability and reliability of **AWS** and configuring all necessary services like **Security, EC2, EBS, ECR, VPC, NAT, Auto Scaling, Route53, S3, AWS CLI, Elastic Load balancers**, can be integrated.
- Implemented the release practice and responsible for pushing builds into QA, UAT, Preproduction and Production stages.
- To create users, roles and groups using Identity Access Management (**IAM**) and attached the required policies can be integrated based on project requirements.
- Once the build is successful, **Jenkins** used for continuous integration and deploys the application to an AWS EC2 instance using deployment scripts. The application's scalability and reliability are ensured through AWS cloud services. Continuous monitoring and integration for efficient scaling and management of resources.

Project :- 2
Responsibilities:

- Developers commit code changes to **Git** repositories on **GitHub**.
- In our work flow, **Docker** will be used for building the docker image of the application, pushing to **DockerHub**, and running docker containers.
- Create **Terraform** scripts to define the AWS infrastructure required for the application, including virtual private cloud (VPC), subnets, security groups, IAM roles-users-policies,

S3 bucket and any other necessary resources. Leverage Terraform to manage the life-cycle of cloud resources, from provisioning to modification and deletion. This ensures consistency and repeatability in infrastructure deployments.

- In our work flow, **Ansible** will be used to configure for hosting the application. To define the desired state, write **Ansible Play-books** and **Ansible Roles**. Tasks such as installing needed packages, defining environment variables, handlers, loops, and deploying the application can be included in play-books.
- **Jenkins** manages the entire development work flow in our application. It checks GitHub repositories for changes, starts Maven builds, analyses code quality with SonarQube, and manages artifices using Nexus as Artifactory. Troubleshooting build issues in Jenkins for both the software configuration management team and the development team. Jenkins interacts perfectly with the technologies in our ecosystem, delivering a smooth and automated pipeline
- Worked with the **Kubernetes** manifests (YAML files) describing pod configurations, services, and deployments. Kubernetes resources like **Deployments, Services, PV, PVC, State-full-Set, Ingress, RBAC, EKS, Cluster Auto Scaler, Pod Affinity, Pod Anti-affinity, Liveness Probe, Readiness Probe, ConfigMaps, Secrets** and **Helm** can be defined to manage the application's administration and extend the deployment **Jenkins** pipeline.
- A very good understanding of Deployment Strategies like Blue/Green Deployment, Roll-out Deployment and Recreate Deployment.
- Utilize Kubernetes features for scaling applications horizontally by adjusting the number of replicas and performing rolling updates to deploy new versions without downtime using **HPA.**
- Deploy **Prometheus** and **Grafana** in the Kubernetes cluster as pods. Configure Prometheus to scrape metrics from the application endpoints. Use Grafana to create dashboards displaying key metrics.
- Used **Logstash** to ingest logs into **Elasticsearch**. Deploy **Kibana** alongside Elasticsearch to visualize and search logs. Create dashboards to monitor log events and troubleshoot issues.
- The expanded work flow now incorporates Docker for containerization, Kubernetes for orchestration, Prometheus and Grafana for monitoring, and the ELK Stack for centralized logging.

Project :- 3

Responsibilities:

- Collaborated with the development team to define and refine the CI/CD workflow requirements, ensuring alignment with project goals and best practices.
- Implemented version control strategies within the **GitHub** repository, facilitating collaboration, code review, and traceability of changes.
- Configured **GitHub Actions** triggers and workflows to respond to events such as code pushes, pull requests, and builds, optimizing the development lifecycle.
- Employed environment variables and secrets management in GitHub Actions to securely store sensitive information such as API keys and credentials.

- Implemented **Dockerfile** optimizations to improve build performance and enhancing deployment efficiency.
- In our work flow, **Docker** will be used for building the docker image of the application, pushing to **DockerHub**.
- Configured **ArgoCD** to automate the deployment process. Kubernetes manifests, including deployment.yaml, service.yaml, and hpa.yaml, defined the desired state of the application within the Kubernetes cluster. These manifests were stored within the GitHub repository, ensuring version control and easy access.
- Ensured deployment reliability and consistency by adhering to **GitOps** principles, which emphasize the use of version-controlled configurations and declarative deployment practices.
- Deploy **Prometheus** and **Grafana** in the Kubernetes cluster as pods. Configure Prometheus to scrape metrics from the application endpoints. Use Grafana to create dashboards displaying key metrics.
- Monitored and optimized CI/CD pipeline performance, identifying bottlenecks and implementing optimizations to enhance build and deployment speed.
- Documented CI/CD pipeline setup, configuration, and troubleshooting procedures for team reference and knowledge sharing.
- Provided mentorship and support to team members on CI/CD best practices, tools, and techniques, fostering a culture of continuous improvement and innovation.

Declaration:

I hereby declare that the information above is true and correct to the best of my knowledge and belief and can refer above for verification.

Abdul Rehman