Nensi Ravaliya

☎ +91 9313181628 ✉ ravaliyanensi@gmail.com LinkedIn GitHub Portfolio Blogs

PROFILE

DevOps Engineer with expertise in automating cloud infrastructure on **AWS** and **Azure**, utilizing tools such as **Linux**, **Scripting** using **Bash** and **Python, Terraform, Docker, Kubernetes and Ansible**. **Microsoft Azure Certified** and **GitHub Admin** and **Actions Certified**, adept at creating efficient CI/CD pipelines through **Jenkins** and **Azure DevOps**. Committed to optimizing deployments, improving security, and ensuring system scalability, with a focus on comprehensive documentation and proactive monitoring for enhanced reliability.

WORK EXPERIENCE

DevOps Trainee | Einfochips An Arrow Company| July – October 2024

- **Optimized cloud infrastructure** on AWS and Microsoft Azure by implementing Terraform for infrastructure-as-code (IaC), reducing provisioning time by 50% and increasing infrastructure scalability.
- **Designed and deployed containerized applications** using Docker and Kubernetes, achieving a 30% reduction in deployment errors and a 25% increase in deployment efficiency.
- **Automated cloud infrastructure management** with Ansible playbooks, reducing manual configuration tasks by 60% and ensuring consistent environments across development, testing, and production.
- **Streamlined CI/CD pipelines** with Jenkins and Azure DevOps, resulting in a 40% improvement in build and deployment efficiency, and ensuring faster and more reliable software releases.
- **Enhanced security and domain management**, configuring SSL/TLS certificates, DNS (CNAME, TXT, MX) records, and ensuring secure communication and compliance, reducing security vulnerabilities by 30%.
- **Collaborated on multi-cloud strategies**, improving fault tolerance and cost optimization by leveraging AWS and Azure managed services, leading to a 20% reduction in cloud operating costs.
- **Developed Helm charts** for Kubernetes-based applications, enabling easy and rapid deployments across environments, reducing update times by 30%.
- **Automated monitoring and alerting** for cloud-native applications using Prometheus, Grafana, and custom dashboards, improving incident response time by 40% and overall system reliability.
- **Adopted remote state management** using Terraform with backend storage in AWS S3 and Azure Blob, ensuring secure and consistent state management, improving collaboration and reducing configuration drift.
- **Detailed documentation** for every project, covering architecture, deployment processes, configurations, CI/CD pipelines.

SKILLS

- **Tools and Software**: Linux, Ansible, Azure DevOps, Helm
- **Containerization and Orchestration**: Docker, Kubernetes
- **Cloud Platforms**: AWS, Azure
- **Version Control Systems**: Git, GitHub, GitLab, Bitbucket
- **Infrastructure as Code**: Terraform
- **Scripting**: Python, Bash
- **CI/CD Tools**: Jenkins, Azure DevOps, GitHub Actions
- **Monitoring and Observability**: Grafana, Prometheus
- **DevSecOps tools**: SonarQube

PROJECTS

E-Commerce Cloud Infrastructure Automation | Source Code

- **Tech Stack:** AWS (EC2, VPC, S3, Route Tables, Security Groups), Kubernetes, Terraform, Ansible, Jenkins, Docker, Helm, GitHub
- **Key Features:**

  Infrastructure Provisioning: Automated provisioning of cloud resources using Terraform, including VPC, EC2 instances, S3 buckets, route tables, security groups, and subnets for network isolation.

Master Node Configuration: Configured Kubernetes master and worker nodes using Ansible playbooks, installing Kubernetes components (kubeadm, kubelet, kubectl) and connecting worker nodes to the master node.

Jenkins CI/CD Pipeline: Set up a Jenkins pipeline for continuous integration and deployment, automating source code checkout, Docker image creation, pushing images to Docker Hub, and deploying the application to Kubernetes using Helm.

Application Deployment: Deployed a Node.js e-commerce application to Kubernetes worker nodes, exposed via a Kubernetes service to ensure secure and scalable access.

- **Security:** Enhanced security with role-based access control (RBAC) in Kubernetes, and restricted network access using security groups and private subnets for sensitive components.
- **Impact:** Reduced deployment time by 50% and operational costs by 30%, while achieving 99.99% uptime. The automated CI/CD pipeline improved deployment efficiency by 40%, allowing for faster and more reliable application updates.

Multi-Cloud DevSecOps Automation| Source Code

- **Tech Stack:** AWS, Azure, Azure DevOps, Terraform, SonarQube, OWASP ZAP, Prometheus, Grafana, Loki, Promtail
- **Key Features:**

Multi-Cloud Integration: Developed a CI/CD pipeline using Azure DevOps, integrating AWS and Azure services to deploy and manage infrastructure across both cloud platforms.

Security-first Approach: Implemented automated security checks during the build process using SonarQube for code quality analysis and OWASP ZAP for vulnerability scanning.

Infrastructure as Code (IaC): Automated infrastructure provisioning on AWS using Terraform, integrated into the Azure DevOps pipeline to ensure consistency and reliability in deployments

Monitoring & Observability: Leveraged Prometheus for metrics collection, Grafana for visualizations, and Loki and Promtail for log aggregation, providing comprehensive observability and monitoring of the infrastructure and application.

- **Impact:** Improved the deployment process across multiple cloud platforms, increasing deployment speed by 40% and lowering security risks by 30%. Enabled better monitoring and faster issue resolution using observability tools, resulting in a 25% increase in system uptime and stability.

Node.js CI/CD Pipeline with AWS Fargate| Source Code

- **Tech Stack:** Node.js, Docker, GitHub Actions, AWS Elastic Container Registry (ECR), AWS Elastic Container Service (ECS), Fargate
- **Key Features:**

Automated CI/CD Pipeline: Set up a GitHub Actions pipeline to automatically build, push Docker images to AWS ECR, and deploy the application to AWS ECS Fargate whenever changes are pushed to the main branch**.**

Containerization: Dockerized the Node.js application to ensure consistency across development, testing, and production environments.

ECR and ECS Integration: Integrated AWS ECR for storing Docker images and ECS Fargate for deploying the containerized application, ensuring a scalable and serverless deployment process.

- **Impact**: Improved the deployment process by 60%, cutting down on manual tasks and making deployments faster and more reliable. Enabled smooth scaling of the application using AWS ECS with Fargate, optimizing resources and reducing operational costs.

ASP.NET Web Application Deployment on Azure Kubernetes(K8s) Services (AKS)| Source Code

- **Tech Stack:** ASP.NET, Docker, Azure Kubernetes Service (AKS), Azure DevOps
- **Key Features:**

Containerization: Developed a Dockerfile for containerizing the application, ensuring consistency across different environments.

Azure Repos Integration: Pushed application changes, including the Dockerfile, to Azure Repos for version control.

Image Building and Deployment: Utilized Azure Container Registry (ACR) to build and push Docker images for deployment.

Kubernetes Deployment: Configured and deployed the application on AKS using deployment.yml and service.yml files for resource management and service exposure.

- **Impact:** Simplified the deployment process for ASP.NET applications, making them easier to scale and more reliable by using Azure's cloud features.

EDUCATION

Government Engineering College Bhavnagar | B.E. Computer Engineering | 8.64 CGPA

CERTIFICATIONS

Microsoft Azure Certified AZ-900 | Microsoft
GitHub Admin Certified | GitHub
GitHub Actions Certified | GitHub