Aravind G.J

DevOps Engineer

jagan0614@outlook.com +1 940-489-2758 United States

Professional Summary

Experienced DevOps Engineer with 9 years of building, automating, and managing cloud infrastructure across AWS and Azure. I've worked with cross-functional teams to deliver secure, scalable, and efficient solutions that support real-world business needs.

· Designed and managed scalable infrastructure on **AWS (EC2, S3, Lambda, RDS)** to support high-availability applications, with a strong focus on automation, performance, and security.
· Provisioned and maintained **Azure** resources, including **Virtual Machines**, **Blob Storage**, and **Key Vault**, enabling secure hybrid cloud solutions aligned with enterprise architecture goals.
· Automated infrastructure deployment using **Terraform** and **CloudFormation** for **AWS** and **ARM** templates for **Azure**, ensuring environment consistency and reducing setup times.
· Developed and maintained **CI/CD** pipelines using **Jenkins** for build orchestration and **Azure DevOps** for release management, enabling faster, more reliable software delivery with built-in testing, approvals, and rollback capabilities.
· Built **Jenkins** pipelines to handle multi-stage deployments, integrated with **Git** workflows for version control and collaboration.
· Containerized applications using **Docker** and deployed to **ECS Fargate** and **Azure Kubernetes Service** (**AKS**) to increase portability, resource efficiency, and uptime.
· Led the deployment of scalable microservices on **Kubernetes**, including **Helm**-based configuration, **ingress** management, and secure multi-tenant environments.
· Implemented **Kubernetes** auto-scaling, health checks, and rolling updates to ensure resilience and minimal downtime during deployments. · Administered **Kubernetes RBAC** policies and namespace isolation to enforce workload separation and compliance requirements.
· Used **Ansible** to automate server provisioning, application deployments, and system configuration across environments, improving speed and consistency.
· Created reusable **Ansible** playbooks for common infrastructure tasks, streamlining team onboarding and daily operations.
· Set up centralized monitoring and alerting using **CloudWatch**, **Datadog**, **Azure Monitor**, and **ELK**, ensuring proactive issue detection and resolution.
· Configured log shipping and analysis using **Fluent Bit**, **CloudWatch Alarms**, and **Azure Log Analytics**, supporting 24/7 visibility into production environments.
· Enforced identity and access management using **AWS IAM**, **Azure RBAC**, and Active Directory, supporting least-privilege principles and regulatory compliance.
· Reduced operational costs by implementing auto-scaling, resource scheduling, and storage optimization policies such as **S3** lifecycle rules and **Azure** Cost Management.
· Designed and tested multi-region disaster recovery strategies with cross-region backups, achieving critical RTO/RPO targets for business continuity.
· Collaborated with cross-functional **Agile** teams to deliver cloud-native solutions, promote **DevOps** culture, and accelerate release cycles.

Professional Experience

Sr. Cloud Infrastructure Engineer- OCC, 03/2024 – 06/2025

Project: DevOps Infrastructure Automation

- Developed reusable **Terraform** modules to automate **AWS** infrastructure provisioning, significantly cutting down manual setup time and improving consistency across environments.
- Managed remote **Terraform** state using **S3** and **DynamoDB** to ensure safe collaboration, version control, and locking during infrastructure updates.
- Deployed and managed containerized applications on **AWS EKS**, orchestrating workloads with **Kubernetes** for better scalability, high availability, and environment isolation.
- Created **Helm** charts and **Kubernetes** YAML files to standardize application deployments, configured liveness/readiness probes, and implemented rolling updates for seamless releases.
- Built **CI/CD** pipelines using **Jenkins** to automate testing, building, and deployment stages, improving software delivery speed and reliability.
- Designed and implemented canary deployments using **Kubernetes Ingress** with **ALB** weighted routing, minimizing production risks during feature rollouts.
- Used **Ansible** to automate server configuration tasks across **EC2** and **EKS** nodes, ensuring consistent environment setup and reducing deployment errors.
- Configured **Kubernetes secrets**, **config maps**, and persistent storage to securely manage configuration data and stateful workloads.
- Set up **Prometheus** and **Grafana** dashboards to monitor cluster performance, resource usage, and application health in real time.
- Integrated **Fluent Bit** with **ELK Stack** and **AWS CloudWatch** for centralized log collection, making it easier to trace issues and improve incident response time.
- Reduced cloud infrastructure costs by implementing **EC2** and **EKS** auto-scaling and scheduled shutdowns for non-production environments, saving up to 35% on compute spend.
- Optimized **Docker** images using multistage builds and implemented them into the **CI/CD** pipeline, improving build efficiency and reducing deployment size.
- Documented provisioning steps, deployment processes, monitoring configurations, and troubleshooting procedures to support onboarding and smooth operations.

Environment: AWS (EC2, S3, EKS, IAM, CloudWatch, DynamoDB), Kubernetes, Helm, Terraform, Jenkins, Git, Ansible, Docker, Prometheus, Grafana, Fluent Bit, ELK Stack, Python, Bash, CI/CD, Infrastructure as Code, DevOps Automation

Cloud Automation Engineer – Ocwen Financial Solutions, 05/2020 – 11/2023

Project: DevOps Modernization of Loan Servicing System

- Led the migration of a legacy document processing platform to **AWS**, using **EC2**, **S3**, and **RDS** to improve scalability, fault tolerance, and cost-efficiency.
- Containerized application components using **Docker**, enabling consistent environments and simplifying deployment across development and production.
- Automated **CI/CD** workflows using **GitHub Actions,** streamlining build and deployment processes triggered by code pushes and pull requests.
- Maintained and optimized **Jenkins** pipelines for advanced deployment use cases, including test automation, artifact handling, and parallelized stages.
- Utilized **Chef** to manage system configurations and enforce consistency across environments, reducing configuration drift and improving reliability.

- Defined infrastructure using **Terraform** and **CloudFormation**, enabling reproducible, version-controlled provisioning of **AWS** resources.
- Monitored system performance and reliability with **AWS CloudWatch** dashboards and alarms, helping the team respond proactively to incidents.
- Aggregated and analyzed logs using **Fluent Bit** and the **ELK Stack**, improving troubleshooting efficiency and incident response times.
- Used **Docker** Compose for local development setups and multi-container integration testing, accelerating development cycles.
- Created and maintained **Chef** cookbooks to automate environment provisioning and manage application-specific configurations.
- Embedded security and quality gates into **GitHub Actions** workflows, automating linting, unit tests, and infrastructure scans before release.
- Improved deployment speed by 0% through **Jenkins** parallelization and build agent tuning, reducing bottlenecks during release cycles.
- Architected **AWS** infrastructure with multi-AZ redundancy, ensuring high availability and disaster recovery readiness.
- Applied **S3** lifecycle rules and intelligent tiering to manage growing storage costs, achieving monthly savings of 20%.
- Collaborated with developers and security teams to implement **DevSecOps** practices, including **IAM** role management, **KMS** encryption, and audit logging for compliance.

Environment: AWS (EC2, S3, RDS, CloudWatch, IAM, KMS, CloudFormation), Docker, Docker Compose, GitHub Actions, Jenkins, Chef, Terraform, Fluent Bit, ELK Stack, Python, Bash, CI/CD, DevSecOps, Infrastructure as Code, Multi-AZ Architecture

Sr. DevOps Engineer- Ocwen Financial Solutions, 05/2019 – 05/2020
Project:DevOps Automation for Customer Data Platform

- Migrated 20+ legacy on-prem applications to **Microsoft Azure** using **Virtual Machines**, **SQL Database**, **Blob Storage**, and **Virtual Network** to address scalability and operational overhead, while building a **DevOps** automation layer that improved delivery speed and reduced manual tasks.
- Deployment cycles were inconsistent and slow, so I implemented full **CI/CD** pipelines using **Azure DevOps**, **Git**, and **Terraform**, which cut release time from hours to under 15 minutes and improved deployment reliability.
- Automated the provisioning of **Azure** App Services, **Load Balancers**, and Auto-Scaling rules using Terraform and **ARM** templates, enabling repeatable infrastructure setups across all environments.
- Enabled zero-downtime deployments by adopting blue-green deployment strategies with **Azure** App Service slots, resulting in seamless and safer releases.
- Connected **Azure Key Vault** with our **CI/CD** pipelines to securely handle secrets and credentials, removing hardcoded values and improving our overall security posture.
- Introduced centralized monitoring using **Azure Monitor**, **Log Analytics**, and custom alerts, which helped reduce incident response time and provided real-time visibility into infrastructure health.
- Legacy architecture caused delays in document processing, so I designed a serverless, event-driven system using **Azure Functions**, **Event Grid**, and **Service Bus**, reducing message delivery latency by 18%.
- Manual access control was inconsistent and insecure, so I automated **RBAC** and policy management via **Azure** CLI scripts, decreasing unauthorized access incidents by 90% and improving audit readiness.
- Built a multi-region disaster recovery plan using **Azure Site Recovery** and **Geo-Redundant Storage (GRS)**, achieving **RTO** under 1 hour and **RPO** under 15 minutes for critical systems.
- Enforced **PCI-DSS** compliance by implementing encryption with **Azure Key Vault** and **TLS**, and applying **NSGs** and segmentation to isolate sensitive workloads.

Environment: Microsoft Azure, Azure Virtual Machines, Azure SQL Database, Azure Blob Storage, Azure Virtual Network, Azure App Services, Azure DevOps, Terraform, ARM Templates, Azure Functions, Azure Event Grid, Azure Service Bus, Azure Site Recovery, Azure Monitor, Log Analytics, Azure Key Vault, Azure CLI, NSG, TLS, RBAC, Git, CI/CD, PowerShell Scripting, Agile, Scrum, Runbooks

DevOps Engineer – Pacem Tech Solutions, 06/2016 – 04/2019

Project: DevOps Pipeline for Document Processing

- I helped modernize a legacy document system that was challenging to update by utilizing **Docker**, and I established **CI/CD** pipelines using **Jenkins** and **Git**. This really streamlined our deployments and cut down on manual errors.
- To avoid delays and inconsistencies in releasing updates, I containerized our services, deployed them on **EC2**, and set up **ALB** and **S3** for smoother performance and better uptime.
- We had some serious security concerns with how secrets were being handled, so I brought in **AWS** Secrets Manager to securely manage credentials. It reduced the chances of sensitive data being exposed.
- Diagnosing system slowdowns was a challenge, so I created custom **CloudWatch** dashboards and used Log Insights to help the team spot issues quicker and take action right away.
- There were concerns about permissions being too broad, so I reviewed and rewrote **IAM** policies to apply least-privilege principles, tightening access without impacting productivity.
- During peak usage, users experienced slow response times, so I reworked our APIs with asynchronous processing. That alone improved responsiveness by 12%.
- Backups and log management were manual and often missed, so I automated those processes with scripts and **S3**. That improved data reliability and saved time.
- Documentation around deployment processes was missing or outdated, so I created clear runbooks and diagrams to support the team onboarding and reduce dependency on individual engineers.
- Our infrastructure wasn't built for failure, so I redesigned it with **Docker** containers behind a load balancer and used multi-AZ **RDS** for fault tolerance. Uptime and reliability improved significantly.
- We didn't have consistent tagging across **AWS** resources, which made cost tracking difficult, so I created and enforced a tagging strategy that helped identify and manage usage more effectively.
- The team didn't have clear insights into how our systems were running, so I set up **CloudWatch** dashboards and alerts, helping everyone stay on top of performance and potential issues.

Environment: AWS EC2, S3, RDS (PostgreSQL), CloudFormation, Application Load Balancer (ALB), AWS CloudWatch, Log Insights, AWS Secrets Manager, AWS KMS, AWS CloudTrail, IAM, Docker, Jenkins, Git, FastAPI, Python, Redis, Shell Scripting, CI/CD, Infrastructure as Code (IaC), Auto Scaling, RBAC, Asynchronous API Optimization, Multi-AZ Deployment, Monitoring & Alerting, Backup Automation

CI/CD Engineer – Pacem Tech Solutions, 11/2014 – 06/2016

Project: DevOps-Driven Helpdesk System

- Deployments were taking too long and often broke in different environments, so I built fully automated **CI/CD** pipelines using **Jenkins** and **Git** for our and FastAPI- based helpdesk system. This sped up deployments and made them more consistent.
- Updating builds and running tests was manual and error-prone, so I introduced automated build, test, and deployment stages. This helped us catch issues earlier and release features more confidently.
- Tracking system errors and debugging issues was difficult without centralized logs, so I integrated the **ELK Stack**, which gave us clearer visibility and made root cause analysis much faster.
- There wasn't a clear access control system in place, so I implemented **RBAC** using 's built-in permission framework. This helped protect sensitive areas of the system and aligned access with user roles.

- Developers were juggling different accounts across tools, so I connected our authentication system to **LDAP**. This simplified logins and made user management much easier.
- Users had difficulty searching the knowledge base, so I enabled full-text search in **PostgreSQL**. This made it easier for them to find solutions and reduced repeated questions to support.
- High-priority issues weren't being escalated quickly, so I created automated workflows to handle escalations. This ensured urgent tickets were prioritized and resolved faster.
- File uploads were scattered and unorganized, so I built a secure upload system that included metadata tagging. It improved how attachments were managed and helped with audits and traceability.

Environment: FastAPI, Jenkins, Git, ELK Stack (Elasticsearch, Logstash, Kibana), PostgreSQL, Python, Shell Scripting, Cron Jobs, RBAC, LDAP, Permission Framework, Audit Trails, Full-Text Search, CI/CD, Automated Testing, Metadata Tagging, Ticket Classification Algorithms, Workflow Automation, Training Documentation

Technical Skills Summary

Programming Languages
Python, Bash, SQL

Databases
PostgreSQL, MySQL, Redis, Elasticsearch

Data Engineering
ETL Pipelines, Data Validation, Data Modeling, Data Catalogs, AWS Glue

Authentication & Security
IAM, OAuth2, OpenID Connect, LDAP, RBAC, PCI-DSS, GDPR Compliance, KMS Encryption

Other Tools
Jira, Agile & Scrum Methodologies, Pydantic, SQLAlchemy, Fabric

DevOps & IaC
Docker, Git, Jenkins, Terraform, Shell Scripting

Messaging & Queues
Celery, RabbitMQ

Monitoring & Logging
Datadog, AWS CloudWatch, Prometheus, Grafana, ELK Stack (Elasticsearch, Logstash, Kibana)

AWS Services
EC2, RDS, S3, VPC, Lambda, Step Functions, SQS, Kinesis, CloudFormation, Glue, Redshift, QuickSight, CloudTrail, Lake Formation

Education & Certifications

Masters in Computer Science, *University of Central Missouri* 05/2025

Education & Certifications

AWS Certified Solutions Architect – Associate, 2025
Amazon Web Services (AWS)