Nurmukhamed

DevOps Engineer

5 years of experience

Experienced DevOps Engineer with 5 years of hands-on experience in architecting, automating, and optimizing mission-critical deployments across diverse infrastructures. Proficient in containerization tools such as Docker and Kubernetes, and skilled in developing robust CI/CD pipelines.

As a collaborative team player with strong organizational and analytical skills, I excel in cross-functional environments, aligning development, operations, and quality assurance efforts. With excellent communication skills, I effectively convey complex technical concepts to both technical and non-technical stakeholders.

In summary, I bring extensive experience and expertise to drive efficiency and innovation as a DevOps Engineer. With a focus on collaboration and continuous improvement, I am well-equipped to contribute to the success of any DevOps team.

Technical Capabilities

Cloud	**AWS(5+), GCP(2+)**
IaC	**Terraform(5+), CloudFormation(2)**
Containerization	**Docker(4+), Kubernetes(5+), Helm(4+)**
CI/CD	**GitlabCi(5), Jenkins(4), GitHub Actions(3+), Circleci(2+)**
Monitoring	**CloudWatch(3), Prometheus/Grafana(3+), Datadog(3+)**
GitOps	**ArgoCD(3+), Gitlab Agent(2+)**
Scripting	**Python(2+), Bash(2)**
Configuration Management	**Ansible (3)**
Git	**GitLab(5), GitHub(5+), Bitbucket(2), AWS Code Commit**
OS	**Ubuntu, RHEL, CentOS, Amazon Linux(5)**

Certifications

Projects:

1. Create a new environment in EKS and automate the deploying of microservice applications

Technologies: AWS EKS, GitlabCi Pipeline, Terraform, GitOps, Kubernetes, Gitlab Agent, EC2, ELB

Description:

Designed and developed a private, flexible Terraform module for AWS EKS (Elastic Kubernetes Service) to build a highly scalable Production Environment. Utilized this custom module to automate the deployment of microservice applications through a custom GitlabCi Pipeline template, ensuring streamlined and automated CI/CD processes. Implemented a GitOps workflow by installing Gitlab Agent within the EKS cluster, facilitating seamless and automated deployments.

Responsibilities:

- Led the end-to-end implementation of the highly scalable Production Environment in AWS EKS, taking ownership of the custom Terraform module development.
- Designed and developed a private, flexible Terraform module specifically tailored for AWS EKS, incorporating best practices and infrastructure-as-code principles.
- Wrote a custom GitlabCi Pipeline template to deploy the Terraform module, enabling efficient and consistent CI/CD processes for the Production Environment.
- Implemented GitOps principles by installing Gitlab Agent within the EKS cluster, automating the deployment and management of microservice applications.
- Streamlined the CI/CD pipeline by automating the registration of Gitlab Runner as a Kubernetes Pod,

optimizing scalability and performance.

- Created a comprehensive Readme file documenting the setup, configuration, and maintenance procedures, providing clear instructions for future reference and facilitating knowledge transfer.
- Collaborated closely with cross-functional teams, including developers, testers, and operations, to align project requirements and ensure effective communication throughout the implementation process.
- Proactively identified and implemented optimizations in containerization, monitoring, and infrastructure management, enhancing efficiency and reliability within the Production Environment.
- Successfully delivered the project on time, meeting all key milestones, and exceeding performance expectations, providing a highly scalable and automated Production Environment in AWS EKS.

2. Autoscaling Optimization with Karpenter on AWS EKS

Technologies: AWS EKS, Karpenter, Terraform, Git, Jenkins, Tfscan, AutoScaling, Spot, On-Demand, and Reserved Instances

Description:

Led a strategic initiative to optimize autoscaling capabilities on AWS EKS by migrating from the traditional Cluster Autoscaler to Karpenter. The project aimed to enhance resource efficiency, improve scalability, and leverage Karpenter's advanced features for more effective autoscaling.

Responsibilities:

- Led the end-to-end execution of the autoscaling optimization project, ensuring alignment with business goals and technical requirements.
- Designed and developed a custom Terraform module for deploying Karpenter on AWS EKS clusters, adhering to best practices in infrastructure-as-code.
- Implemented the deployment pipeline using Jenkins, ensuring a streamlined and automated process for provisioning Karpenter across multiple environments.
- Integrated tfscan into the deployment pipeline for regular security and vulnerability scanning of the Terraform code, ensuring a secure and compliant infrastructure.
- Collaborated with cross-functional teams to communicate project objectives, gather feedback, and

ensure a smooth migration process.

- Created comprehensive change requests detailing the migration from Cluster Autoscaler to Karpenter, facilitating the necessary approvals and compliance with company policies.
- Generated detailed documentation for the deployment process, ensuring clear instructions and knowledge transfer for future reference.
- Conducted extensive testing on different environments to validate the effectiveness and compatibility of Karpenter, ensuring a smooth transition to production.
- Analyzed and presented cost optimization strategies, leading to a significant 30% reduction in infrastructure costs through the adoption of Karpenter.
- Proactively identified areas for continuous improvement in resource management and autoscaling, contributing to ongoing efficiency enhancements.

3. Automated IAM Access Key Rotation Monitoring and Notification

Technologies: AWS Lambda, Boto3, CloudFormation, SNS

Description:
"Led the development of an automated solution leveraging AWS Lambda, Boto3, and CloudFormation to monitor IAM access key rotations across our organization's AWS accounts. The solution efficiently identifies and notifies IAM users about expiring access keys, ensuring enhanced security and compliance.

Utilizing Boto3, the Lambda function systematically checks all organization accounts for access keys nearing expiration, achieving a 20% reduction in overdue access key rotations through timely alerts.

Integrated with AWS Simple Notification Service (SNS), the solution provides seamless notifications, empowering users to promptly rotate access keys. The CloudFormation template streamlines deployment across multiple AWS accounts, reducing deployment time by 30% and ensuring consistent, scalable implementations.

To maintain proactive monitoring, a CloudFormation stack orchestrates a scheduled CloudWatch Events rule triggering the Lambda function at month-end. This ensures 99.9% accuracy in scheduled access key rotation monitoring, contributing to a well-maintained security posture across the organization."

Responsibilities:

- Developed a Boto3-powered Lambda function for efficient organization-wide IAM access key rotation monitoring.
- Integrated AWS SNS for timely notifications, achieving a 20% reduction in overdue access key rotations through precise alerts.
- Created a CloudFormation template for uniform deployment, reducing deployment time by 30% across multiple AWS accounts.
- Orchestrated a CloudWatch Events rule for scheduled Lambda execution, ensuring 99.9% accuracy in monitoring access key rotations.
- Contributed to enhanced security and compliance by automating and optimizing IAM access key rotation processes.

4. Implementation of Scalable and Resilient Infrastructure for Deploying a Fully Serverless Web Application

Technologies: AWS Services (S3, DynamoDB, Lambda, API Gateway, Route 53, CloudFront, ACM)

Description:
Led the end-to-end implementation of a scalable and resilient infrastructure for deploying a fully serverless web application on AWS. The project aimed to provide a seamless user experience while securely storing user data in DynamoDB.

The web application consisted of a three-tier architecture, comprising a frontend, backend API, and database. The frontend was built as a static website hosted in an S3 bucket, allowing users to interact with the application's user interface. The backend API, developed using AWS Lambda and API Gateway, processed incoming requests, validated user data, and stored it securely in DynamoDB.

To ensure high availability and low-latency content delivery, CloudFront was configured as a content delivery network (CDN), caching and serving the static website's assets from edge locations around the world. SSL certificates from ACM were integrated with CloudFront to secure communication between users and the application.

The deployment process followed a robust CI/CD pipeline utilizing CircleCI. Changes to the Lambda functions and the S3 website were continuously deployed and tested, ensuring that new features and improvements were delivered seamlessly. Throughout the project, a strong focus was placed on monitoring the Lambda functions using CloudWatch, enabling proactive error detection and implementing alerting mechanisms for prompt issue resolution.

Responsibilities:

- Designed and implemented the deployment of the web application by deploying the static website to an S3 bucket. Ensured efficient content delivery and high availability.
- Created and managed a REST API using API Gateway, enabling seamless integration with the backend services and allowing users to interact with the application securely.
- Architected and developed the backend functionality by writing Python code in Lambda. This code was responsible for processing user data and persisting it in DynamoDB, ensuring efficient and reliable data storage.
- Configured CloudFront to provide low-latency content delivery and improved performance for the web application, enabling a faster and more responsive user experience.
- Integrated an SSL certificate with CloudFront, ensuring secure and encrypted communication between users and the application.
- Implemented a robust CI/CD pipeline using CircleCI, enabling continuous updates and deployments of Lambda functions and the S3 website. This ensured the smooth delivery of new features and improvements.
- Monitored Lambda functions using CloudWatch, proactively detecting errors and implementing alerting mechanisms to promptly address any issues. This enabled efficient troubleshooting and ensured a seamless user experience.
- Implemented caching mechanisms in CloudFront to optimize content delivery and improve application performance.
- Implemented Route 53 for DNS management and set up custom domain routing to provide a user-friendly and branded experience.
- Utilized ACM (AWS Certificate Manager) for managing SSL certificates, ensuring secure and trusted communication.
- Leveraged CloudFormation for infrastructure-as-code, enabling consistent and repeatable deployments.
- Implemented centralized logging and monitoring solutions using AWS CloudTrail and CloudWatch,

providing visibility into system behavior and performance.
- Conducted load testing and performance optimization to ensure the infrastructure could handle high traffic and maintain optimal performance.

5. Deploying Applications in Kubernetes Cluster with Continuous Deployment using ArgoCD

Technologies: Kubernetes, Helm, GKE, EKS, Docker, GitOps, ECR, GCR, ArgoCD, Prometheus, Grafana, Secret Manager, IAM

Description:
Implemented deployment of applications in Kubernetes clusters (GCP GKE and AWS EKS) leveraging Helm for package management and Docker for containerization. Utilized GitOps principles to manage deployment configurations and version control. Implemented Continuous Deployment (CD) pipeline using ArgoCD for automated application deployment from Git repositories. Monitored Kubernetes clusters and objects using Prometheus for metrics collection and Grafana for visualization, enabling proactive troubleshooting and optimization.

Responsibilities:

- Created and configured ArgoCD applications to deploy services from Git repositories, ensuring automated and consistent deployments.
- Deployed highly available applications on GKE and EKS clusters, utilizing Kubernetes features like rolling updates, automatic scaling, and self-healing to enhance reliability.
- Utilized Kubernetes resources such as Services, Deployments, StatefulSets, Role-Based Access Control (RBAC), Persistent Volumes (PV), Persistent Volume Claims (PVCs), Pods, and Namespaces.
- Implemented monitoring solutions using Prometheus and Grafana to monitor Kubernetes clusters, built custom dashboards, created alerts, and integrated Alertmanager with Slack for real-time notifications.
- Leveraged Secret Manager services provided by the cloud providers (e.g., AWS Secrets Manager, Google Secret Manager) for secure storage and management of sensitive information.

6. DevSecOps Implementation with Automated Vulnerability Scanning

Technologies: GitLab, Wiz, SonarQube, Veracode SAST, Bash/Shell scripting, API Integration

Description:
Implemented a robust DevSecOps framework leveraging GitLab CI/CD pipelines integrated with security tools like Wiz, SonarQube, and Veracode SAST. The project focused on automating vulnerability scanning throughout the development lifecycle, ensuring prompt identification and resolution of security issues. By halting build and deployment processes upon detecting high or very high-severity vulnerabilities, the project aimed to maintain the integrity and security of the software being developed. Additionally, the automated creation of GitLab issues based on scanning reports facilitated efficient tracking and resolution of security issues.

Responsibilities:

- Designed and developed flexible GitLab CI/CD templates for seamless integration of security tools into the development workflow.
- Implemented automated vulnerability scanning stages within GitLab pipelines, halting builds and deployments upon detecting critical vulnerabilities.
- Configured GitLab pipelines to receive and process vulnerability reports from security tools like Wiz, SonarQube, and Veracode SAST.
- Developed custom Bash/Shell scripts to automate various tasks within GitLab pipelines, enhancing efficiency and reliability.
- Integrated APIs to establish communication channels between GitLab and external security tools, facilitating seamless integration and data exchange.
- Collaborated with the team to promote DevSecOps best practices and ensure adherence to security policies throughout the development process.
- Provided guidance and support to team members on using GitLab CI/CD and security tools effectively for secure software development.

7. Automating Manual Jobs in AWS using Python module boto3 and Bash

Technologies: Python boto3, AWS, Lambda, Eventbridge, RDS, S3

Description:

Led the successful implementation of a comprehensive automation solution for AWS infrastructure using Python module boto3 and Bash scripting. This initiative aimed to replace error-prone and time-consuming manual tasks with efficient and reliable automated processes. The project involved designing and developing robust workflows to streamline infrastructure changes, resulting in significant improvements in productivity, scalability, and operational stability.

Responsibilities:

- Oversaw the management of EC2 instance roles and policies, ensuring adherence to security best practices and proper access control.
- Implemented standardized approaches for provisioning, monitoring, and decommissioning instances, resulting in optimized resource utilization.
- Developed and deployed Lambda functions integrated with Eventbridge to enforce strict tagging policies for all resources and users.
- Implemented a comprehensive tagging framework, preventing the creation of resources and users without the necessary tags, improving resource tracking, and facilitating accurate cost allocation.
- Conducted regular audits to verify the security of S3 buckets and proactively address any potential vulnerabilities.
- Implemented robust access controls and encryption mechanisms, ensuring the confidentiality, integrity, and availability of data stored in S3.
- Created highly resilient and efficient Bash scripts for both RDS Cluster and NonCluster disaster recovery scenarios.
- Conducted regular testing and continuous improvement to minimize recovery time objectives (RTO) and recovery point objectives (RPO) in the event of failures.
- Developed Python scripts for filtering and deleting unused security groups, reducing the attack surface and enhancing network security.
- Implemented automated processes to update the specific source IP across all security groups, maintaining stringent access controls while accommodating dynamic IP requirements.
- Designed and implemented a robust Bash script to enable password-based authentication on EC2 instances, catering to specific use cases and enhancing flexibility in authentication mechanisms.

8. Advanced Monitoring and Alerting using Prometheus and Grafana

Technologies: Prometheus, Grafana, Metrics, Alerts, Kustomize, Kube-Prometheus-Stack, Helm, K8s, Vault, AWS Load Balancer

Description:

Led the implementation of an advanced monitoring and alerting system using Prometheus and Grafana. The project aimed to enhance the visibility and troubleshooting capabilities of the application and infrastructure by collecting comprehensive metrics and quickly detecting and resolving issues. By leveraging the power of Prometheus and Grafana, the project delivered real-time insights, proactive alerting, and intuitive visualizations, significantly improving the overall stability and performance of the system.

Responsibilities:

- Installed and configured Prometheus to collect metrics from various sources, including the application and infrastructure components.
- Designed and implemented scalable and efficient Prometheus deployments, ensuring reliable metric collection and storage.
- Developed and configured Prometheus exporters to capture custom metrics specific to the application and integrate with third-party services.
- Expanded the monitoring capabilities by collecting metrics from diverse sources, enabling comprehensive performance analysis.
- Utilized Prometheus's alerting capabilities to create well-defined alerts for critical metrics, ensuring timely notifications for potential issues.
- Configured and fine-tuned alert rules to minimize false positives and prioritize actionable alerts.
- Integrated Prometheus with Grafana, enabling seamless data visualization and correlation of metrics with intuitive dashboards.
- Leveraged Grafana's rich feature set to enhance the monitoring experience, facilitating in-depth analysis and troubleshooting.
- Designed and created visually appealing and informative Grafana dashboards to display metrics, alerts, and system performance in real time.
- Collaborated with relevant teams to ensure the dashboards met their specific monitoring and reporting requirements.
- Leveraged Grafana's alerting capabilities to create customized alert rules, enabling proactive

identification and resolution of critical issues.
- Configured notifications through various channels, including email, Slack, and other collaboration tools, ensuring prompt responses to incidents.

9. Upgrading EKS and GKE Clusters

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.

10. Infrastructure Automation using Ansible

Technologies: Ansible, YAML, Bash, AWS, Roles, Playbook, IAM, Secret Manager, Webhooks

Description:

Led the implementation of infrastructure automation using Ansible to streamline and standardize deployment processes, improve scalability, and enhance operational efficiency. The project focused on managing configuration drift, reducing manual efforts, and ensuring consistent and reproducible deployments across multiple environments.

Responsibilities:

- Designed and developed Ansible playbooks using YAML syntax to automate the provisioning and configuration of infrastructure components.
- Implemented best practices for creating reusable and modular playbooks, enabling flexible and scalable automation.
- Orchestrated complex deployments and configurations of various resources, including virtual machines, load balancers, databases, and networking components.

- Utilized Ansible's rich library of modules to interact with different infrastructure providers, such as AWS, Azure, and VMware.
- Managed and maintained the desired state of infrastructure configurations using Ansible's declarative approach.
- Implemented role-based configurations to enforce consistency and eliminate configuration drift across multiple servers and environments.
- Integrated Ansible with CI/CD pipelines to automate the deployment and configuration management processes.
- Collaborated with development teams to ensure smooth integration of Ansible playbooks into the overall software delivery lifecycle.
- Developed Ansible playbooks to perform routine system checks, health monitoring, and maintenance tasks.
- Implemented automated processes for log collection, performance monitoring, and system updates, ensuring proactive maintenance and minimizing downtime.
- Integrated security best practices into Ansible playbooks, including access controls, firewall rules, and encryption configurations.
- Ensured compliance with industry standards and regulatory requirements by implementing automated security audits and remediation tasks.

11. Continuous Integration and Deployment using Azure DevOps Pipeline and AKS Cluster

Technologies: Azure DevOps, Azure Kubernetes Service (AKS), Docker, Helm, PIpeline

Description:
Led the implementation of a robust Continuous Integration and Deployment (CI/CD) pipeline using Azure DevOps and AKS cluster to streamline the application release process, improve scalability, and ensure consistent and reliable deployments. The project focused on automating build, test, and deployment workflows, including containerization of applications, managing Helm charts, and deploying them to the AKS cluster. Additionally, artifacts were generated and securely stored to facilitate version control and efficient software distribution.

Responsibilities:

- Designed and configured a CI/CD pipeline using Azure DevOps to automate the build, test, and deployment processes.
- Defined pipeline stages and tasks for code compilation, testing, artifact generation, containerization, and deployment to the AKS cluster.
- Containerized applications using Docker, creating lightweight and portable artifacts for deployment.
- Managed Helm charts to define, package, and version the Kubernetes deployments, ensuring consistency and reproducibility across environments.
- Provisioned and configured an AKS cluster, leveraging Azure Kubernetes Service to manage containerized applications efficiently.
- Defined infrastructure as code (IaC) using tools like Azure Resource Manager (ARM) templates or Terraform to automate cluster provisioning and configuration.
- Implemented automated build and test processes within the CI/CD pipeline to ensure code quality and adherence to defined standards.
- Integrated unit tests, integration tests, and code analysis tools to enable continuous integration of code changes.
- Orchestrated the deployment of applications to the AKS cluster using Azure DevOps release pipelines.
- Configured release gates and approvals to ensure proper validation and control over the deployment process.
- Generated artifacts, such as Docker images, Helm charts, or packaged application bundles, as part of the CI/CD pipeline.
- Securely stored artifacts in an artifact repository, such as Azure Container Registry (ACR), for version control and efficient distribution.

12. Docker Image Optimization using Multi-Stage Build and Best Practices

Technologies: Docker, Dockerfile, Docker Build, Docker Cache, Multi-stage build, Docker Compose, Volumes

Description:

Led the optimization efforts to reduce the build time of the application Docker image, which was initially taking more than 1.5 hours. Implemented a multi-stage Docker build approach and applied various best

practices to streamline the build process and improve efficiency. By optimizing Docker commands, reducing unnecessary layers, and leveraging the Docker cache effectively, the build time was significantly reduced to only 7 minutes, resulting in faster application delivery and improved development productivity.

Responsibilities:

- Reviewed and analyzed the existing Dockerfile to identify inefficiencies and areas for optimization.
- Reorganized and consolidated Docker commands, eliminating redundant or unnecessary steps to streamline the build process.
- Implemented a multi-stage Docker build approach to separate the build environment from the runtime environment.
- Utilized separate build stages to compile dependencies, build the application, and package it into a lightweight runtime image.
- Leveraged Docker cache effectively to reduce build times by reusing previously built layers for unchanged dependencies or source code.
- Optimized the order of commands in the Dockerfile to maximize cache utilization and minimize unnecessary rebuilds.
- Employed techniques to reduce the final image size, such as removing unnecessary files, dependencies, or build artifacts.
- Utilized lightweight base images and minimized the inclusion of unnecessary packages or libraries.
- Leveraged package managers or build tools' caching mechanisms to speed up dependency installation during the build process.
- Utilized techniques like copying dependency manifest separately and only running dependency installation steps when needed.
- Adhered to Docker best practices, such as using explicit version tags for base images and dependencies.
- Employed security best practices, including scanning images for vulnerabilities and ensuring only trusted sources were used.

13. Terraform Optimization and Cost Management with Terragrunt

Technologies: Terragrunt, Terraform, Lambda Functions, ECS, Route 53, Cost Optimization, Terraform Modules, Code Scanning

Description:

Contributed to an ongoing project focused on optimizing Terraform configurations and managing costs using Terragrunt. The primary goal was to enhance the efficiency and cost-effectiveness of AWS infrastructure provisioning while maintaining scalability and reliability. Responsibilities included analyzing existing Terraform modules, identifying opportunities for optimization, and implementing enhancements to improve resource utilization and cost efficiency. Additionally, developed flexible modules for provisioning Lambda functions, ECS clusters, and Route 53 configurations, ensuring reusability and flexibility across different projects and environments.

Responsibilities:

- Analyzed existing Terraform configurations to identify inefficient resource allocations and opportunities for cost optimization.
- Implemented cleanup routines using Terragrunt to remove redundant or unused resources, reducing infrastructure costs and improving resource management.
- Wrote flexible Terraform modules for provisioning Lambda functions, ECS clusters, and Route 53 configurations, enabling standardized and reusable infrastructure components across projects.
- Optimized existing Terraform modules to enhance dynamism and flexibility, allowing for dynamic configuration changes based on environment variables or external inputs.
- Collaborated with the DevOps team to review and refine Terraform configurations, ensuring alignment with best practices and cost optimization strategies.
- Documented optimization techniques and best practices for Terraform and Terragrunt usage, providing guidance to team members on efficient infrastructure management.
- Conducted cost analysis and monitoring of AWS resources to identify cost-saving opportunities and recommend optimizations to minimize infrastructure expenses.
- Provided support and assistance to team members on Terraform and Terragrunt usage, troubleshooting issues, and implementing solutions to optimize infrastructure provisioning and management.