RESUME POINTS FOR AWS AND DEVOPS

------------------------------AWS Three-Tier Application Development--

--->Developed and Deployed a Scalable Three-Tier Architecture on AWS

--->Architected a robust three-tier application leveraging AWS services, including EC2, RDS, and Elastic Load Balancing (ELB), to ensure high availability and scalability

--->Utilized AWS CloudFormation/Terraform to automate the deployment and configuration of AWS resources, improving deployment efficiency and consistency.

--->Optimized application performance by configuring Auto Scaling groups, setting up load balancers, and implementing health checks to ensure seamless operation under varying loads.

--->Implemented AWS security best practices, including VPC configurations, IAM policies, and encryption mechanisms to safeguard application data and infrastructure.

---Route 53--

1.Successfully managed and registered multiple domains using AWS Route 53, ensuring seamless domain name resolution across global services.

2.Implemented DNS management strategies, including configuration of hosted zones, and creation of A, AAAA, CNAME records to support various application and infrastructure requirements.

3.Implememted Hosted zones for both public and privated hosted zones for bothe internal and external routing purpose.

4.Configured and optimized Route 53 traffic routing policies, including simple, latency-based, failover, and geolocation routing to ensure high availability and performance.

5.Implemented health checks and failover mechanisms to monitor and automatically route traffic to healthy endpoints, ensuring continuous service availability for across region High Avaiablity.

---EC2--

1.Provisioned and managed AWS EC2 instances for scalable application hosting, configuring instance types, storage options, and networking settings to meet performance and availability requirements.

2.Implemented Auto Scaling groups to dynamically adjust EC2 instance count based on traffic and load, ensuring high availability and cost-efficiency.

3.Designed and configured EC2 security groups and network ACLs to enforce access control and protect instances from unauthorized access while maintaining compliance with security policies.

4.Utilized EC2 Elastic Load Balancers (ELB) to distribute incoming traffic across multiple instances, improving application reliability and reducing latency.

5.Configured EC2 instances with Amazon EBS volumes for persistent storage, managing snapshots and volume backups to ensure data durability and recovery

6. Configured AMI to take backup of application and for launch template reference to give provision for Autoscling group.

--- VPC ---

1.Designed, deployed, and managed Amazon Virtual Private Clouds (VPCs) with custom subnets, route tables, and network ACLs to meet specific application and security requirements.

2.Configured security groups and network ACLs to enforce fine-grained network security controls, ensuring secure access to resources within AWS environments.

3.Deployed and configured Elastic Load Balancers (Application, Network) to distribute incoming application traffic across multiple EC2 instances, improving application availability and fault tolerance.

4.Automated network infrastructure deployment and management using Terraform, and AWS CLI, ensuring consistency, repeatability, and reduced manual intervention.

5.Integrated AWS networking configurations with CI/CD pipelines to streamline environment provisioning and updates

6.Developed and implemented disaster recovery strategies for network infrastructure, including setting up cross-region VPCs and Route 53 failover policies to ensure business continuity

7.Architected and deployed custom VPCs with multiple subnets (public, private, and isolated) across multiple Availability Zones (AZs) to ensure high availability and fault tolerance.

8.Strategically segmented network traffic within the VPC by creating dedicated subnets for different application tiers (e.g., web, application, database) to enhance security and control.

9.Implemented and configured Internet Gateways (IGWs) to enable internet access for resources in public subnets, and Configured route tables for private subnets to route traffic through NAT gateways for secure internet access, ensuring instances in private subnets remain isolated from direct internet exposure.

---RDS---

1.Provisioned and managed Amazon RDS instances for various database engines, including MySQL, PostgreSQL, Oracle, SQL Server, and MariaDB, tailored to meet specific application requirements.

2.Configured RDS instances with custom parameter groups, option groups, and database settings to optimize performance, security, and compatibility with application workloads

3.Implemented Multi-AZ deployments for RDS to ensure high availability and automatic failover in case of infrastructure failures, minimizing downtime and ensuring business continuity.

4.Set up automated backups, snapshots, and point-in-time recovery to protect data and enable quick recovery from data loss or corruption.

5.Utilized read replicas to offload read traffic and scale read-heavy applications, improving database responsiveness and reducing the load on primary instances.

6.Configured RDS instances within VPCs, using security groups, network ACLs, and subnet groups to control and secure database access.

7.Optimized RDS costs by selecting the appropriate instance types, storage types, and purchasing Reserved Instances based on usage patterns and performance needs.

8.Set up cross-region read replicas to enable data replication and disaster recovery across AWS regions, providing geographic redundancy and improving application availability.

9.Configured RDS global databases for low-latency global read and disaster recovery solutions, ensuring data availability across multiple regions.

------------------------------------=--------------Cloud Front--

1.Configured CloudFront to distribute content across multiple AWS regions and edge locations, ensuring high availability and low latency for users across the globe.

2.Leveraged CloudFront's integration with AWS Global Accelerator to improve application performance for geographically dispersed users, providing consistent and fast user experiences.

3.Seamlessly integrated CloudFront with AWS services such as S3, EC2, and Elastic Load Balancing (ELB) to deliver dynamic and static content efficiently.

--AutoScaling---

1.Designed and configured Auto Scaling Groups (ASGs) to automatically scale Amazon EC2 instances based on predefined policies and metrics, ensuring optimal resource utilization and cost-efficiency.

2.Implemented multiple scaling policies, including dynamic scaling based on CloudWatch alarms, scheduled scaling for predictable workloads, and step scaling to handle varying load conditions.

3.Configured target tracking scaling policies to maintain performance and availability by automatically adjusting the number of instances to meet specified target utilization levels.

4.Integrated Auto Scaling Groups with Elastic Load Balancers (ELBs) to ensure even distribution of incoming traffic across instances, improving application availability and performance.

5.Configured health checks within Auto Scaling and ELBs to automatically replace unhealthy instances and maintain application reliability.

6.Designed Auto Scaling configurations to support disaster recovery and high availability strategies, including multi-AZ deployments and cross-region replication to ensure continuous application availability.

---s3--

1.Configured and managed Amazon S3 buckets for scalable object storage, ensuring efficient data management and high availability for application assets.

2.Implemented S3 versioning to keep track of object changes and maintain historical versions, enhancing data integrity and recovery options.

3.Applied S3 bucket policies and IAM roles to enforce granular access control, securing sensitive data and maintaining compliance with organizational security standards.

4.Utilized S3 lifecycle policies to automate the transition of objects between storage classes, optimizing storage costs and managing data retention effectively.

5.Implemented cross-region replication (CRR) to enhance data durability and availability, ensuring redundancy across multiple AWS regions.

6.Configured and managed static website hosting for s3 bucket.

--IAM ---

1.Designed and implemented IAM policies and roles to manage user and application permissions, ensuring secure and least-privilege access across AWS resources.

2.Configured IAM roles for AWS services to enable secure cross-service communication and automate access management in a scalable environment.

3.Conducted IAM policy reviews and audits to ensure compliance with organizational security standards and regulatory requirements, enhancing overall security posture.

---Terraform ---

1.Developed and managed infrastructure as code (IaC) using Terraform, automating the provisioning and management of AWS resources and ensuring consistent and repeatable deployments.

2.Designed and implemented a three-tier application architecture using Terraform, orchestrating the provisioning of web, application, and database layers on AWS.

3.Designed and implemented Terraform modules to encapsulate reusable infrastructure components, improving modularity and maintainability of code.

4.Integrated Terraform with CI/CD pipelines to automate infrastructure deployments and updates, enhancing development workflows and accelerating delivery cycles.

5.Utilized Terraform State Management to track infrastructure changes and ensure consistency across environments, preventing configuration drift.

6.Applied Terraform workspaces and variable files to manage multiple environments (e.g., development, staging, production) efficiently and maintain environment-specific configurations.

7.Developed reusable Terraform modules for each tier of the application (web servers, application servers, and databases), ensuring consistent and scalable deployments.

8. Configured Terraform statefile dissater configured with help of s3 replication across region.

9.Provisioned and managed AWS RDS databases using Terraform, automating database instance creation, backups, and scaling to support the application layer.

10.Implemented security best practices with Terraform by configuring AWS Security Groups, IAM roles, and policies to restrict access and secure communication between application tiers.

11.Integrated AWS Elastic Load Balancers (ELB) with Terraform to distribute traffic across web and application servers, improving load management and fault tolerance.

12.Configured Terraform state management with remote backends such as AWS S3 and DynamoDB, maintaining infrastructure state consistency and enabling collaborative management.

Terraform Infrastructure Automation

✅ Designed, developed, and maintained Terraform scripts to provision infrastructure across AWS ensuring consistency and repeatability.

✅ Implemented Infrastructure as Code (IaC) using Terraform modules to provision cloud resources, such as EC2 instances, VPCs, S3 buckets, and RDS instances.

✅ Utilized Terraform workspaces to manage multiple environments (e.g., dev, staging, and production) within the same configuration.

✅ Implemented state management using remote backends like S3,, or Terraform Cloud to manage and version control Terraform state files.

Cloud Infrastructure Management with Terraform

✅ Automated cloud resource provisioning for AWS using Terraform, including EC2 instances, VPCs, subnets, IAM roles, security groups, and S3 buckets.

✅ Optimized cloud infrastructure by using Terraform auto-scaling configurations and configuring load balancers for high availability.

CI/CD Integration with Terraform

✅ Integrated Terraform with Jenkins, GitLab CI, and GitHub Actions to automate infrastructure provisioning in continuous deployment pipelines.

✅ Configured Terraform with CI/CD pipelines to automatically trigger infrastructure changes based on version control updates.

✅ Developed custom CI/CD workflows to deploy infrastructure updates while ensuring zero-downtime for applications.

Version Control & Collaboration

✅ Used Git for version control of Terraform configurations, ensuring consistency and collaboration across teams.

✅ Collaborated with development teams to align infrastructure deployments with application requirements, following GitOps principles.

Security & Compliance Automation

✅ Automated security groups, IAM roles, and permissions management using Terraform to enforce security best practices and reduce manual configuration errors.

✅ Integrated Terraform with third-party tools (e.g., Vault, AWS KMS) to securely manage secrets and access policies.

✅ Created compliance rules to ensure that all infrastructure is provisioned according to company security standards.

Terraform State Management & Collaboration

✅ Managed remote state using Terraform Cloud, S3, and DynamoDB for state locking and consistency in collaborative teams.

✅ Used Terraform workspaces for managing different deployment environments (e.g., development, staging, production).

✅ Ensured Terraform state file security by encrypting it at rest and restricting access using IAM policies.

Cost Optimization & Monitoring

✅ Leveraged Terraform cost estimation tools to predict cloud resource costs before provisioning.

✅ Integrated Terraform with cost management tools to analyze and optimize the spending for cloud resources (e.g., EC2 instance types, storage costs).

--JENKINS--

1. Designed and implemented CI/CD pipelines using Jenkins, reducing deployment time by 50%.
2. Developed Jenkins Pipelines (Declarative & Scripted) for automated build, test, and deployment processes.
3. Integrated Jenkins with Git, Docker, Kubernetes, and AWS for seamless deployment.
4. Automated infrastructure provisioning using Terraform and Ansible.
5. Optimized build and release processes, improving efficiency by 40%.
6. Managed Jenkins plugins and system security, ensuring compliance with best practices.
7. Monitored Jenkins job performance and troubleshot pipeline failures.

CI/CD Pipeline Development & Automation

✅ Designed and implemented CI/CD pipelines using Jenkins, reducing deployment time by 50% and improving release frequency.

✅ Developed Declarative & Scripted Pipelines in Jenkins to automate build, test, and deployment stages.

✅ Configured Jenkins to automatically trigger builds via GitHub Webhooks, SCM Polling, and API Calls.

✅ Automated rollback mechanisms in Jenkins to quickly revert to a stable release in case of deployment failure.

✅ Integrated Jenkins pipelines with Docker & Kubernetes for containerized application deployment.

✅ Reduced manual intervention in deployments by implementing parameterized builds and approval gates.

Jenkins Integration with Tools & Technologies

✅ Integrated Jenkins with GitHub, GitLab ensuring smooth version control and automated builds.

✅ Configured Maven builds in Jenkins for Java applications.

✅ Set up Jenkins Agents/Slaves on Linux machines for distributed build execution.

✅ Deployed and managed Jenkins on Kubernetes clusters, improving scalability and performance.

✅ Integrated SonarQube with Jenkins for automated code quality and security analysis.

✅ Configured Jenkins pipelines to deploy applications to AWS (EC2, S3, Lambda)

✅ Integrated Terraform and Ansible into Jenkins pipelines for Infrastructure as Code (IaC) automation.

Jenkins Plugin Management & Administration

✅ Managed and configured Jenkins plugins for source control, build tools, notifications, and deployment automation.

✅ Optimized Jenkins by managing executor limits, retention policies, and concurrent builds to improve performance.

✅ Set up Role-Based Access Control (RBAC) in Jenkins to enforce security best practices.

✅ Enabled Blue Ocean UI for a better visual representation of Jenkins pipelines.

Performance Optimization & Troubleshooting

✅ Reduced build failures by 30% by implementing robust error-handling mechanisms in Jenkins pipelines.

✅ Troubleshot and resolved Jenkins job failures by analyzing build logs, system logs, and agent connectivity issues.

✅ Optimized Jenkins performance by tuning Garbage Collection (GC), increasing heap memory, and using external databases for job history storage.

✅ Automated log rotation and backup strategies for Jenkins, ensuring system stability.

--GIT--

Git Repository Management & Workflow Optimization

✅ Managed and maintained Git repositories on GitHub, GitLab, and Bitbucket for code versioning.

✅ Designed and enforced Git branching strategies (Git Flow, Feature Branching) to improve team collaboration.

✅ Migrated legacy repositories to Git, ensuring history retention and minimal downtime.

Branching, Merging & Conflict Resolution

✅ Established structured branching models (e.g., Git Flow, Trunk-Based Development) to improve release management.

✅ Resolved complex merge conflicts and improved collaboration by implementing rebase and cherry-pick workflows.

Git CI/CD & Automation

✅ Integrated Git with Jenkins, GitHub Actions, and GitLab CI/CD for automated build and deployment.

✅ Configured webhooks to trigger builds and deployments based on Git events (push, pull request).

Security & Access Control

✅ Managed SSH keys and access permissions for private Git repositories.

✅ Configured protected branches and enforced code reviews to maintain high code quality.

✅ Audited Git repositories for security vulnerabilities and compliance with best practices.

---DOCKER---

Docker Containerization & Image Management

✅ Built and optimized Docker images using Dockerfile best practices to reduce image size by 40%.

✅ Managed and maintained Docker Hub, Amazon ECR, and private Docker registries.

✅ Reduced build time by 30% using multi-stage Docker builds.

✅ Automated container image vulnerability scanning using tools like Trivy.

Docker Compose & Multi-Container Applications

✅ Designed and deployed multi-container applications using Docker Compose for efficient local development.

✅ Configured volumes & bind mounts to persist data across container restarts.

✅ Optimized Docker networking (bridge, host, overlay) to improve inter-container communication.

CI/CD Pipeline Integration with Docker

✅ Integrated Docker with Jenkins, GitHub Actions, GitLab CI/CD, for automated builds and deployments.

✅ Deployed containerized applications to Kubernetes clusters using Docker images.

---MAVEN---

Maven Build & Dependency Management

✅ Automated Java application builds using Maven, reducing manual intervention by 80%.

✅ Managed dependencies & transitive dependencies in Maven to avoid conflicts and optimize package size.

✅ Configured Maven profiles to support different environments (development, testing, production).

✅ Implemented Maven dependency exclusions and scope management to prevent version conflicts.

Maven Lifecycle, Plugins & Customization

✅ Optimized build performance by configuring Maven clean, compile, test, package, verify, install, and deploy phases.

✅ Created custom Maven goals to automate repetitive development tasks.

✅ Used Maven Wrapper (mvnw) for project portability and consistency across different systems.

CI/CD & Artifact Repository Management

✅ Integrated Maven with Jenkins, GitHub Actions, and GitLab CI/CD for automated builds and deployments.

✅ Managed artifact repositories using jfrog & Artifactory, optimizing artifact storage and retrieval.

✅ Published and versioned Maven artifacts for internal and external package sharing.

✅ Enforced Maven repository security policies to restrict unauthorized access to internal artifacts.

Performance Optimization & Troubleshooting

✅ Reduced build times by 30% by optimizing the Maven dependency resolution strategy.

✅ Debugged and resolved Maven build failures using mvn -X (debug mode) and mvn dependency:tree.

---SONARQUBE---

SonarQube Setup & Code Analysis

✅ Configured and maintained SonarQube servers for static code analysis, bug detection, and security auditing.

✅ Implemented SonarQube Quality Gates to enforce minimum standards for code quality before deployment.

✅ Managed SonarQube rules, quality profiles, and quality gates to align with industry best practices.

✅ Performed branch analysis and pull request validation using SonarQube.

SonarQube Integration with CI/CD & Build Tools

✅ Integrated SonarQube with Jenkins, GitHub Actions, and GitLab CI/CD to enable automated code scanning.

✅ Configured SonarQube scanners for Maven to analyze Java, JavaScript, and Python applications.

✅ Automated security scans using SonarQube, OWASP Dependency Check, and SAST tools to detect vulnerabilities.

✅ Set up SonarQube webhooks for real-time notifications on code quality issues.

Security & Compliance

✅ Enforced compliance with OWASP Top 10 and CWE security standards through SonarQube rulesets.

✅ Improved code maintainability by reducing technical debt through automated SonarQube reports.

--KUBERNETES---

Kubernetes Cluster Management & Deployment

✅ Designed, deployed, and maintained highly available Kubernetes clusters on AWS EKS.

✅ Deployed and managed microservices architecture on Kubernetes using Deployments, StatefulSets, and DaemonSets.

✅ Automated cluster provisioning using Terraform.

✅ Configured Kubernetes Ingress Controllers (NGINX) for routing and load balancing.

Networking & Security

✅ Configured Kubernetes networking using ClusterIP, NodePort, LoadBalancer, and Ingress Controllers.

✅ Enforced Kubernetes Network Policies to restrict pod-to-pod communication and improve security.

✅ Managed Role-Based Access Control (RBAC) to implement least privilege access for Kubernetes resources.

Storage & Persistent Data Management

✅ Configured Persistent Volumes (PV), Persistent Volume Claims (PVC), and Storage Classes for data persistence.

✅ Integrated Container Storage Interface (CSI) for dynamic storage provisioning.

✅ Deployed stateful applications like databases (MySQL, PostgreSQL, MongoDB, Elasticsearch) on Kubernetes.

Kubernetes Monitoring & Logging

✅ Set up Prometheus & Grafana to monitor cluster health, node metrics, and application performance.

✅ Configured ELK (Elasticsearch, Logstash, Kibana) and Loki for centralized logging.

CI/CD & Automation with Kubernetes

✅ Integrated Jenkins, GitHub Actions, and GitLab CI/CD with Kubernetes for automated deployments.

✅ Implemented ArgoCD for GitOps-based Kubernetes deployments.

✅ Used Helm Charts for packaging and deploying Kubernetes applications efficiently.

✅ Automated blue-green and canary deployments using Kubernetes Deployment Strategies.

Autoscaling & High Availability

✅ Configured Horizontal Pod Autoscaler (HPA) to optimize resources.

✅ Deployed Kubernetes workloads across multiple availability zones to ensure high availability.

Infrastructure as Code (IaC) & Kubernetes Security

✅ Managed Kubernetes infrastructure using Terraform.

✅ Scanned Kubernetes configurations for vulnerabilities using Trivy.

---ARGO CD--

ArgoCD Deployment & GitOps Integration

✅ Implemented GitOps-based deployment pipelines using ArgoCD, enabling automated Kubernetes application deployment.

✅ Integrated Helm Charts & Kustomize with ArgoCD for dynamic and reusable application deployment.

✅ Managed ArgoCD declarative configurations using Git repositories for version-controlled Kubernetes deployments.

CI/CD & Automation

✅ Integrated ArgoCD with Jenkins, GitHub Actions, and GitLab CI/CD for automated software releases.

✅ Automated ArgoCD Sync Policies with Auto-Sync, Manual-Sync, and Sync Waves to control deployment order.

Argo Rollouts for Progressive Deployment

✅ Used Argo Rollouts for blue-green and canary deployments, ensuring zero-downtime releases.

✅ Configured traffic shifting and analysis using Argo Rollouts with NGINX Ingress Controller.

Monitoring & Troubleshooting

✅ Integrated Prometheus & Grafana for ArgoCD metrics monitoring.

--GRAFNA PROMETHEUS --

Prometheus Setup & Configuration

✅ Deployed and managed Prometheus servers for metrics collection, monitoring, and alerting across a Kubernetes infrastructure.

✅ Ensured high availability of Prometheus using Prometheus HA setup with replication.

Grafana Dashboards & Visualizations

✅ Created custom Grafana dashboards for visualizing metrics from Prometheus, Kubernetes, application performance, and infrastructure health.

✅ Designed interactive and user-friendly visualizations to represent complex time-series data using Grafana panels (graphs, tables, and heatmaps).

✅ Integrated Prometheus alerting rules with Grafana alerts, allowing proactive monitoring of system performance and application health.

Kubernetes Integration

✅ Integrated Prometheus with Kubernetes clusters using the Prometheus Operator, simplifying deployment and management of monitoring services.

✅ Monitored Kubernetes resource utilization and performance using Prometheus queries on pods, nodes, and services.

✅ Configured horizontal pod autoscaling in Kubernetes based on Prometheus metrics to scale applications automatically.

--ANSIBLE--

Ansible Configuration Management & Automation

✅ Automated server provisioning, configuration management, and application deployment using Ansible Playbooks.

✅ Developed reusable Ansible roles to standardize infrastructure setups, reducing deployment times.

✅ Managed Linux and Windows environments using Ansible for configuration management and software updates.

✅ Automated system tasks like patch management, configuration auditing, and environment setup across multiple servers.

✅ Integrated Ansible Vault for securely managing sensitive data like passwords, API keys, and encryption keys.

Cloud Automation with Ansible

✅ Automated cloud infrastructure provisioning on AWS using Ansible modules (EC2, S3, RDS, and VPC).

✅ Configured auto-scaling groups in AWS using Ansible to manage elastic environments.

✅ Developed cloud-specific playbooks for seamless deployment and scaling of resources in cloud environments.

Dynamic Inventory Management

✅ Configured dynamic inventories in Ansible to pull real-time data from cloud providers (AWS, Azure, GCP) and manage infrastructure resources.

✅ Used Ansible dynamic inventory scripts to retrieve and manage resources dynamically across various environments.