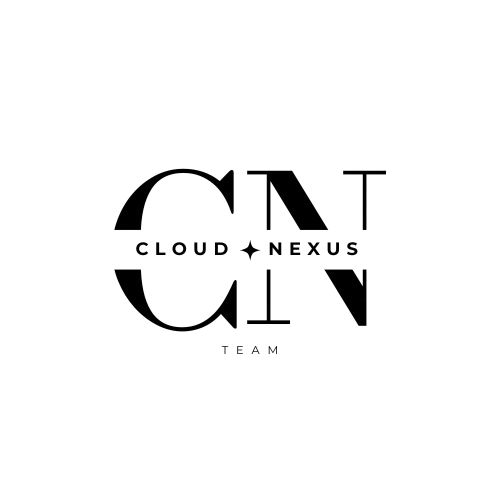


**CAA 900**Deployment Strategy – Draft

E2E & Environments



*Drafted by*,

Group Name : **CLOUD NEXUS**

Date : 13/02/2025

**Executive Summary:**

The Extensive deployment strategy for the Multi-Cloud Resource Insights & Cost Projection Platform (MCRP) provides a detailed outline of architecture design, infrastructure provisioning, CI/CD workflows, automation, security, cloud resource management, and optimization techniques. It ensures scalable, secure, and efficient deployments across development, staging, and production environments hosted on AWS.

**Architecture Design :**

* **Virtual Machines (VMs):**
  + EC2 instances provisioned via *Terraform* with detailed instance types (t3.medium for dev, t3.large for production), auto-scaling groups, and elastic load balancers.
  + AWS Backup integrated for scheduled backups, AMI creation for disaster recovery, and CloudWatch for performance metrics.
* **Docker Containers:**
  + Multi-stage Dockerfiles with Alpine-based images for minimal footprint.
  + Docker Compose for local development and integration testing.
  + Amazon ECR for managing image lifecycles with lifecycle policies for automated pruning.
* **Kubernetes (EKS):**
  + AWS EKS with managed node groups, taints, and tolerations for workload segregation.
  + Helm charts with values.yaml files for environment-specific configurations.
  + Service mesh with Istio for traffic management, observability, and security.
* **Networking:**
  + VPC with CIDR, *public/private* subnets with route tables and NAT gateways.
  + Elastic Load Balancers (ALB for HTTP traffic, NLB for TCP/UDP).
  + Route 53 for DNS management and failover routing.
  + AWS WAF for protection against common attacks (SQLi, XSS).

**Infrastructure Provisioning (Terraform Design):**

* **Modular Code Structure:** Separate modules for VPC, EC2, RDS, S3, IAM, and EKS.
* **State Management:** *Remote state* stored in S3 with DynamoDB locking for concurrent operations.
* **Resource Provisioning:**
  + VPC with public/private subnets, NAT gateways for outbound traffic from private subnets.
  + EC2 instances with user-data scripts for boot-time configuration.
  + RDS PostgreSQL with parameter groups, read replicas for high availability.
  + S3 buckets with versioning, lifecycle policies for log achievement.

**CI/CD Pipeline (Workflows):**

* **GitHub Actions:**
  + Linting (ESLint for React (Front-end), ESLint + Prettier for Node.js (Back-end)).
  + Unit tests with coverage reports using Jest (frontend) and Mocha/Chai (backend).
  + Docker image build with caching for faster builds, scanned by Docker Bench Security and Trivy.
  + Deployment jobs using kubectl and Helm for Kubernetes deployments.
  + Post-deployment smoke tests, Cypress end-to-end tests, and Slack notifications.

**Automation Tools :**

* **Bash Scripts:**
  + Automated instance provisioning, daily backups, log clean-up, and health checks.
* **Ansible Playbooks:**
  + Config management for EC2, Docker, Kubernetes nodes.
  + Rolling updates with minimal downtime.
* **Terraform Modules:**
  + Modular code for reusability across environments.

**Deployment Process (Comprehensive Enhancements):**

* **Deployment Strategies:**
  + Blue-Green Deployments with Route 53.
  + Canary Releases via Kubernetes.
  + Rolling Updates with Kubernetes Deployment objects.
* **Auto-scaling:**
  + Kubernetes *HPA* for pod scaling.
  + AWS *Auto Scaling Groups* for EC2.

**Security:**

* **IAM:**
  + *Role-based access* control with detailed policies.
  + Federation with AWS SSO.
* **Network Security:**
  + Security Groups for resource-level security.
  + NACLs for subnet-level control.
* **Secrets Management:**
  + AWS Secrets Manager for environment variables.
* **Container Security:**
  + Image signing, vulnerability scanning, and runtime security with Falco.

**Architecture Deployment Steps:**

An in-depth description of each deployment step to achieve a seamless, secure, and scalable deployment for MCRP:

1. **Provision Infrastructure with Terraform:**
   * Clone the GitHub repository containing Terraform code.
   * Configure AWS CLI with appropriate IAM roles.
   * Run terraform init to initialize plugins and backend.
   * Execute terraform plan to preview infrastructure.
   * Apply infrastructure with terraform apply to provision VPCs, EC2 instances, EKS clusters, RDS databases, S3 buckets, and IAM roles.
   * Store state files in S3 with locking in DynamoDB for team collaboration.
2. **Build Docker Images:**
   * Write Docker files for frontend (React.js) and backend (Node.js/Express) with multi-stage builds.
   * Use docker build to create optimized images.
   * Scan images for vulnerabilities using Trivy.
   * Push images to Amazon ECR for centralized storage.
3. **Deploy with Kubernetes (EKS):**
   * Install kubectl, Helm, and AWS IAM Authenticator.
   * Use Helm charts to deploy services, ingress controllers, config maps, and secrets.
   * Apply Kubernetes manifests (YAML files) for Pods, Services, Deployments, ConfigMaps.
4. **CI/CD Automation:**
   * Define GitHub Actions workflows (.github/workflows/\*) for linting, testing, building, and deploying.
   * Trigger workflows on PR merges.
   * Automate rollbacks if failures occur during deployment.
5. **Monitoring & Logging:**
   * Deploy Prometheus for metrics collection and Grafana for visualization.
   * Integrate AWS CloudWatch for instance-level monitoring and AWS X-Ray for tracing API requests.
6. **Security Implementation:**
   * Define least privilege IAM roles.
   * Configure NACLs and Security Groups for network access.
   * Use AWS Secrets Manager for sensitive data.
   * Enable container runtime protection with Falco.

**Weekly (Cloud-Native) Deliverables :**

* **Architecture Design:** Advanced VM setups, Kubernetes optimizations, Docker enhancements.
* **Automation Development:** CI/CD pipelines, Bash scripts, Ansible configurations.
* **Deployment Optimization:** Cost efficiency (Spot Instances, rightsizing), enhanced scalability.
* **Security Audits:** Regular reviews, updates, and incident response drills.

**Conclusion:**

The deployment strategy provides a technically well-defined roadmap as well as emphasizes best practices in *scalability*, *security*, *automation*, and *cost-efficiency*. It ensures that the MCRP project leverages cutting-edge cloud technologies, enabling smooth deployments, seamless collaboration, and optimal performance across all environments. The integration of modern DevOps tools, modular infrastructure, and comprehensive security measures guarantees a robust cloud environment capable of handling *dynamic* project needs and future growth.