POC 07: Digital Media Platform - Kubernetes Monitoring and Release Support

Prepared for: **Saurabh Dindokar**

Role: **AWS DevOps Engineer**

Public-safe project name: **Digital Media Platform**

Confidentiality Note

This document is intentionally public-safe. It does not include real client names, internal project names, organization-owned source code, private URLs, IP addresses, credentials, account IDs, screenshots, or any confidential implementation details.

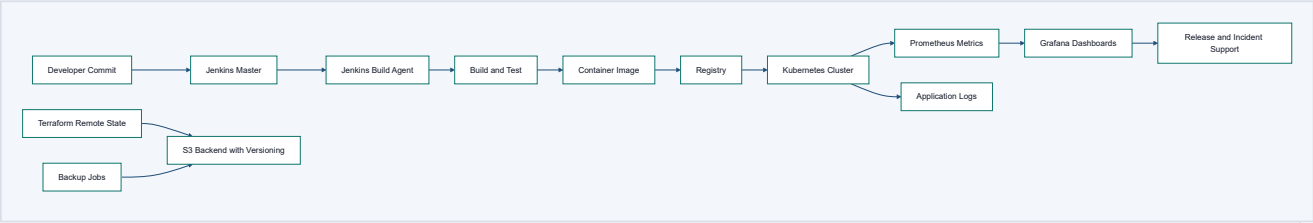
Work Sample Summary

Public-safe work sample for Kubernetes operations, Jenkins master-agent build pattern, S3 backup strategy, Terraform backend versioning, Prometheus, Grafana, and release support.

Problem Statement

- â€¢ The platform required stronger visibility into Kubernetes workloads and release behavior.
- â€¢ CI/CD worker execution needed a scalable master-agent style pattern.
- â€¢ Backup and Terraform state practices needed documentation to support safe operations.

Mermaid Architecture Diagram



Mermaid architecture diagram

My Responsibilities

- â€¢ Supported Kubernetes deployment and monitoring practices for application release workflows.
- â€¢ Documented Jenkins master-agent pipeline model for build execution and separation of workloads.
- â€¢ Prepared monitoring flow using Prometheus and Grafana for workload visibility.
- â€¢ Captured Terraform state, S3 backup, and release support practices in public-safe form.

Implementation Approach

- â€¢ Jenkins agents execute build steps while the master coordinates pipeline flow.
- â€¢ Kubernetes workloads are observed through metrics, logs, dashboards, and deployment events.

â€¢ Terraform state is stored remotely with versioning and access controls.

â€¢ Backup jobs and release notes support incident response and recovery confidence.

Reliability, Security and Operations Focus

â€¢ Health checks, validation steps, and rollback planning were treated as part of deployment quality, not afterthoughts.

â€¢ Secrets, access boundaries, private networking, backup retention, and monitoring visibility were documented wherever applicable.

â€¢ The POC is written so another engineer can understand the flow, reproduce the approach, and extend it safely without exposing confidential data.

Validation Checklist

Review architecture and confirm each component has a clear responsibility.

Validate deployment or automation steps in a non-production environment first.

Confirm logs, health checks, backup behavior, and rollback steps are documented.

Confirm access, secrets, and network boundaries follow least-privilege expectations.

Capture final runbook notes for handover and support readiness.

Expected Outcomes

â€¢ Better visibility into Kubernetes releases.

â€¢ Cleaner CI/CD execution model using build agents.

â€¢ Safer Terraform state handling through remote backend/versioning.

â€¢ Improved operational readiness for support and incident response.

Technologies Used

Kubernetes, Jenkins, Jenkins Agents, Docker, Prometheus, Grafana, S3, Terraform Backend, Shell Scripting, AWS

Naukri Work Sample Description

Public-safe DevOps POC by Saurabh Dindokar covering architecture, responsibilities, implementation approach, validation checklist, operational focus, and measurable delivery outcomes. The document uses generic project naming and does not disclose any confidential client or organization information.