30-Day Emergency Plan

1. Immediate Fixes to Achieve 90%+ Deployment Success Rate

To stabilize deployments and reduce failures, the following emergency fixes will be implemented

- Standardize deployment process across all microservices using a single CI/CD pipeline (e.g., GitHub Actions).
- Add automated unit and integration tests to block faulty builds before deployment.

- Enforce environment configuration consistency by introducing Infrastructure as Code (IaC) templates (Terraform/Ansible).
- Implement automated database migration validations using pre-deployment dry runs.
- Introduce container image scanning and dependency scanning at build time to shift security left.

2. Rollback Strategy within 15 Minutes

To reduce downtime and risk during failed deployments, a rollback strategy will be introduced:
- Implement Blue-Green deployments for critical services to allow quick rollback.(e.g. Code deploy)
- Use versioned artifacts (Docker images and Helm charts) for instant re-deployment of last known good state.
- Automate rollback triggers based on health check failures within 5 minutes of deployment.
- Maintain backup and restore procedures for databases with point-in-time recovery.

3. Establish Basic Monitoring

To quickly detect and address issues in deployments, basic monitoring will be established:
- Integrate centralized logging (ELK/EFK stack or AWS CloudWatch).
- Enable metrics collection with Prometheus + Grafana dashboards for microservices.
- Implement alerting for deployment failures, health check failures, and error-rate thresholds.
- Provide service owners with real-time notifications (Slack/Teams integration).

4. Timeline

- **Week 1**: Standardize CI/CD pipeline + enable rollback via blue-green deployments.
- **Week 2 - 3**: Implement IaC for environment consistency and start central logging.
- **Week 4 - 5:** Automate DB migrations and integrate unit/integration testing.
- **5 Days:** Enable monitoring dashboards, alerts,