Platform and SRE Support Handbook for Java-Based Development Environments

Abul Hasan Fahad

May 1, 2025

Contents

1 Introduction

This document is intended as a comprehensive reference for Platform and Site Reliability Engineers (SREs) who support Java-based development environments using Angular, Spring Boot, Scala, Spark, Jenkins, Cassandra, and Hadoop.

2 System Overview

2.1 Tech Stack

- **Frontend:** Angular (served via Nginx)
- **Backend:** Spring Boot (Java)
- **Data Processing:** Spark (Scala)
- **CI/CD:** Jenkins
- **Data Storage:** Cassandra, HDFS (Hadoop)
- **Runtime:** Linux Virtual Machines (VMs)

2.2 Environment Tiers

- Development (DEV)
- System Integration Testing (SIT)
- Pre-Production Acceptance Testing (PAT)
- Production (PROD)

3 Application Types and Repository Structure

3.1 Angular Frontend

- `angular.json` - project config
- `package.json` - dependencies and scripts
- `src/environments/` - env-specific settings
- `dist/` - build output

3.2 Spring Boot Services

- `src/main/java/` - source code
- `application.yml / application-$env.yml` - config
- `target/*.jar` - deployment artefact

3.3 Spark Jobs (Scala)

- `src/main/scala/` - Spark job logic
- `conf/$env.conf` - job configs
- `target/*.jar` - Spark job package

4 CI/CD with Jenkins

4.1 Pipeline Structure

- Multi-branch pipelines for dev/test/prod
- Shared libraries for reusable stages
- Parameterized builds for environment selection

4.2 Sample Jenkinsfile Snippet

```
pipeline {
  agent any
  parameters {
    string(name: 'TARGET_ENV', defaultValue: 'dev')
  }
  stages {
    stage('Build') {
      steps { sh './mvnw␣clean␣package␣-DskipTests' }
    }
    stage('Deploy') {
      steps { build job: "deploy-${params.TARGET_ENV}" }
    }
  }
}
```

5 Deployment Patterns

5.1 Angular

- Build: `npm run build -- --configuration=$ENV`
- Deploy: Serve via Nginx

5.2 Spring Boot

- Deployable: `.jar` file
- Start via systemd or Docker

5.3 Spark Jobs

- Run with `spark-submit`
- Monitor on YARN or Spark UI

6 Cassandra and Hadoop Integration

6.1 Cassandra

- Use `cqlsh` for queries
- Monitor with `nodetool status`

6.2 Hadoop

- Store bulk data in HDFS
- Use `hdfs dfs -ls /path/` to browse

7 Monitoring and Observability

7.1 Spring Boot

- Use Spring Actuator: `/actuator/health`, `/actuator/prometheus`

7.2 Spark

- Monitor via Spark UI or YARN RM Web UI

7.3 Cassandra

- Use `nodetool cfstats`, and integrate with Prometheus

8 Incident Response Playbook

- Angular down → Check Nginx logs and SPA errors
- Spring Boot fail → Check service logs and DB connection
- Spark job stalled → Check YARN resource availability
- Cassandra inconsistency → Run `nodetool repair`

9 Backup and Restore Procedures

9.1 Cassandra

- Use `nodetool snapshot` to backup
- Restore from snapshot with `sstableloader`

9.2 HDFS

- Create snapshot of important directories
- Use `hdfs dfs -cp` for restores

10 Security and Access Controls

- Use LDAP or AD integration for SSH and Jenkins
- Vault or SSM for secrets management
- Limit sudo access to release accounts only

11 Configuration Management

- Use Ansible for consistent deployments
- Template `.yml`, `.conf`, `.properties` per env

12 Networking and Ports Map

- Angular via Nginx → Port 80/443
- Spring Boot → Port 8080
- Cassandra → 9042 (CQL), 7199 (JMX)
- Spark UI → 4040+, YARN → 8088

13 Runbook Appendices

13.1 Spring Boot First-Time Startup

1. Copy `.jar` to server
2. Run `java -jar app.jar --spring.profiles.active=dev`
3. Check `/actuator/health`

13.2 Spark Job Submission

```
spark-submit --master yarn --deploy-mode cluster \
--conf config.file=conf/dev.conf target/job.jar
```