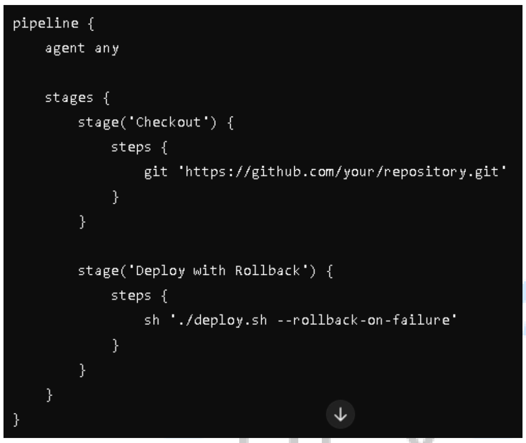
JENKINSFILE SCENARIOS WITH AUTMATIONS

1) Continuous Deployment with Rollback Strategy:

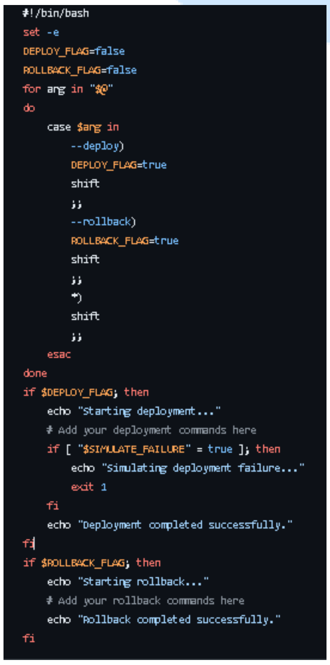
- **Scenario**: Automatically deploy applications to production with the ability to rollback on failure.
- **Benefit**: Ensures minimal downtime and quick recovery from deployment failures, maintaining application availability and user trust.

Jenkinsfile link - scenario_based_learnings/Jenkinsfiles/jenkinsfile1.txt at main · praveen1994dec/scenario_based_learnings (github.com)



```
pipeline {
    agent any

    stages {
        stage('Checkout') {
            steps {
                git 'https://github.com/your/repository.git'
            }
        }

        stage('Deploy with Rollback') {
            steps {
                sh './deploy.sh --rollback-on-failure'
            }
        }
    }
}
```

Script Link - scenario_based_learnings/Shell Scripts/1-deploy.sh at main · praveen1994dec/scenario_based_learnings (github.com)

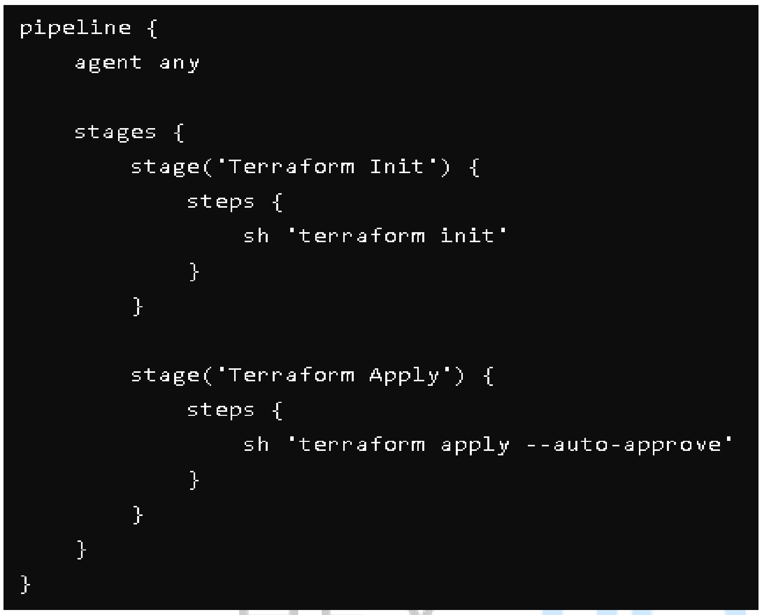


```
#!/bin/bash
set -e
DEPLOY_FLAG=false
ROLLBACK_FLAG=false
for arg in "$@"
do
    case $arg in
        --deploy)
        DEPLOY_FLAG=true
        shift
        ;;
        --rollback)
        ROLLBACK_FLAG=true
        shift
        ;;
        *)
        shift
        ;;
    esac
done
if $DEPLOY_FLAG; then
    echo "Starting deployment..."
    # Add your deployment commands here
    if [ "$SIMULATE_FAILURE" = true ]; then
        echo "Simulating deployment failure..."
        exit 1
    fi
    echo "Deployment completed successfully."
fi
if $ROLLBACK_FLAG; then
    echo "Starting rollback..."
    # Add your rollback commands here
    echo "Rollback completed successfully."
fi
```

2) Infrastructure as Code (IaC) with Terraform:

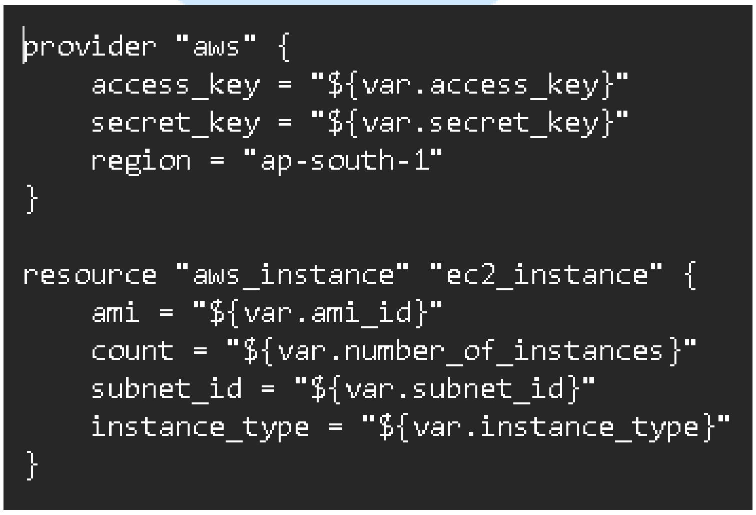
- **Scenario**: Use Terraform to provision and manage infrastructure.
- **Benefit**: Enables consistent, repeatable, and automated infrastructure provisioning, reducing human error and speeding up the setup process.

Jenkinfile Link - [scenario_based_learnings/Jenkinsfiles/jenkinsfile2.txt at main · praveen1994dec/scenario_based_learnings (github.com)]



```
pipeline {
    agent any

    stages {
        stage('Terraform Init') {
            steps {
                sh 'terraform init'
            }
        }

        stage('Terraform Apply') {
            steps {
                sh 'terraform apply --auto-approve'
            }
        }
    }
}
```

Script Link - [scenario_based_learnings/Terraform/1-Terraform.tf at main · praveen1994dec/scenario_based_learnings (github.com)]

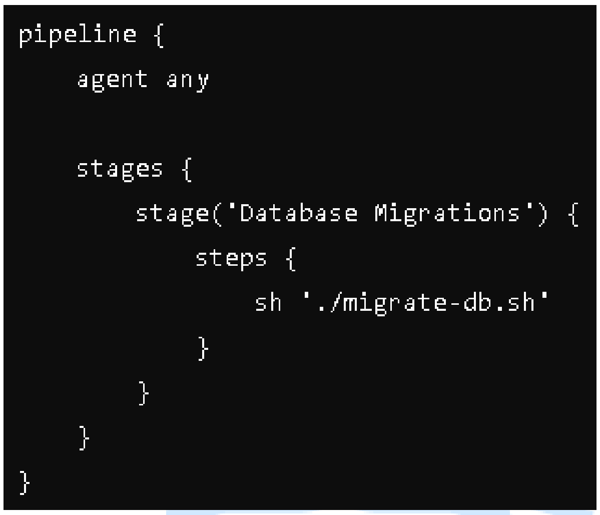


```
provider "aws" {
    access_key = "${var.access_key}"
    secret_key = "${var.secret_key}"
    region = "ap-south-1"
}

resource "aws_instance" "ec2_instance" {
    ami = "${var.ami_id}"
    count = "${var.number_of_instances}"
    subnet_id = "${var.subnet_id}"
    instance_type = "${var.instance_type}"
}
```

3)Automated Database Migrations:

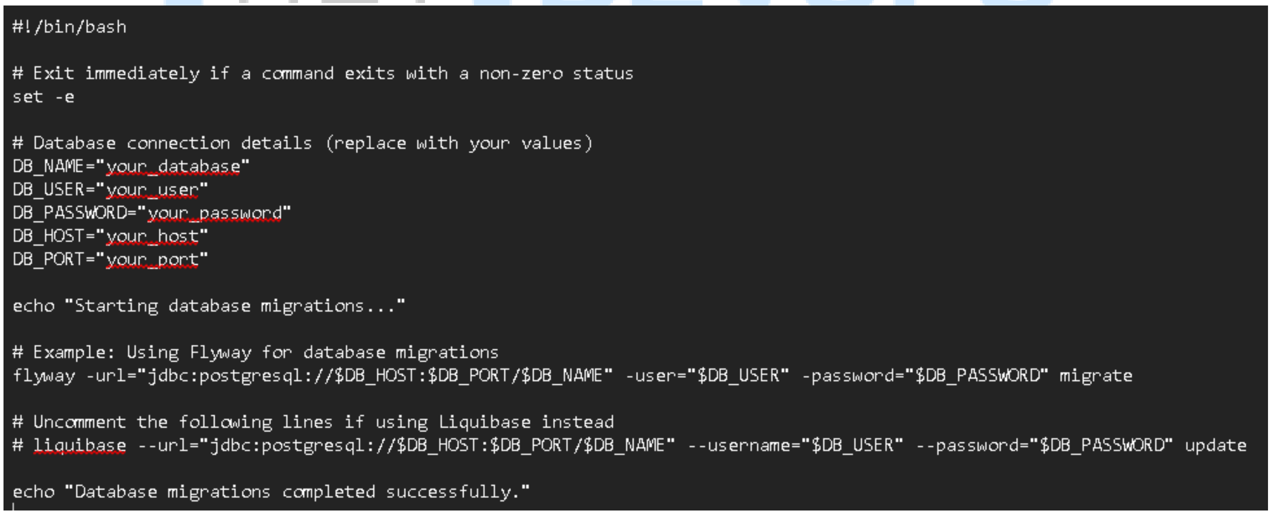
- **Scenario**: Apply database migrations as part of the deployment process.
- **Benefit**: Ensures database schema changes are applied in sync with application deployments, preventing mismatches and runtime errors.

Jenkinfile link - scenario_based_learnings/Jenkinsfiles/jenkinsfile3.txt at main · praveen1994dec/scenario_based_learnings (github.com)



```
pipeline {
    agent any

    stages {
        stage('Database Migrations') {
            steps {
                sh './migrate-db.sh'
            }
        }
    }
}
```

Script Link - scenario_based_learnings/Shell Scripts/2-migrate-db.sh at main · praveen1994dec/scenario_based_learnings (github.com)

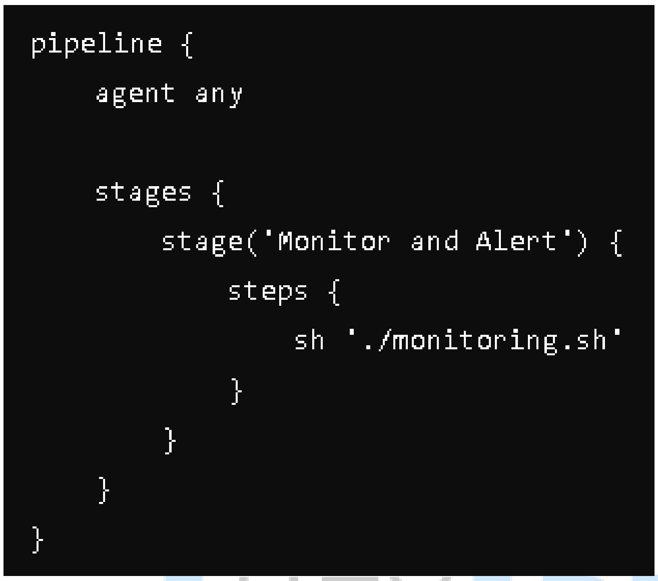


```
#!/bin/bash

# Exit immediately if a command exits with a non-zero status
set -e

# Database connection details (replace with your values)
DB_NAME="your_database"
DB_USER="your_user"
DB_PASSWORD="your_password"
DB_HOST="your_host"
DB_PORT="your_port"

echo "Starting database migrations..."

# Example: Using Flyway for database migrations
flyway -url="jdbc:postgresql://$DB_HOST:$DB_PORT/$DB_NAME" -user="$DB_USER" -password="$DB_PASSWORD" migrate

# Uncomment the following lines if using Liquibase instead
# liquibase --url="jdbc:postgresql://$DB_HOST:$DB_PORT/$DB_NAME" --username="$DB_USER" --password="$DB_PASSWORD" update

echo "Database migrations completed successfully."
```

4)Monitor and Alert on Application Health:

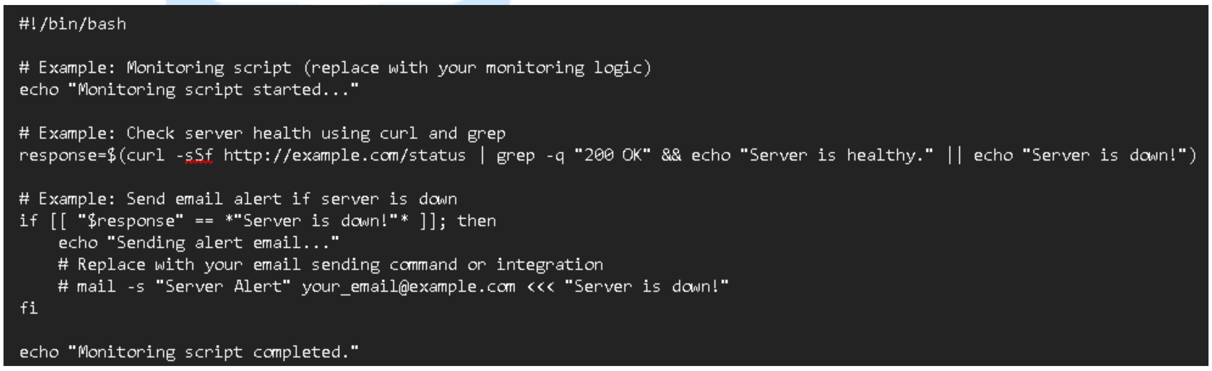
- **Scenario**: Monitor application health and alert if any issues are detected.
- **Benefit**: Provides early detection of issues, enabling quick response and resolution, thus maintaining application performance and reliability.

Jenkinfile link - [scenario_based_learnings/Jenkinsfiles/jenkinsfile4.txt at main · praveen1994dec/scenario_based_learnings (github.com)]()



```
pipeline {
    agent any

    stages {
        stage('Monitor and Alert') {
            steps {
                sh './monitoring.sh'
            }
        }
    }
}
```

Script Link - [scenario_based_learnings/Shell Scripts/3-monitoring.sh at main · praveen1994dec/scenario_based_learnings (github.com)]()

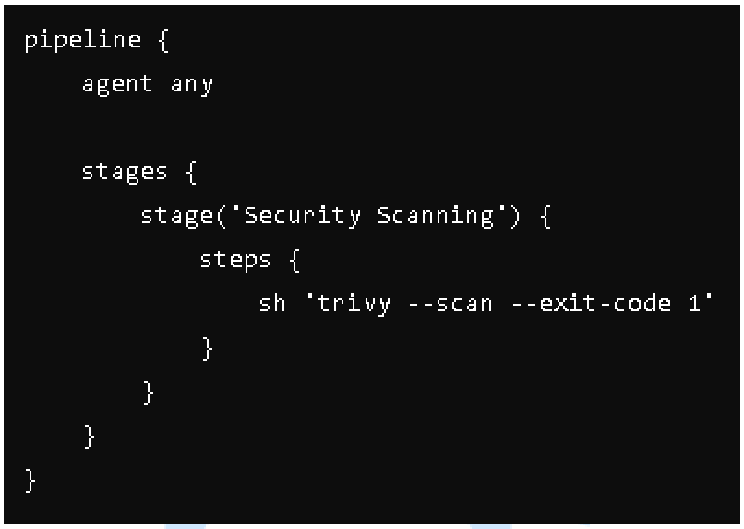


```
#!/bin/bash

# Example: Monitoring script (replace with your monitoring logic)
echo "Monitoring script started..."

# Example: Check server health using curl and grep
response=$(curl -sSf http://example.com/status | grep -q "200 OK" && echo "Server is healthy." || echo "Server is down!")

# Example: Send email alert if server is down
if [[ "$response" == *"Server is down!"* ]]; then
    echo "Sending alert email..."
    # Replace with your email sending command or integration
    # mail -s "Server Alert" your_email@example.com <<< "Server is down!"
fi

echo "Monitoring script completed."
```

5)Automated Security Scanning:

- **Scenario**: Perform security scanning of the application and Docker images.
- **Benefit**: Detects vulnerabilities early in the development process, enhancing application security and compliance with industry standards.

Jenkinfile link - scenario_based_learnings/Jenkinsfiles/jenkinsfile5.txt at main · praveen1994dec/scenario_based_learnings (github.com)

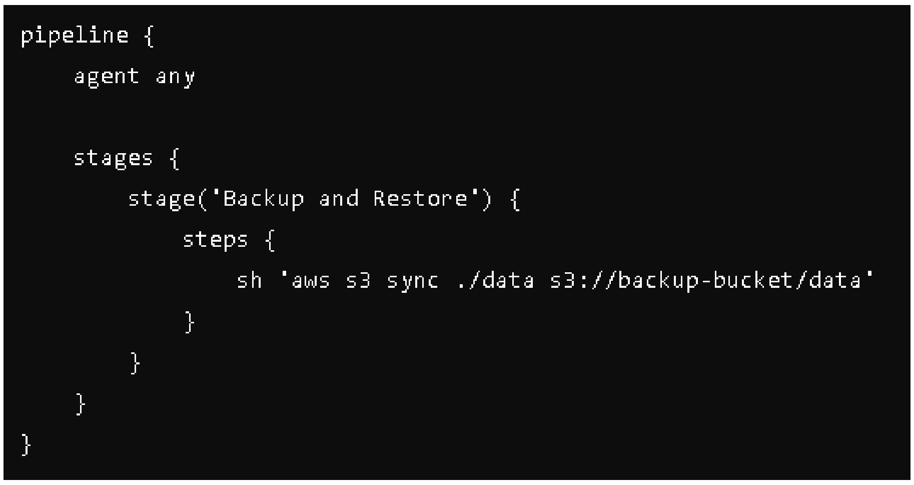


```
pipeline {
    agent any

    stages {
        stage('Security Scanning') {
            steps {
                sh 'trivy --scan --exit-code 1'
            }
        }
    }
}
```

6)Backup and Restore Strategy:

- **Scenario**: Implement a backup and restore strategy for databases and application data.
- **Benefit**: Protects against data loss and corruption, ensuring business continuity and data integrity.

Jenkinfile link - scenario_based_learnings/Jenkinsfiles/jenkinsfile6.txt at main · praveen1994dec/scenario_based_learnings (github.com)

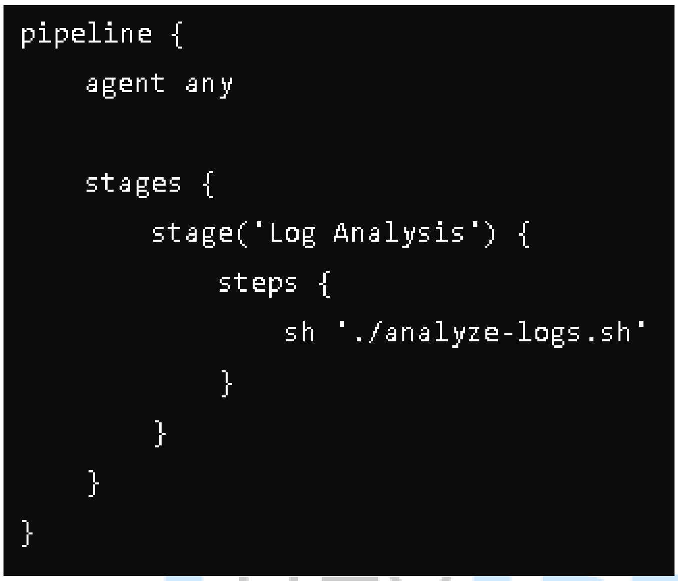


```
pipeline {
    agent any

    stages {
        stage('Backup and Restore') {
            steps {
                sh 'aws s3 sync ./data s3://backup-bucket/data'
            }
        }
    }
}
```

7)Log Aggregation and Analysis:

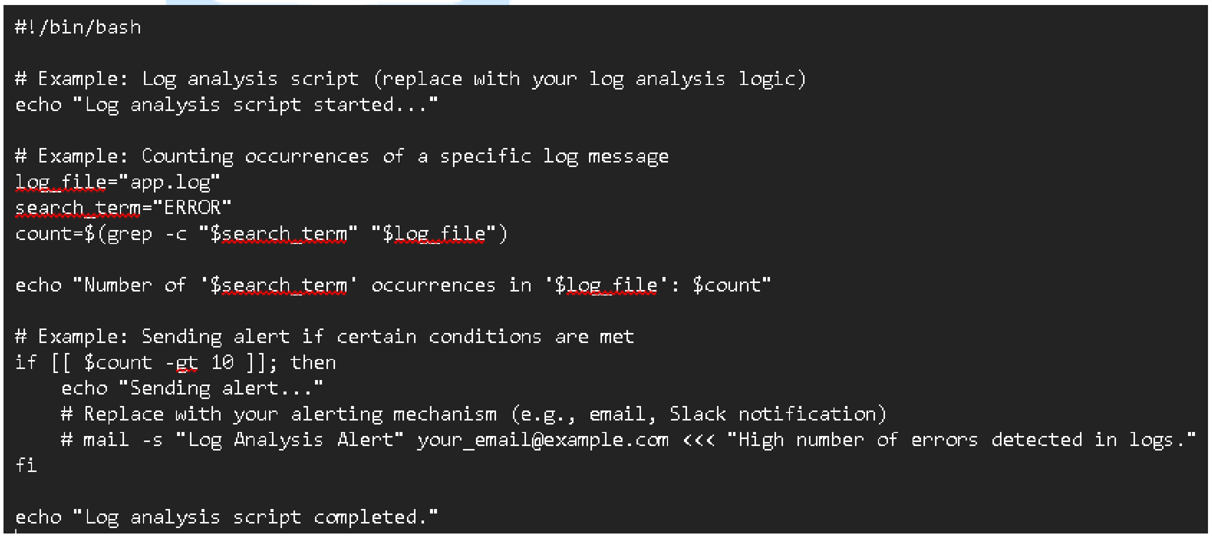
- **Scenario**: Collect and analyze logs from the application.
- **Benefit**: Centralizes log data for easier troubleshooting, performance monitoring, and anomaly detection, leading to quicker issue resolution.

Jenkinfile link - [scenario_based_learnings/Jenkinsfiles/jenkinsfile7.txt at main · praveen1994dec/scenario_based_learnings (github.com)]()



```
pipeline {
    agent any

    stages {
        stage('Log Analysis') {
            steps {
                sh './analyze-logs.sh'
            }
        }
    }
}
```

Script Link -[scenario_based_learnings/Shell Scripts/4-analyze-logs.sh at main · praveen1994dec/scenario_based_learnings (github.com)]()

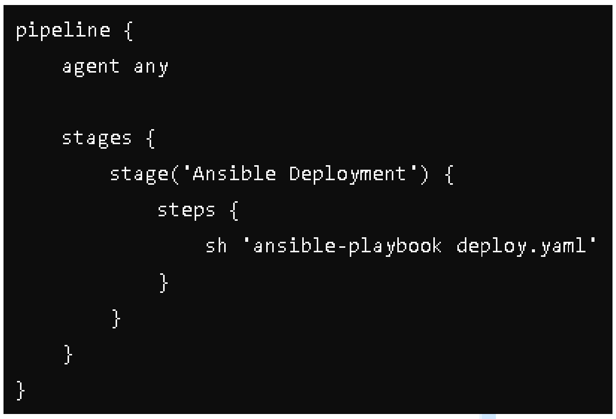


```
#!/bin/bash

# Example: Log analysis script (replace with your log analysis logic)
echo "Log analysis script started..."

# Example: Counting occurrences of a specific log message
log_file="app.log"
search_term="ERROR"
count=$(grep -c "$search_term" "$log_file")

echo "Number of '$search_term' occurrences in '$log_file': $count"

# Example: Sending alert if certain conditions are met
if [[ $count -gt 10 ]]; then
    echo "Sending alert..."
    # Replace with your alerting mechanism (e.g., email, Slack notification)
    # mail -s "Log Analysis Alert" your_email@example.com <<< "High number of errors detected in logs."
fi

echo "Log analysis script completed."
```

8)Configuration Management with Ansible:

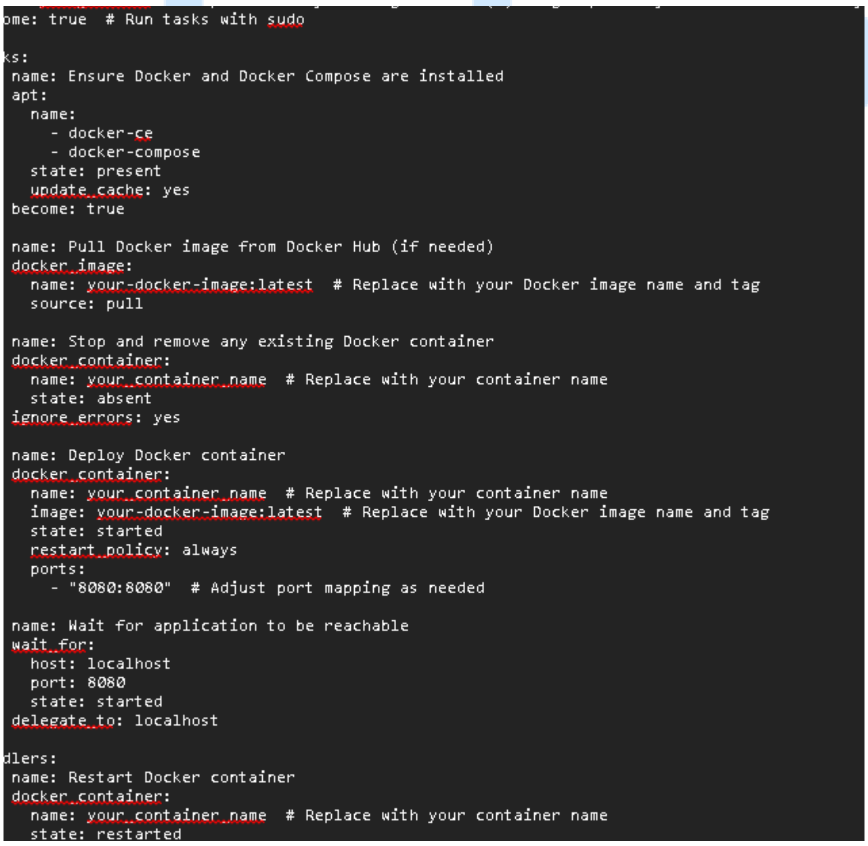
- **Scenario**: Use Ansible to manage server configurations and deployments.
- **Benefit**: Automates and standardizes server configuration management, reducing configuration drift and ensuring consistency across environments.

Jenkinfile link - [scenario_based_learnings/Jenkinsfiles/jenkinsfile8.txt at main · praveen1994dec/scenario_based_learnings (github.com)]



```
pipeline {
    agent any

    stages {
        stage('Ansible Deployment') {
            steps {
                sh 'ansible-playbook deploy.yaml'
            }
        }
    }
}
```

Playbook Link - [scenario_based_learnings/Ansible Playbook/1-ansible-playbook.yaml at main · praveen1994dec/scenario_based_learnings (github.com)]

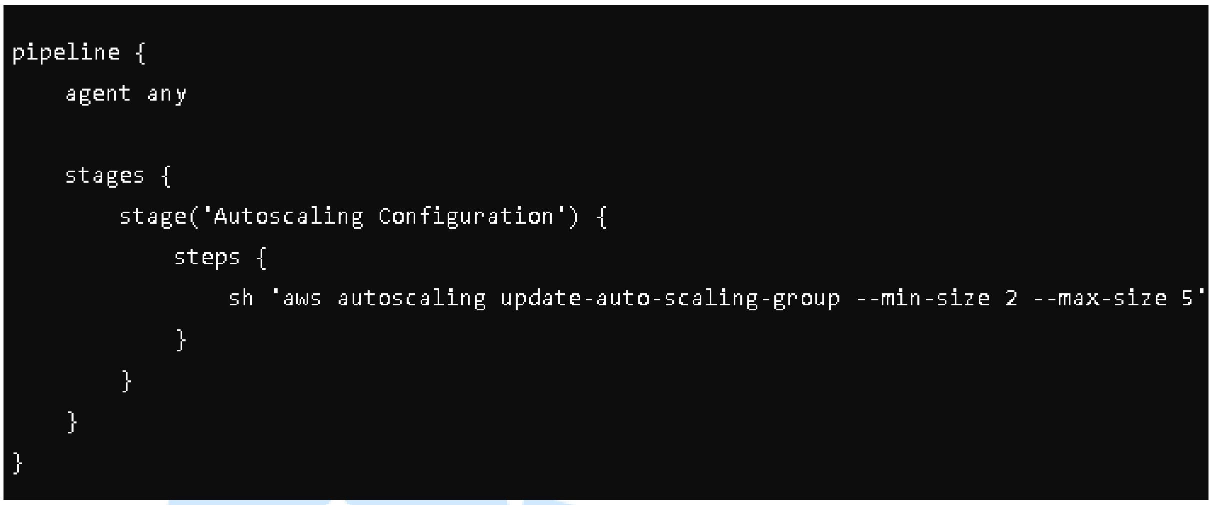


```
ome: true  # Run tasks with sudo

ks:
 name: Ensure Docker and Docker Compose are installed
 apt:
   name:
     - docker-ce
     - docker-compose
   state: present
   update_cache: yes
 become: true

 name: Pull Docker image from Docker Hub (if needed)
 docker_image:
   name: your-docker-image:latest  # Replace with your Docker image name and tag
   source: pull

 name: Stop and remove any existing Docker container
 docker_container:
   name: your_container_name  # Replace with your container name
   state: absent
 ignore_errors: yes

 name: Deploy Docker container
 docker_container:
   name: your_container_name  # Replace with your container name
   image: your-docker-image:latest  # Replace with your Docker image name and tag
   state: started
   restart_policy: always
   ports:
     - "8080:8080"  # Adjust port mapping as needed

 name: Wait for application to be reachable
 wait_for:
   host: localhost
   port: 8080
   state: started
 delegate_to: localhost

dlers:
 name: Restart Docker container
 docker_container:
   name: your_container_name  # Replace with your container name
   state: restarted
```

9)Autoscaling Configuration:

- **Scenario**: Configure autoscaling for dynamic resource management.
- **Benefit**: Ensures optimal resource usage by automatically scaling resources up or down based on demand, reducing costs and improving performance.

Jenkinfile link - [scenario_based_learnings/Jenkinsfiles/jenkinsfile9.txt at main · praveen1994dec/scenario_based_learnings (github.com)]

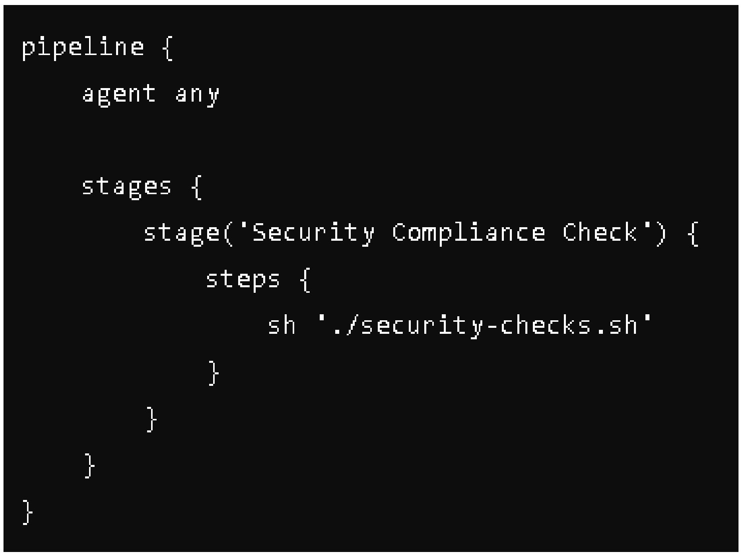


```
pipeline {
    agent any

    stages {
        stage('Autoscaling Configuration') {
            steps {
                sh 'aws autoscaling update-auto-scaling-group --min-size 2 --max-size 5'
            }
        }
    }
}
```

10)Security Compliance Check:

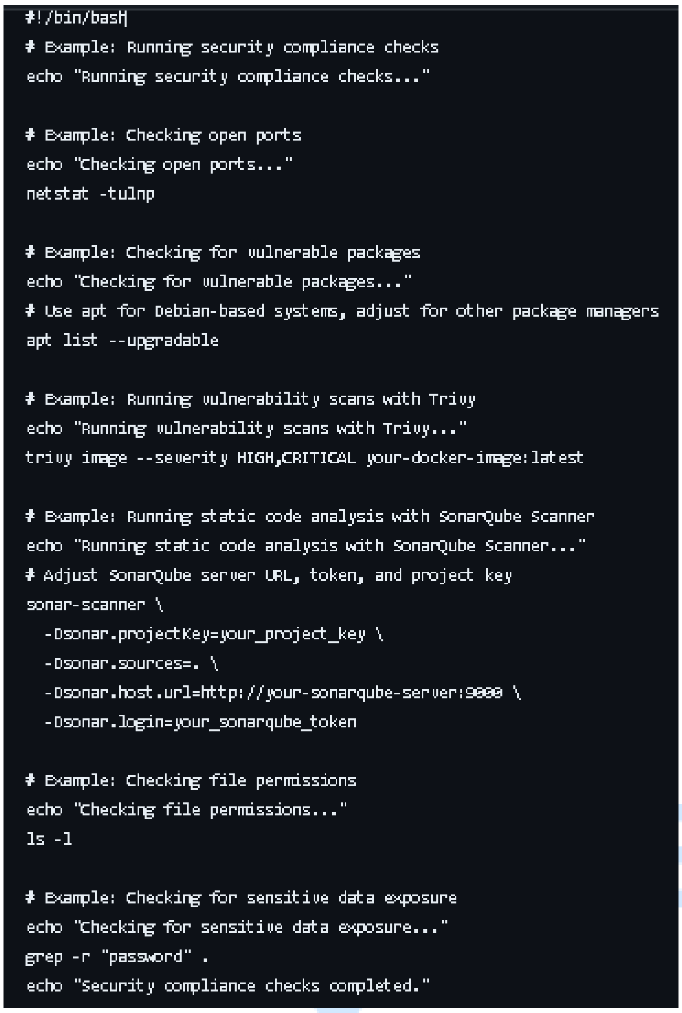
- **Scenario**: Integrate security compliance checks in the pipeline.
- **Benefit**: Ensures the application adheres to security standards and compliance requirements, reducing vulnerabilities and legal risks.

Jenkinfile link - [scenario_based_learnings/Jenkinsfiles/jenkinsfile10.txt at main · praveen1994dec/scenario_based_learnings (github.com)]



```
pipeline {
    agent any

    stages {
        stage('Security Compliance Check') {
            steps {
                sh './security-checks.sh'
            }
        }
    }
}
```

Script Link -scenario_based_learnings/Shell Scripts/5-security-checks.sh at main · praveen1994dec/scenario_based_learnings (github.com)

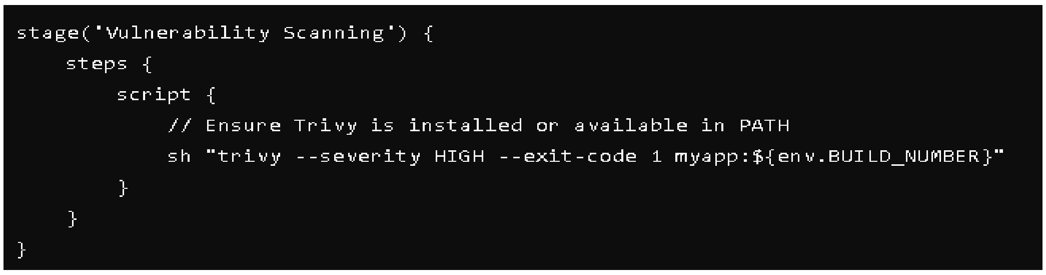


```
#!/bin/bash
# Example: Running security compliance checks
echo "Running security compliance checks..."

# Example: Checking open ports
echo "Checking open ports..."
netstat -tulnp

# Example: Checking for vulnerable packages
echo "Checking for vulnerable packages..."
# Use apt for Debian-based systems, adjust for other package managers
apt list --upgradable

# Example: Running vulnerability scans with Trivy
echo "Running vulnerability scans with Trivy..."
trivy image --severity HIGH,CRITICAL your-docker-image:latest

# Example: Running static code analysis with SonarQube Scanner
echo "Running static code analysis with SonarQube Scanner..."
# Adjust SonarQube server URL, token, and project key
sonar-scanner \
  -Dsonar.projectKey=your_project_key \
  -Dsonar.sources=. \
  -Dsonar.host.url=http://your-sonarqube-server:9000 \
  -Dsonar.login=your_sonarqube_token

# Example: Checking file permissions
echo "Checking file permissions..."
ls -l

# Example: Checking for sensitive data exposure
echo "Checking for sensitive data exposure..."
grep -r "password" .
echo "Security compliance checks completed."
```

11)Vulnerability Scanning in CI/CD Pipeline:

- **Scenario**: Integrate Trivy into the CI/CD pipeline to scan Docker images for vulnerabilities.
- **Benefit**: Identifies security vulnerabilities early in the development process, ensuring that only secure images are deployed to production.

Jenkinfile link - scenario_based_learnings/Jenkinsfiles/jenkinsfile11.txt at main · praveen1994dec/scenario_based_learnings (github.com)

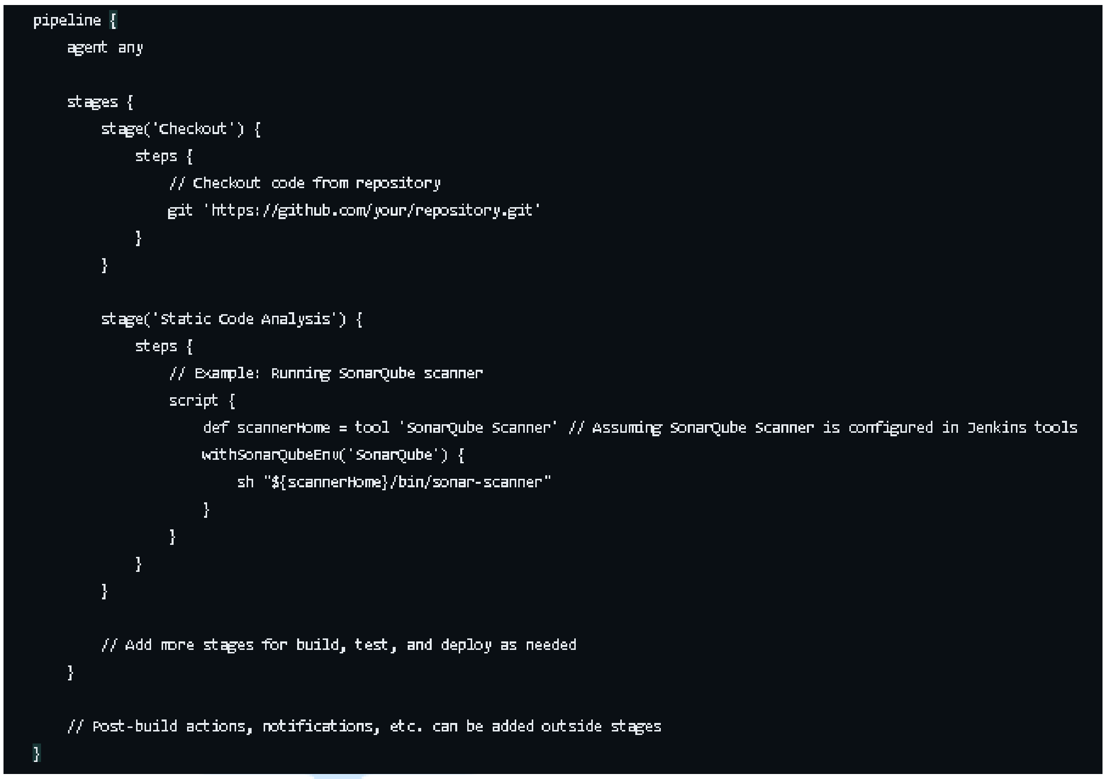


```
stage('Vulnerability Scanning') {
    steps {
        script {
            // Ensure Trivy is installed or available in PATH
            sh "trivy --severity HIGH --exit-code 1 myapp:${env.BUILD_NUMBER}"
        }
    }
}
```

12)Static Code Analysis in CI/CD Pipeline:

- **Scenario**: Integrate SonarQube for static code analysis as part of the CI/CD pipeline.
- **Benefit**: Ensures code quality and security by detecting bugs, code smells, and vulnerabilities before deployment.

Jenkinfile link - [scenario_based_learnings/Jenkinsfiles/jenkinsfile12.txt at main · praveen1994dec/scenario_based_learnings (github.com)]

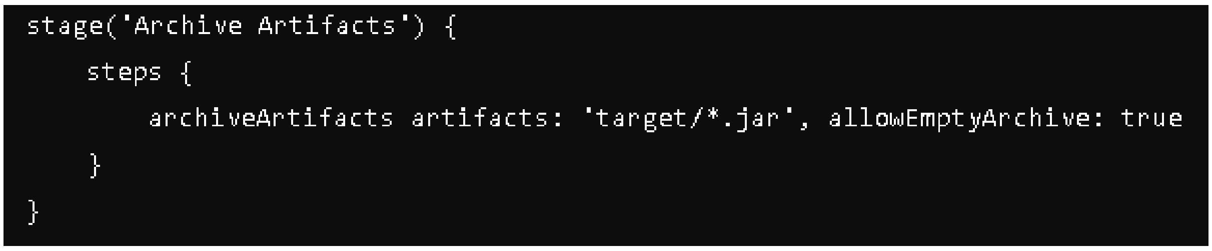


```
pipeline {
    agent any

    stages {
        stage('Checkout') {
            steps {
                // Checkout code from repository
                git 'https://github.com/your/repository.git'
            }
        }

        stage('Static Code Analysis') {
            steps {
                // Example: Running SonarQube scanner
                script {
                    def scannerHome = tool 'SonarQube Scanner' // Assuming SonarQube Scanner is configured in Jenkins tools
                    withSonarQubeEnv('SonarQube') {
                        sh "${scannerHome}/bin/sonar-scanner"
                    }
                }
            }
        }

        // Add more stages for build, test, and deploy as needed
    }

    // Post-build actions, notifications, etc. can be added outside stages
}
```

13)Storing Build Artifacts:

- **Scenario**: Use Artifactory to store build artifacts from the CI/CD pipeline.
- **Benefit**: Provides a central repository for managing and sharing build artifacts, ensuring consistency and traceability.

Jenkinfile link - [scenario_based_learnings/Jenkinsfiles/jenkinsfile13.txt at main · praveen1994dec/scenario_based_learnings (github.com)]

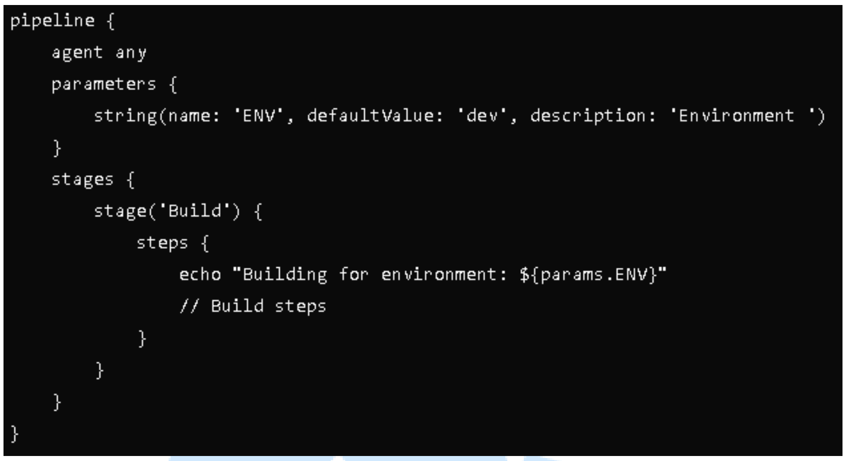


```
stage('Archive Artifacts') {
    steps {
        archiveArtifacts artifacts: 'target/*.jar', allowEmptyArchive: true
    }
}
```

14) Passing Parameters:

- Scenario: Accept user inputs at build time to customize the build process.
- Benefits**:** Allows customization of pipeline behavior based on different parameters such as environment (dev, test, prod), branch (master, develop), or specific settings (enable tests).

Jenkinfile link - [scenario_based_learnings/Jenkinsfiles/jenkinsfile14.txt at main · praveen1994dec/scenario_based_learnings (github.com)]

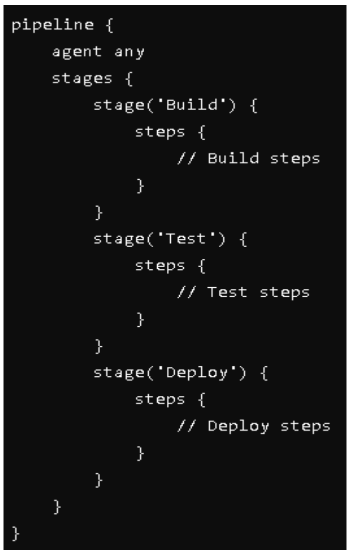


```
pipeline {
    agent any
    parameters {
        string(name: 'ENV', defaultValue: 'dev', description: 'Environment ')
    }
    stages {
        stage('Build') {
            steps {
                echo "Building for environment: ${params.ENV}"
                // Build steps
            }
        }
    }
}
```

15) Multi-Stage Pipeline:

- Scenario: Organize the build process into multiple stages.
- Benefits:Facilitates the sequential execution of distinct stages (e.g., build, test, deploy) within a single pipeline.

Jenkinfile link - [scenario_based_learnings/Jenkinsfiles/jenkinsfile15.txt at main · praveen1994dec/scenario_based_learnings (github.com)]

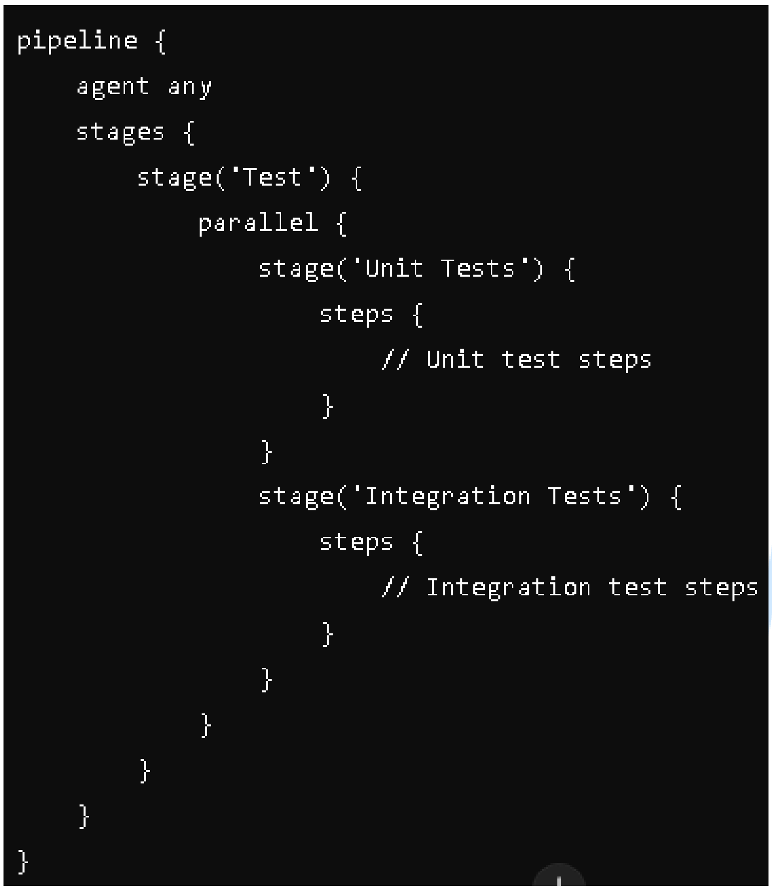


```
pipeline {
    agent any
    stages {
        stage('Build') {
            steps {
                // Build steps
            }
        }
        stage('Test') {
            steps {
                // Test steps
            }
        }
        stage('Deploy') {
            steps {
                // Deploy steps
            }
        }
    }
}
```

16) Parallel Jobs:

- Scenario: Run different stages or steps in parallel to speed up the pipeline.
- Benefits**:**Accelerates pipeline execution by running multiple tasks concurrently, reducing overall build/test time.and also Maximizes the utilization of available resources (agents/containers) by parallelizing independent tasks

Jenkinfile link - .scenario_based_learnings/Jenkinsfiles/jenkinsfile16.txt at main · praveen1994dec/scenario_based_learnings (github.com)



```
pipeline {
    agent any
    stages {
        stage('Test') {
            parallel {
                stage('Unit Tests') {
                    steps {
                        // Unit test steps
                    }
                }
                stage('Integration Tests') {
                    steps {
                        // Integration test steps
                    }
                }
            }
        }
    }
}
```