CD Jenkins in FEG

Following how to will describe how Automated Deployment can be done in FEG. It is not the only way to do things but it fulfils the recommended minimum at least for Java applications.

This guide to deployment of applications to OpenShift assumes that your application has an image in Quay and has a git repository with working Helm 3 chart.

It also assumes there is a hierarchy of value files that can be accessed via parametrised paths(like mychartdir/values/<country>/<environment>-values.yaml)

All deployments must be run from jenkins MASTER! Ci jenkins in FEG has access to OCP clusters only from agent with label "master"! No other slave has access there and will not be able to deploy anything.

General process of deployment

The process of deployment should at least consist of following steps:

1. Check deployment JIRA Ticket(for STG/PRD environments)
    a. Comment Jira ticket deployment started
2. Checkout HELM chart and values files
3. Upgrade helm release
4. Run functional tests(executes external parametrised test pipeline)
5. Mark Jira as deployed
    a. comment Jira ticket deployment finished

Prerequisites of such deployment

Deployment into OCP cluster has to be done via some user. To handle correct RBAC settings Service account is created for each service with appropriate access rights.

In order to work correctly one has to have:

- Deployment service account created for his Namespaces in all clusters
- K8SCONFIG file stored in Vault - this contains tokens, contexts and cluster API endpoints to be used with OC/KUBECTL commands inside pipeline

Those are usually created and handled by DevOps team.

Example of k8sconfig file without account tokens:

K8sconfig

```
apiVersion: v1
kind: Config
preferences: {}
```

```
clusters:
- cluster:
    insecure-skip-tls-verify: true
    server: https://czdcd-os1-master.lx.ifortuna.cz:8443
  name: dev
- cluster:
    insecure-skip-tls-verify: true
    server: https://czdct-os1-master.lx.ifortuna.cz:8443
  name: tst
- cluster:
    insecure-skip-tls-verify: true
    server: https://czdcs-os1-master.lx.ifortuna.cz:8443
  name: stg
- cluster:
    insecure-skip-tls-verify: true
    server: https://czdcp-os1-master.lx.ifortuna.cz:8443
  name: czprd
- cluster:
    insecure-skip-tls-verify: true
    server: https://czdcp-os1-master.lx.ifortuna.cz:8443
  name: roprd
- cluster:
    insecure-skip-tls-verify: true
    server: https://master-ocp01-sk.p.dc1.sk.ipa.ifortuna.cz:8443
  name: skprd
- cluster:
    server: https://master-ocp01-pl.p.dc1.pl.ipa.ifortuna.cz:8443
    insecure-skip-tls-verify: true
  name: plprd

contexts:
- context:
    cluster: dev
    namespace: bet-cz
    user: system:serviceaccount:bet:helm-bet-deployer/czdcd-os1-master-lx-ifortuna-cz:8443
  name: bet-cz/dev
- context:
    cluster: dev
    namespace: bet-sk
    user: system:serviceaccount:bet:helm-bet-deployer/czdcd-os1-master-lx-ifortuna-cz:8443
  name: bet-sk/dev
- context:
    cluster: dev
    namespace: bet-ro
    user: system:serviceaccount:bet:helm-bet-deployer/czdcd-os1-master-lx-ifortuna-cz:8443
  name: bet-ro/dev
- context:
    cluster: dev
    namespace: bet-pl
    user: system:serviceaccount:bet:helm-bet-deployer/czdcd-os1-master-lx-ifortuna-cz:8443
  name: bet-pl/dev
- context:
    cluster: tst
    namespace: bet-cz
    user: system:serviceaccount:bet:helm-bet-deployer/czdct-os1-master-lx-ifortuna-cz:8443
  name: bet-cz/tst
- context:
    cluster: tst
    namespace: bet-sk
    user: system:serviceaccount:bet:helm-bet-deployer/czdct-os1-master-lx-ifortuna-cz:8443
  name: bet-sk/tst
- context:
    cluster: tst
    namespace: bet-ro
    user: system:serviceaccount:bet:helm-bet-deployer/czdct-os1-master-lx-ifortuna-cz:8443
  name: bet-ro/tst
- context:
    cluster: tst
    namespace: bet-pl
    user: system:serviceaccount:bet:helm-bet-deployer/czdct-os1-master-lx-ifortuna-cz:8443
```

```
  name: bet-pl/tst
- context:
    cluster: stg
    namespace: bet-cz
    user: system:serviceaccount:bet:helm-bet-deployer/czdcs-os1-master-lx-ifortuna-cz:8443
  name: bet-cz/stg
- context:
    cluster: stg
    namespace: bet-sk
    user: system:serviceaccount:bet:helm-bet-deployer/czdcs-os1-master-lx-ifortuna-cz:8443
  name: bet-sk/stg
- context:
    cluster: stg
    namespace: bet-ro
    user: system:serviceaccount:bet:helm-bet-deployer/czdcs-os1-master-lx-ifortuna-cz:8443
  name: bet-ro/stg
- context:
    cluster: stg
    namespace: bet-pl
    user: system:serviceaccount:bet:helm-bet-deployer/czdcs-os1-master-lx-ifortuna-cz:8443
  name: bet-pl/stg
- context:
    cluster: czprd
    namespace: bet-cz
    user: system:serviceaccount:bet:helm-bets-deployer/czdcp-os1-master-lx-ifortuna-cz:8443
  name: bet-cz/czprd
- context:
    cluster: roprd
    namespace: bet-ro
    user: system:serviceaccount:bet:helm-bets-deployer/czdcp-os1-master-lx-ifortuna-cz:8443
  name: bet-ro/roprd
- context:
    cluster: skprd
    namespace: bet-sk
    user: system:serviceaccount:bet:helm-bet-deployer/master-ocp01-sk.p.dc1.sk.ipa.ifortuna.cz:8443
  name: bet-sk/skprd
- context:
    cluster: plprd
    namespace: bet-pl
    user: system:serviceaccount:bet:helm-bet-deployer/master-ocp01-pl.p.dc1.pl.ipa.ifortuna.cz:8443
  name: bet-pl/plprd

current-context: bet-cz/dev

users:
- name: system:serviceaccount:bet:helm-bet-deployer/czdcd-os1-master-lx-ifortuna-cz:8443
  user:
    token: <REMOVED FOR SECURITY REASONS>
- name: system:serviceaccount:bet:helm-bet-deployer/czdct-os1-master-lx-ifortuna-cz:8443
  user:
    token: <REMOVED FOR SECURITY REASONS>
- name: system:serviceaccount:bet:helm-bet-deployer/czdcs-os1-master-lx-ifortuna-cz:8443
  user:
    token: <REMOVED FOR SECURITY REASONS>
- name: system:serviceaccount:bet:helm-bet-deployer/czdcp-os1-master-lx-ifortuna-cz:8443
  user:
    token: <REMOVED FOR SECURITY REASONS>
- name: system:serviceaccount:bet:helm-bet-deployer/master-ocp01-sk.p.dc1.sk.ipa.ifortuna.cz:8443
  user:
    token: <REMOVED FOR SECURITY REASONS>
- name: system:serviceaccount:bet:helm-bet-deployer/master-ocp01-pl.p.dc1.pl.ipa.ifortuna.cz:8443
  user:
    token:  <REMOVED FOR SECURITY REASONS>
```

Once you have such file stored in vault you can easily switch contexts(ocp pojects) via(assuming there is an ENV variable pointing to k8sconfig file called KUBECONFIG):

Example switch project to BET-CZ on DEV cluster

```
kubectl config use-context bet-cz/dev
```

Helper functions in shared library

There are various shared library functions that one can use to compose his ideal deployment pipeline. Shared library is linked to all builds and can be found here.

Honorable mentions among those:

- **jiraCheckDeployTicket**
- **jiraCheckDeployTicket.requiresDeployTicket('CZPRD')**
- **addJiraComment**
- **jiraMarkDeployedToStage**
- **jiraMarkDeployedToProd**

Custom FEG plugin for quay tags

Deployment pipelines are usually parametrised. There is a custom FEG plugin imageTag that allows to fetch available tags from quay for given image.

You can find Plugin sources here.

Usage:

Example imageTag plugin usage

```
imageTag(name: 'IMAGE', credentialId: 'quay-api-token', organization: 'bet', image: 'betslip-service')
```

It will create a select box with last few newest image TAGs available in Quay repo - by default it selects the newest one.

Example pipeline

Here you will find a complete example of working deployment pipeline.

vars / betslipJavaDeployPipeline.groovy

master OS / jenkins-shared-library

```
/**
 * Betslip services java apps CI Deployment pipeline v2 script.
 */
def call(String gitURL, String deploymentServiceName, String helmChartName, String
testSuite) {

    def secrets = [
            [$class: 'VaultSecret', path: 'jenkins/bet', secretValues: [
                    [$class: 'VaultSecretValue', envVar: 'BET_DOCKER_CONFIG', vaultKey:
'docker-config'],
                    [$class: 'VaultSecretValue', envVar: 'BET_K8S_CONFIG', vaultKey:
'kubeconfig']
```

```
                ]
            ]
    ]

    def targetNamespaces = []
    def targetCluster = "<undefined>"
    def nonProdClusters = ["dev", "tst", "stg"]

    pipeline {

        agent {
            label 'master'
        }

        environment {
            KUBECONFIG='./k8sconfig'
        }

        parameters {
            choice(
                    name: 'TARGET_CLUSTER',
                    choices: ['dev', 'tst', 'stg', 'czprd', 'roprd', 'skprd', 'plprd',
'hrprd', 'cpprd'],
                    description: ' Which target cluster will we use?')

            booleanParam(name: 'COUNTRY_RO', defaultValue: false, description: ' Deploy to
RO?')
            booleanParam(name: 'COUNTRY_PL', defaultValue: false, description: ' Deploy to
PL?')
            booleanParam(name: 'COUNTRY_SK', defaultValue: false, description: ' Deploy to
SK?')
            booleanParam(name: 'COUNTRY_CZ', defaultValue: false, description: ' Deploy to
CZ?')
            booleanParam(name: 'COUNTRY_CP', defaultValue: false, description: ' Deploy to
CP?')
            booleanParam(name: 'COUNTRY_HR', defaultValue: false, description: ' Deploy to
HR?')
            booleanParam(name: 'SHARED', defaultValue: false, description: ' Deploy to
SHARED?')

            string(name: 'SOURCE_BRANCH', defaultValue: 'master', description: ' Branch
name for checkout deploy source files')

            imageTag(name: 'IMAGE', credentialId: 'quay-api-token', organization: 'bet',
image: deploymentServiceName)

            string(name: 'DEPLOY_TICKET', defaultValue: 'none', description: ' Which Jira
DEPLOY TICKET will we use?')

            booleanParam(defaultValue: true, description: 'Should pipeline run functional
tests?', name: 'RUN_FUNCTIONAL_TESTS')
        }

        options {
            timeout(time: 20, unit: 'MINUTES')
            timestamps()
            buildDiscarder(logRotator(numToKeepStr: '10'))
            skipDefaultCheckout()
        }
```

```
        stages {
            stage(' Check Jira deploy ticket') {
                when {
                    expression {
                        return jiraCheckDeployTicket.requiresDeployTicketProd(params.
TARGET_CLUSTER)
                    }
                }
                steps {
                    script {
                        if (jiraCheckDeployTicket.isEmpty(params.DEPLOY_TICKET)) {
                            if (!jiraCheckDeployTicket.isInValidState(params.
DEPLOY_TICKET)) {
                                error "Jira DEPLOY ticket ${params.DEPLOY_TICKET} is not
in valid state - deploy most likely not approved by Release Management!"
                            }
                        } else {
                            error "You need to provide jira DEPLOY ticket ID to deploy to
${params.TARGET_CLUSTER} environment!"
                        }
                    }
                }
            }

            stage(' Init') {
                steps {
                    script {
                        targetCluster = params.TARGET_CLUSTER

                        if ("${params.SHARED}".toBoolean()) {
                            targetNamespaces << "bet-shared"
                        }
                        if ("${params.COUNTRY_CZ}".toBoolean()) {
                            targetNamespaces << "bet-cz"
                        }
                        if ("${params.COUNTRY_RO}".toBoolean()) {
                            targetNamespaces << "bet-ro"
                        }
                        if ("${params.COUNTRY_SK}".toBoolean()) {
                            targetNamespaces << "bet-sk"
                        }
                        if ("${params.COUNTRY_PL}".toBoolean()) {
                            targetNamespaces << "bet-pl"
                        }
                        if ("${params.COUNTRY_CP}".toBoolean()) {
                            targetNamespaces << "bet-cp"
                        }
                        if ("${params.COUNTRY_HR}".toBoolean()) {
                            targetNamespaces << "bet-hr"
                        }

                        if(targetNamespaces.size() == 0) {
                            echo "No country was selected for deploy. Aborting build."
                            currentBuild.result = 'ABORTED'
                            error('No country was selected for deploy.')
                        } else if(!nonProdClusters.contains(targetCluster) &&
targetNamespaces.size() > 1) {
                            echo "Cannot deploy multiple PROD countries at once. Aborting
```

```
build."
                            currentBuild.result = 'ABORTED'
                            error('Cannot deploy multiple PROD countries at once. Only one
per run.')
                        }
                        echo "countries to update: ${targetNamespaces}"
                    }
                }
            }

            stage(' Clean & Checkout Helm chart') {
                steps {
                    deleteDir()
                    echo "Checking out Helm chart sources"
                    checkout scm: [$class: 'GitSCM',
                                   userRemoteConfigs: [[url: gitURL, credentialsId:
'bitbucket-global']],
                                   branches: [[name: params.SOURCE_BRANCH]],
                                   extensions: [[$class: 'SparseCheckoutPaths',
sparseCheckoutPaths: [[$class: 'SparseCheckoutPath', path: 'src/main/helm']]]]
                    ], poll: false
                    //move helm to ./
                    sh "mv src/main/helm/* ./"
                    echo "Setting up OCP clusters access configuration."
                    //setup kubeconfig from vault values
                    withVault(vaultSecrets: secrets) {
                        sh "echo $BET_K8S_CONFIG | base64 -d  > ./k8sconfig"
                    }
                }
            }

            stage(' Upgrade release via Helm') {
                steps {
                    script {
                        targetNamespaces.each { namespace ->
                                  stage(" Deploy ${namespace.substring(4)}") {
                                echo "Going to helm upgrade namespace: ${namespace}
/${params.TARGET_CLUSTER}"
                                sh "kubectl config use-context ${namespace}/${params.
TARGET_CLUSTER}"

                                VALUES_FILE = "./values/${namespace.substring(4)}/${params.
TARGET_CLUSTER}-values.yaml"
                                def datas = readYaml(file: "${VALUES_FILE}")
                                if (datas.serviceName == null) {
                                  datas = readYaml(file: "./${helmChartName}/values.yaml")
                                }
                                deploymentServiceName = datas.serviceName ?:
deploymentServiceName
                                sh "helm upgrade ${deploymentServiceName} .
/${helmChartName} -f ${VALUES_FILE} --install --set image.tag=${params.IMAGE} --wait"
                                  }
                        }
                    }
                }
            }

            stage(' Functional tests') {
                when {
```

```
                    expression {
                        return params.RUN_FUNCTIONAL_TESTS
                    }
                }
                steps {
                    script {
                        targetNamespaces.each { namespace ->
                            stage(" Testing ${namespace.substring(4)}") {
                                def testsRun = build job: 'Betting/deployment/Automated
tests/_CD pipeline tests', parameters: [
                                        string(name: 'TARGET_ENV', value: params.
TARGET_CLUSTER.toUpperCase()),
                                        string(name: 'TEST_SUITE', value: "${testSuite}"),
                                        string(name: 'COUNTRY', value: "${namespace.substring
(4).toUpperCase()}")
                                ], propagate: false, wait: true
                                if (!("SUCCESS").equalsIgnoreCase(testsRun.result)) {
                                    sh "helm rollback ${deploymentServiceName}"
                                    error "Functional tests failed - rolled back previous
release."
                                }
                            }
                        }
                    }
                }
            }

            stage(' Mark Jira ticket as Deployed') {
                when {
                    expression {
                        return false;// jiraCheckDeployTicket.requiresDeployTicketProd
(params.TARGET_CLUSTER)
                    }
                }
                steps {
                    script {
                        if (jiraCheckDeployTicket.requiresDeployTicketProd(params.
TARGET_CLUSTER)) {
                            if ('stg'.equalsIgnoreCase(params.TARGET_CLUSTER)) {
                                jiraMarkDeployedToStage(params.DEPLOY_TICKET)
                            }
                            if ('prd'.equalsIgnoreCase(params.TARGET_CLUSTER.substring
(2))) {
                                jiraMarkDeployedToProd(params.DEPLOY_TICKET)
                            }
                        }
                    }
                }
            }
        }

        post {
            always {
              cleanWs notFailBuild: true /* cleans up the workspace */
            }
            failure {
                emailext (
                        mimeType: 'text/html',
                        subject: "FAILED DEPLOYMENT: Job '${env.JOB_NAME} [${env.
```

```
BUILD_NUMBER}]'",
                        body: """<p>FAILURE: Deployment job '${env.JOB_NAME} [${env.
BUILD_NUMBER}]':</p>
                            <p>Check console output at &QUOT;<a href='${env.
BUILD_URL}'>${env.JOB_NAME} [${env.BUILD_NUMBER}]</a>&QUOT;</p>""",
                        recipientProviders: [[$class: 'CulpritsRecipientProvider']]
                )
                script {
                    if (jiraCheckDeployTicket.requiresDeployTicketProd(params.
TARGET_CLUSTER)) {
                      wrap([$class: 'BuildUser']) {
                        addJiraComment(ISSUE_ID: params.DEPLOY_TICKET, TEXT: "Deployment
[${env.JOB_NAME} [${env.BUILD_NUMBER}]|${env.BUILD_URL}] executed by [~${BUILD_USER_ID}]
has failed.")
                      }
                    }
                }
            }
        }
    }
}
```

Known Issues

In case that very first helm deployment fails one has to manually uninstall the deployment from CMD line. It is a bug in helm that upon initial install failure all following installs will fail.

Related articles

- How to deploy mongodb with Mongo operator
- How To use Slack Relay
- CD Jenkins in FEG
- 2020-03-16 - regular status
- CI Jenkins in FEG