OpenShift Pipeline Builds

Introduction

Whether you are creating a simple website or a complex web of microservices, use OpenShift Pipelines to build, test, deploy, and promote your applications on OpenShift.

In addition to standard Jenkins Pipeline Syntax, the OpenShift Jenkins image provides the OpenShift Domain Specific Language (DSL) (through the OpenShift Jenkins Client Plug-in), which aims to provide a readable, concise, comprehensive, and fluent syntax for rich interactions with an OpenShift API server, allowing for even more control over the build, deployment, and promotion of applications on your OpenShift cluster.

This example demonstrates how to create an OpenShift Pipeline that will build, deploy, and verify a **Node.js/MongoDB** application using the **[nodejs-mongodb.json](https://github.com/sclorg/nodejs-ex/blob/master/openshift/templates/nodejs-mongodb.json)** template.

Creating the Jenkins Master

To create the Jenkins master, run:

$ oc project <project\_name>

$ oc new-app jenkins-ephemeral

|  |  |
| --- | --- |
|  | Select the project that you want to use or create a new project with **oc new-project <project\_name>**. |
|  | If you want to use persistent storage, use **jenkins-persistent** instead. |
|  | If Jenkins auto-provisioning is enabled on your cluster, and you do not need to make any customizations to the Jenkins master, you can skip the previous step. |

The Pipeline Build Configuration

Now that the Jenkins master is up and running, create a BuildConfig that employs the Jenkins pipeline strategy to build, deploy, and scale the **Node.js/MongoDB** example application.

Create a file named **nodejs-sample-pipeline.yaml** with the following content:

**kind: "BuildConfig"**

**apiVersion: "v1"**

**metadata:**

**name: "nodejs-sample-pipeline"**

**spec:**

**strategy:**

**jenkinsPipelineStrategy:**

**jenkinsfile: <pipeline content from below>**

**type: JenkinsPipeline**

For more information about configuring the Pipeline Build Strategy, see [Pipeline Strategy Options](https://docs.openshift.com/online/dev_guide/builds/build_strategies.html#pipeline-strategy-options).

The Jenkinsfile

Once you create a BuildConfig with a **jenkinsPipelineStrategy**, tell the pipeline what to do by using an inline **jenkinsfile**. This example does not set up a Git repository for the application.

The following **jenkinsfile** content is written in Groovy using the OpenShift DSL. For this example, include inline content in the BuildConfig using the [YAML Literal Style](http://www.yaml.org/spec/1.2/spec.html#id2795688), though including a **jenkinsfile** in your source repository is the preferred method.

The completed BuildConfig can be viewed in the OpenShift Origin repository in the examples directory,**[nodejs-sample-pipeline.yaml](https://github.com/openshift/origin/tree/master/examples/jenkins/pipeline/nodejs-sample-pipeline.yaml)**.

**def templatePath = 'https://raw.githubusercontent.com/openshift/nodejs-ex/master/openshift/templates/nodejs-mongodb.json'**

**def templateName = 'nodejs-mongodb-example'**

**pipeline {**

**agent {**

**node {**

**label 'nodejs'**

**}**

**}**

**options {**

**timeout(time: 20, unit: 'MINUTES')**

**}**

**stages {**

**stage('preamble') {**

**steps {**

**script {**

**openshift.withCluster() {**

**openshift.withProject() {**

**echo "Using project: ${openshift.project()}"**

**}**

**}**

**}**

**}**

**}**

**stage('cleanup') {**

**steps {**

**script {**

**openshift.withCluster() {**

**openshift.withProject() {**

**openshift.selector("all", [ template : templateName ]).delete()**

**if (openshift.selector("secrets", templateName).exists()) {**

**openshift.selector("secrets", templateName).delete()**

**}**

**}**

**}**

**}**

**}**

**}**

**stage('create') {**

**steps {**

**script {**

**openshift.withCluster() {**

**openshift.withProject() {**

**openshift.newApp(templatePath)**

**}**

**}**

**}**

**}**

**}**

**stage('build') {**

**steps {**

**script {**

**openshift.withCluster() {**

**openshift.withProject() {**

**def builds = openshift.selector("bc", templateName).related('builds')**

**timeout(5) {**

**builds.untilEach(1) {**

**return (it.object().status.phase == "Complete")**

**}**

**}**

**}**

**}**

**}**

**}**

**}**

**stage('deploy') {**

**steps {**

**script {**

**openshift.withCluster() {**

**openshift.withProject() {**

**def rm = openshift.selector("dc", templateName).rollout().latest()**

**timeout(5) {**

**openshift.selector("dc", templateName).related('pods').untilEach(1) {**

**return (it.object().status.phase == "Running")**

**}**

**}**

**}**

**}**

**}**

**}**

**}**

**stage('tag') {**

**steps {**

**script {**

**openshift.withCluster() {**

**openshift.withProject() {**

**openshift.tag("${templateName}:latest", "${templateName}-staging:latest")**

**}**

**}**

**}**

**}**

**}**

**}**

**}**

|  |  |
| --- | --- |
|  | Path of the template to use. |
|  | Name of the template that will be created. |
|  | Spin up a **node.js** slave pod on which to run this build. |
|  | Set a timeout of 20 minutes for this pipeline. |
|  | Delete everything with this template label. |
|  | Delete any secrets with this template label. |
|  | Create a new application from the **templatePath**. |
|  | Wait up to five minutes for the build to complete. |
|  | Wait up to five minutes for the deployment to complete. |
|  | If everything else succeeded, tag the **$ {templateName}:latest** image as **$ {templateName}-staging:latest**. A pipeline BuildConfig for the staging environment can watch for the **$ {templateName}-staging:latest** image to change and then deploy it to the staging environment. |
|  | The previous example was written using the **declarative pipeline** style, but the older **scripted pipeline**style is also supported. |

Creating the Pipeline

You can create the BuildConfig in your OpenShift cluster by running:

**$ oc create -f nodejs-sample-pipeline.yaml**

If you do not want to create your own file, you can use the sample from the Origin repository by running:

**$ oc create -f https://raw.githubusercontent.com/openshift/origin/master/examples/jenkins/pipeline/nodejs-sample-pipeline.yaml**

For more information about the OpenShift DSL syntax used here, see [OpenShift Jenkins Client Plug-in](https://github.com/openshift/jenkins-client-plugin/blob/master/README.md).

Starting the Pipeline

Start the pipeline with the following command:

**$ oc start-build nodejs-sample-pipeline**

|  |  |
| --- | --- |
|  | Alternatively, you can start your pipeline with the OpenShift Web Console by navigating to the Builds → Pipeline section and clicking **Start Pipeline**, or by visiting the Jenkins Console, navigating to the Pipeline that you created, and clicking **Build Now**. |

Once the pipeline is started, you should see the following actions performed within your project:

* A job instance is created on the Jenkins server.
* A slave pod is launched, if your pipeline requires one.
* The pipeline runs on the slave pod, or the master if no slave is required.
  + Any previously created resources with the **template=nodejs-mongodb-example** label will be deleted.
  + A new application, and all of its associated resources, will be created from the **nodejs-mongodb-example** template.
  + A build will be started using the **nodejs-mongodb-example** BuildConfig.
    - The pipeline will wait until the build has completed to trigger the next stage.
  + A deployment will be started using the **nodejs-mongodb-example** deployment configuration.
    - The pipeline will wait until the deployment has completed to trigger the next stage.
  + If the build and deploy are successful, the **nodejs-mongodb-example:latest** image will be tagged as **nodejs-mongodb-example:stage**.
* The slave pod is deleted, if one was required for the pipeline.

|  |  |
| --- | --- |
|  | The best way to visualize the pipeline execution is by viewing it in the OpenShift Web Console. You can view your pipelines by logging into the web console and navigating to Builds → Pipelines. |