The Diamond CI/CD Platform

Thursday, 10 May 2018   08:27

At some point the majority of these instructions will be converted to an Ansible playbook.

IMPORTANT	The cluster MUST support Docker's Overlay File System if you do not use a agent-specific compatible volume mounted at /var/lib/containers. CentOS 7 with an all default setup will not be compatible. Please see Liu Kai's post at https://linuxer.pro/2017/03/what-is-d_type-and-why-docker-overlayfs-need-it. As a quick rule, if you are using RHEL 7 or CentOS 7, and your filesystem is created by default without specifying an parameter, you can almost be 100% sure that d_type is not turned on on your filesystem.

Preparation

Note: The test Jenkins server is expect to be located at the domain **apps.xchem.diamond.ac.uk** in the project so that the URL to the Jenkins instance is **jenkins-fragalysis-cicd.apps.xchem.diamond.ac.uk**. If it isn't then you will need to either modify the SERVERS variable in the **configure-cicd.py** module (discussed later).

1. Create a project in OpenShift, ideally **fragalysis-cicd** as this is the default project name in the OpenShift templates.
    a. **oc project fragalysis-cicd**
2. Create a "diamond" service account in the project. It will need...
    a. **oc create sa diamond**
    b. **oc adm policy add-role-to-user admin -z diamond**
    c. **oc adm policy add-scc-to-user anyuid -z diamond**
3. Create a "jenkins" user available to OpenShift. Then add system:registry and system:image-builder roles to the new user and add admin role for the project needing access to the registry...
    a. **sudo htpasswd /etc/origin/master/users.htpasswd jenkins**
    Then...
    b. **oc adm policy add-role-to-user system:registry jenkins**
    c. **oc adm policy add-role-to-user system:image-builder jenkins**
    d. **oc adm policy add-role-to-user admin jenkins**
    e. **oc adm policy add-scc-to-user -z jenkins privileged**
4. Create a "fs-input" and "fs-jenkins" PersistentVolumes and the Input claim. You will need a PV for the Django and Neo4j data. This will be the source of the graph and application data, built at regular intervals by the Jenkins jobs form within the Buildah Slave Agent.
    a. Get the DLS repo code with **git clone https://github.com/InformaticsMatters/dls-fragalysis-stack-openshift.git** and you'll find NFS-related material in the **openshift** directory.
    b. **cd dls-fragalysis-stack-openshift/openshift**
    c. **oc process -f fs-input-pv-nfs.yaml | oc create -f -**
    d. **oc process -f fs-jenkins-pv-nfs.yaml | oc create -f -**
    e. **oc process -f fs-input-pvc.yaml | oc create -f -**

Configuring Jenkins (System)

5. **Install the Buildah Agent** image stream
    a. **git clone https://github.com/InformaticsMatters/openshift-jenkins-buildah-slave.git**

    Login to the openshift cluster then...

    b. **cd openshift-jenkins-buildah-slave**
    c. **oc process -f slave.yaml | oc create -f -**
6. **Install Jenkins**.
    a. If using NFS you will need to create a corresponding **ReadWriteOnce** PersistentVolume with a claimRef->name of **jenkins** (done in the previous steps). There's no need for a PVC, when you deploy Jenkins it should then bind to this volume.
    b. Set the **Memory Limit** to at least **4G**i and **Volume Capacity** matching the PV template (normally **10Gi**).
    c. Jenkins comes with an **OpenShift Sample** project. You might want to delete this from the item list to avoid confusion.
7. **Update the Jenkins Plugins**. A number of plugins are likely to be out of date and some updates are needed to progress.
    a. Navigate to **Jenkins -> Manage Jenkins -> Manage Plugins**. If there are any listed navigate to the foot of the page and select **All** and then click the **Download now and install after restart** button. The update can take a while, depending on the number of plugins that need to be updated.
        i. When the plugins have installed (it may stall at the penultimate **OpenShift Sync** plugin) you should restart Jenkins manually by adding **/safeRestart** to the end of the base Jenkins URL and navigating to it.
    b. While you're here you might want to add some useful plugins, like the...
        i. **Slack Notification**
        ii. **Global Slack Notifier**
    c. Ands some that are not-so-useful...
        i. **Green Balls**
8. **Route timeouts (optional)**. This might not be needed if your server response is fast enough but, in order to access the Jenkins configuration pages, you might need to extend the application's Route timeout. Adding the following annotation to the Jenkins **Route** should fix the problem in most cases:

    ```
    metadata:
      annotations:
        haproxy.router.openshift.io/timeout: 120s
    ```
9. **Configure the Buildah Agent**. Navigate to the system configuration (**Jenkins -> Manage Jenkins -> Configure System**) and locate the **Cloud** section. In the **Kubernetes -> Images** block you will find **Kuberneted Pod Template** definitions for each known Agent. If you installed the agent *before* Jenkins then the **buildah-slave** should be there. If not then you will need to restart Jenkins by adding **/safeRestart** to the end of the base Jenkins URL and navigating to it.
    a. In the **Containers** block..
        i. expand the **Advanced...** section and select **Run in privileged mode**
    b. In the **Volumes** block add two volumes
        i. A **Persistent Volume Claim** with the Claim Name set to the volume housing the build data files for Django and Neo4J (this will be **fs-input-claim** if you followed the earlier steps). Mark it as **Read Only** for safety and set the Mount path to **/fragalysis**
        ii. An **empty-dir** volume with a Mount Path of **/var/lib/containers**
    c. Limit the number of concurrent agents by setting **Max number of instances** to **1**.

   d. To improve build speed you might want to keep the agent around between jobs. You can do this by setting the **Time in minutes to retain slave when idle**. To allow the image to clean itself out every now and again you should make suer this period is much less than an hour, say **15** minutes. There may be some advantage to letting the agent die (in order to allow builder's disk space to be cleaned up with the use of a new image).

   Click Save at the bottom of the screen to save your changes.

10. **Configure credentials**.
    a. You will need to create a **Secret Text** credential to be used for access to the external Docker hub. It's a password suitable for the Fragalysis Stack job. At the time of writing we used Alan's password and the credential should be called **abcDockerPassword**. The password should match the chosen DOCKER_USER in the Fragalysis Stack Jenkinsfile.
       i. The credential **Scope** should be G**lobal (Jenkins, nodes, items, all child items, etc)**.
    b. You need a **Secret Text** credential in Jenkins for the integration token (see below) where you'll need to provide an **Integration Token Credential ID** property using the Slack Jenkins application's **Integration Token**. Suggest your credential is called **slackJenkinsIntegrationToken**.
       i. The credential **Scope** should be G**lobal (Jenkins, nodes, items, all child items, etc)**.
11. **Configure Global Slack Notifier Settings**. You will need the provide a **Base URL** (typically something like **https://informaticsmatters.slack.com/services/hooks/jenkins-ci/**) and the **Integration Token Credential ID** created .above
    a. You can use the #general channel to test the connection. Individual jobs will actually post to their own job-specific channels.
    b. You will need to create the channels you expect to be using in the CI/CD process before you run the Jobs. You are advised to create *private* channels so that only invited individuals can see the automated build activity for a given channel/job. So, create the following channels: -
       i. **cicd-dls-graphimage**
       ii. **cicd-dls-webimage**
       iii. **cicd-dls-backupimage**
12. **Get the Jenkins User and API token**. In order to backup and restore the Jenkins jobs using the **configure-cicd.py** (discussed later) you will need to setup the following environment variables using the Jenkins **User ID** and **API Token. the** user ID and API token can be obtained from your Jenkins login. Click the name (upper-right corner), click Configure (left-side menu) and then click 'Show API Token'. Once done set the environment variables: -
    a. **FRAG_CICD_USER** (likely to be something like **jenkins-admin**)
    b. **FRAG_CICD_TOKEN**

Installing the Fragalysis Application suite

You will need to provision NFS volumes and define Volumes in OpenShift. The templates and related files are all contained in the DLS repository.

13. **Get the DLS repo code.** **git clone https://github.com/InformaticsMatters/dls-fragalysis-stack-openshift.git** and you'll find NFS-related material in the **openshift** directory.
14. **Create PVs and Deploy**. There's a **mkpvs.sh** script that makes the NFS volumes and a **deploy.sh** script to create the Persistent Volume Claims and then launch the application (using templates in <root>/templates).

Configuring Jenkins (Jobs)

If you are re-creating the jobs you should be able to run the **configure-cicd.py** module in the **dls-fragalysis-stack-open-shift** repository to create the jobs: -

```
$ configure-cicd.py set test
```

To create the jobs "from scratch" navigate to the main page in Jenkins and create Jobs that will build the Neo4J and stack (Django) service containers: -

15. **Create jobs for the Fragalysis Stack**. The jobs should have the **Do not allow concurrent builds** option set and you should **Discard old builds** (keeping builds for 5 days and up to 25 builds). You will need to create the following Pipeline Jobs: -
    a. "**Fragalysis**".
        i. This **Pipeline** build will poll **https://github.com/xchem/fragalysis.git** using the Script Path **Jenkinsfile**.
    b. "**Fragalysis Frontend**".
        i. This **Pipeline** build will poll **https://github.com/xchem/fragalysis-frontend.git** using the Script Path **Jenkinsfile**.
    c. "**Fragalysis Backend**".
        i. This **Pipeline** build will poll **https://github.com/xchem/fragalysis-backend.git** using the Script Path **Jenkinsfile**.
        ii. The Job should also **Build after other projects are built** by watching the **Fragalysis** Job.
    d. "**Fragalsys Stack**".
        i. This **Pipeline** build will poll **https://github.com/xchem/fragalysis-stack.git** using the Script Path **Jenkinsfile**.
        ii. The Job should also **Build after other projects are built** by watching the **Fragalysis Backend** and **Fragalysis Frontend** Jobs.

16. **Create a "Graph Image" Job**. It should be a **Pipeline** Job, then click **OK**.
    a. In the **General** section
        i. select **Discard old builds** (this will have Disk space on the Jenkins server). You might want to keep jobs for **28** days and only **28** builds.
        ii. Select **Do not allow concurrent builds**
    b. In the **Build Triggers** section set a **Build periodically** trigger (you can always run the build manually).
        i. A daily build at 2:01am (UTC) is defined with the **Schedule** value of **1 2 * * ***.
        ii. Also click the **Trigger builds remotely (e.g., from scripts)** option and provide man **Authentication Token**. Anthony can then trigger builds using the provided URL.
    c. In the Pipeline section select **Pipeline script from SCM**.
        i. Set the **SCM** section that appears set this to **Git**.
        ii. Define the repository (**https://github.com/InformaticsMatters/dls-fragalysis-stack-openshift.git**). It's a public repo so you won't need credentials.
        iii. Set the **Script Path** to **images/graph/Jenkinsfile**
    d. Click **Save**
17. **Create a "Web Image" Job**.
    a. It should be a **Pipeline** Job. As this Job is very similar to the previous one you can use the **Copy from** facility (a the bottom of the page) to save some time by starting to type the name of the existing .job (i.e. Graph Image).
    b. In the **Build Triggers** section…

        i. Set **Build after other projects are built** and watch **Fragalysis Stack**.
        ii. Also click the **Trigger builds remotely (e.g., from scripts)** option and provide man **Authentication Token**.
    c. In the Pipeline section select **Pipeline script from SCM**.
        i. Set the **Script Path** to **images/web/Jenkinsfile**

18. **Create a "Backup Image" Job**.
    a. It should be a **Pipeline** Job that **Discards old builds** with **Max # of builds to keep** of 3 and **Do not allow concurrent builds**.
    b. In the Build Triggers section: -
        i. Poll SCM every 10 minutes (**H/10** * * * *)
    c. In the Pipeline section
        i. Define the repository (**https://github.com/InformaticsMatters/dls-fragalysis-stack-openshift.git**). It's a public repo so you won't need credentials.
        ii. Add **Additional Behaviours** and set **Polling ignores commits in certain paths** and set Included Regions to **images/backup/.+**
        iii. Set the **Script Path** to **images/backup/Jenkinsfile**

Archive the Jobs (Test Server)

You can use the **configure-cicd.py** utility in the **dls-fragalysis-stack-openshift** repository to keep a backup off the jobs and then recreate them.

If you have setup the **FRAG_CICD_USER** and **FRAG_CICD_TOKEN** environment variables you can extract the builds from jenkins with the following command (run from the **jenkins** subdirectory of the repo): -

```
$ configure-cicd-py get test
```

And then recreate them with: -

```
    $ configure-cicd-py set test
```