Table of Contents

OpenShift

OpenShift HowTo's

API

BDD Stack

GIT

Jenkins

Jenkins Pipelines

What is OpenShift?

OpenShift is a platform to help you develop and deploy applications to one or more hosts. These can be public facing web applications, or backend applications, including micro services or databases. Applications can be implemented in any programming language you choose. The only requirement is that the application can run within a container.

In terms of cloud service computing models, OpenShift implements the functionality of both a Platform as a Service (PaaS) and a Container as a Service (CaaS).

Using OpenShift as a CaaS, you can bring a pre-existing container image built to the OpenShift Container Initiative (OCI) Image Specification (image-spec) and deploy it.

The PaaS capabilities of OpenShift build on top of the ability to deploy a container image, by providing a way for you to build in OpenShift your own container image direct from your application source code and have it deployed.

The application source code can include a `Dockerfile` with instructions to build a container image. Or, you can use a Source-to-Image (S2I) builder, which takes your application source code and converts it into a container image for you, without you needing to know how to write instructions for building a container image.

OpenShift vs Kubernetes

Kubernetes implements a system for automating deployment, scaling and management of containerized applications. It is often referred to as being a Container as a Service (CaaS).

Kubernetes alone does not provide any support for building the container image it runs. You need to run a build tool to create your application image on a separate system and push that to an image registry from which it can be deployed. This is because CaaS focuses on just running containers.

OpenShift builds on top of Kubernetes to implement a Platform as Service (PaaS) environment which is more friendly to developers, as well as provide the additional tools and services needed by operations to implement a comprehensive container application platform.

OpenShift Books

The following free eBooks are available on OpenShift.

- OpenShift for Developers - Grant Shipley & Graham Dumpleton
- DevOps with OpenShift - Stefani Picozzi, Mike Hepburn & Noel O'Connor

Other book titles available are as follows.

- OpenShift in Action - Jamie Duncan & John Osborne

Run OpenShift on your local system

To quickly start up an OpenShift cluster locally inside of a virtual machine (VM), you can use Minishift. This isn't an OpenShift distribution, but a tool which you can run to create a minimal VM which includes a container service. Minishift then downloads and launchs a pre-formatted container image containing OpenShift.

Running Minishift requires a hypervisor to run the VM containing OpenShift. Depending on your host operating system, you have the choice of the following hypervisors:

- macOS: xhyve (default), VirtualBox
- GNU/Linux: KVM (default), VirtualBox
- Windows: Hyper-V (default), VirtualBox

To download the latest Minishift release and view any release notes, visit the Minishift releases page. Before using Minishift, ensure you check out the installation instructions for any pre−requisites your system must satisfy.

Minishift is based on the upstream OpenShift Origin project. If you want to use the exact same version of OpenShift as provided by Red Hat via a subscription to OpenShift Container Platform, you can sign up to the free Red Hat Developers program and download the Red Hat Container Development Kit.

User ID within a running POD

When you deploy an application it will appear that it is running as a random user ID, overriding what user ID the image itself may specify that it should run as.

The user ID isn't actually entirely random, but is an assigned user ID which is unique to your project. In fact, your project is assigned a range of user IDs that applications can be run as. The set of user IDs will not overlap with other projects. You can see what range is assigned to a project by running `oc describe` on the project.

Within the output of `oc describe` for the project, the set of annotations recorded against the project show the assigned range of user IDs.

```
openshift.io/sa.scc.uid-range=1008050000/10000
```

By default, any image you deploy to a project will be run as the first user ID in the range assigned to the project. If necessary, you can define an alternate user ID within the range to be used in the deployment configuration for an application. Any attempt to specify that an application should run as a user ID outside of the range will fail.

The purpose of assigning each project a distinct range of user IDs is so that in a multitenant environment, applications from different projects never run as the same user ID. When using persistent storage, any files created by applications will also have different ownership in the file system.

Running processes for applications as different user IDs means that if a security vulnerability were ever discovered in the underlying container runtime, and an application were able to break out of the container to the host, they would not be able to interact with processes owned by other users, or from other applications, in other projects.

Use of assigned user IDs is a part of the multi layer security strategy employed by OpenShift to reduce risks were an application or the container runtime compromised.

Being forced to run as an arbitrary user ID does mean that some container images may not run out of the box in OpenShift. This will be the case where images do not adopt security best practices and need to be run as the `root` user ID even though they have no actual requirement to run as `root`. Even an image which has been setup to run as a fixed user ID which isn't `root` may not work.

To ensure container images are portable and will work with container deployment platforms which enforce additional separation between applications by overriding the user ID, images and the applications they contain, should be designed to be able to be run as an arbitrary user ID.

Author: Matthew J. Robson @ RedHat

Container Native Storage

Documentation: https://access.redhat.com/documentation/en-us/red_hat_gluster_storage/3.3/html/container-native_storage_for_openshift_container_platform/

What is CNS?

CNS is realized by containerizing GlusterFS in a Docker container. This is deployed on a running OpenShift or Kubernetes cluster using native container orchestration provided by the platform. To provide integration with the storage provisioning framework an additional management component is added to GlusterFS called heketi. It serves as an API and CLI front-end for storage lifecyle operations. In addition it supports multiple deployment scenarios. Heketi runs containerized along side the GlusterFS pods on OpenShift or Kubernetes cluster.

- OpenShift <==> Heketi <==> Gluster
- CNS: Provides dynamic persistent storage for OpenShift with Gluster in a hyper-converged fashion
- Heketi: The high-level service interface to gluster to manage the lifecycle of volumes in one or more Gluster clusters.

Container Native Storage (CNS) vs Container Ready Storage (CRS) vs Gluster (RHGS)

- Container-Native Storage (CNS) runs gluster in a containerized form inside the OpenShift gluster. If provides dynamic provisioning capabilities which enable end-users to create their own Persistent Volumes backed by Gluster volumes. This is done without having to pre-create the volumes or have any knowledge of Gluster. The volume will be created as requested when the claim request comes in, and the volume will be sized exactly as requested.
- Container-Ready Storage (CRS) is deployed as a stand-alone Red Hat Gluster Storage cluster outside of OpenShift. CRS also provides the same type of dynamic provisioning.

- Traditional Gluster which also runs as a stand-alone cluster outside OpenShift does not provide any dynamic provisioning. This is what is currently backing existing Persistent Volumes in OpenShift.
- CNS and CRS support two storage types
  - BLOCK (gluster-block)
  - FILE (gluster-file)
- Traditional Gluster supports FILE storage only

CNS BLOCK vs CNS FILE

Access Modes:

- ReadWriteOnce (RWO) – the volume can be mounted as read-write by a single node
- ReadOnlyMany (ROX) – the volume can be mounted read-only by many nodes
- ReadWriteMany (RWX) – the volume can be mounted as read-write by many nodes

---

- Block storage allows the creation of high performance individual storage units as iSCSI targets. Unlike the traditional file storage capability that glusterfs supports, each storage volume/block device can be treated as an independent disk drive, so that each storage volume/block device can support an individual file system.
- Block storage supports and guarantees RWO access and

Gluster Block Design: https://github.com/gluster/gluster-kubernetes/blob/master/docs/design/gluster-block-provisioning.md

- FILE supports all access types but can not guarantee RWO access to a single client (RWX, RWO, ROX)

FAQ

When to use BLOCK storage?

- Access is guaranteed RWO
- for metrics, logging, jenkins data
- non-critical, non-production data
- Deployment strategy "RECREATE"

- Single Mount

When to use FILE storage?

- Access RWX is required
- Any deployment strategy
- Multiple PODs mounting and writing
- If the data stored on PV is production data

How do I select file or block?

- Each type of storage has an associated storage class
- The storage class provides the mechanisms for how storage of that type is dynamically provisioned
- If you want a file backed PV, you select the file storage class
- If you want a block backed PV, you select the block storage class
- If you want an existing traditional gluster PV, you select the manual-storage class

Example Storage Class Object:

```
apiVersion: storage.k8s.io/v1beta1
kind: StorageClass
metadata:
  name: gluster-file
provisioner: kubernetes.io/glusterfs
parameters:
  resturl: "http://<url>"
  restuser: "<user>"
  volumetype: "replicate:3"
  clusterid: "<cluster>"
  secretNamespace: "default"
  secretName: "heketi-secret"
```

How to convert an existing Traditional Gluster PVC to CNS PVC manually?

- Dynamically create a CNS PVC
- Add CNS PVC to deployment (can be done via GUI, not this will trigger a re-deploy)
- OC RSH to POD with both PV types (CSR and Traditional Gluster)
- Copy data (use cp or tar, don't forget the -p option to preserve dates, ownership, etc)

- Change the deploy-config (edit yaml) to connect to CNS PV only
- Verify
- Delete Legacy Gluster PVC

What is the reclaim policy and what does it mean?

- Default reclaim policy of CNS is **Delete**
- If you delete your PVC, it will automatically delete the PV and the gluster volumes backing it. All data will be list

Can I change the reclaim policy?

Policy Options:

- Retain – manual reclamation
- Recycle – basic scrub
- Delete – associated storage asset (gluster volume) is deleted

---

- It can be manually changes on a case by case basis

```
oc patch pv <pv_name> -p '{"spec":{"persistentVolumeReclaimPolicy":"Retain"}}'
```

Should I manually delete a PV when I no longer need it?

- Never manually delete a PV
- Always delete the PVC you created which will in turn delete the PV and all associated objects

What are the quotas for CNS PV?

- Gluster-File: 10pv and 200Gi
- Gluster-Block: 5pv and 200Gi

What does a PVC object look like?

```
kind: PersistentVolumeClaim
apiVersion: v1
metadata:
  name: dynamicclaim
```

```
    annotations:
      volume.beta.kubernetes.io/storage-class: <storage_class_name>
  spec:
    accessModes:
      - ReadWriteMany
    resources:
      requests:
        storage: 1Gi
```

Author: Shea Stewart @ Artic

Purpose

During regular maintenance of a node, which could include (OS) operating system or application (ie. OpenShift) upgrade tasks, a procedure is undertaken with evacuates the node of all running pods. Depending on the application deployment, this process may affect running applications. This document outlines a few key details as it pertains to running applications on an OpenShift node and strategies to ensure application uptime.

TL'DR (Too long;didn't read)

Planned maintenance activities will cause disruption to your application if appropriate action is not taken. Pod evictions will not wait for the application to be fully up and running. To reduce or remove impact;

- Increase replica counts > 1
  - This technique must support the application availability architecture
  - Pods will try to "spread out" across nodes as much as possible by default
- Utilize `PodDisruptionBudget` on OCP ≥ 3.6
- Increase deployments and replicate at the application layer where appropriate
  - More appropriate for applications such as elasticsearch or databsases
  - Use Pod Affinity/Anti-affinity rules when appropriate

Evacuation Process

An administrative command `oc adm drain` or `kubectl drain` is used to peform the following tasks:

- Set a desired node to `unschedulable` , preventing new workloads from arriving on the node
- Evactuate pods with **graceful termination** by restarting on another node prior to termination on the existing node
  - Timeout periods apply before a pod is forcefully terminated
  - Some pods are never terminated gracefully based on their schedule type, such as daemonsets

    - Containers using `emptyDir` for storage will lose any stored data and should be fully architected as empheral pods

What Graceful Termination Is

Graceful termination at a glance:

- Upon termination, the pod is set to shut down
    - Shows up as "Terminating"
    - Executes any `preStop` hooks that may be configured
- The replication controller starts another pod on a `schedulable` node

What Graceful Termination Is NOT

Graceful termination, despite the name, does not take into account the intelligence of deployment stragtegies or deployments in general. The termination is not waiting for the application to be running (if appropriate) before cutting over services, but is simply the act of a replication controller restarting a pod on a new available node. The behaviour here is **exactly** the same as simply deleting your pod.

By example with a sample; notice that the container is still in `ContainerCreating` status even though the previous container has been completely removed.

- The following output is an example of a pod being evicted from a node

```
[root@osjump01 ~]# oc get pods -o wide
NAME              READY     STATUS    RESTARTS   AGE       IP             NODE
jenkins-1-g5npk   0/1       Running   0          1m        10.129.2.237   osa102.d
ev.arctiq.ca
[root@osjump01 ~]$ oc adm drain osa102.dev.arctiq.ca
node "osa102.dev.arctiq.ca" cordoned
pod "jenkins-1-g5npk" evicted
node "osa102.dev.arctiq.ca" drained
[root@osjump01 ~]#
```

- The following is observed during the evactuation process

```
[root@osjump01 ~]# oc get pods -o wide
NAME              READY     STATUS              RESTARTS   AGE       IP
  NODE
jenkins-1-5tcq7   0/1       ContainerCreating   0          9s        <none>
  osa103.dev.arctiq.ca
jenkins-1-g5npk   0/1       Terminating         0          2m        10.129.2.237
```

```
  osa102.dev.arctiq.ca
[root@osjump01 ~]# oc get pods -o wide
NAME              READY     STATUS              RESTARTS   AGE       IP
  NODE
jenkins-1-5tcq7   0/1       ContainerCreating   0          11s       <none>
  osa103.dev.arctiq.ca
[root@osjump01 ~]#
```

Increasing Availability to Survive Planned Maintenance

In order to increase availability of an application during planned maintenance activities;

Increase Replica Count + PodDisruptionBudgets

Increase the replica count of the desired pods and set a PodDisruptionBudget that ensures at least one pod is running at all times.

By example:

- A **node-js** app with 2 replicas on the same node (this was a forced task, typical behaviour would schedule pods across multiple nodes)

```
[root@osjump01 ~]# oc get pods -o wide
NAME               READY     STATUS    RESTARTS   AGE       IP             NODE
node-app-1-crf5f   1/1       Running   0          2m        10.128.2.168   osa101.
dev.arctiq.ca
node-app-1-cw6gm   1/1       Running   0          1m        10.128.2.172   osa101.
dev.arctiq.ca
```

- A PodDisruptionBudget with minimum 1 required pod

```
[root@osjump01 ~]# oc describe PodDisruptionBudget
Name:        node-pdb
Min available:    1
Selector:    app=node-app
Status:
    Allowed disruptions:    1
    Current:            2
    Desired:            1
    Total:            2
Events:                <none>
```

- Draining the node will not let the **Ready** pods to drop below the minimum of **1**

```
[root@osjump01 ~]# oc get pods -o wide
NAME               READY     STATUS              RESTARTS   AGE       IP
   NODE
node-app-1-crf5f   1/1       Terminating         0          3m        10.128.2.168
   osa101.dev.arctiq.ca
node-app-1-cw6gm   1/1       Running             0          2m        10.128.2.172
   osa101.dev.arctiq.ca
node-app-1-k5fpn   0/1       ContainerCreating   0          2s        <none>
   osa103.dev.arctiq.ca
....
[root@osjump01 ~]# oc get pods -o wide
NAME               READY     STATUS        RESTARTS   AGE       IP             NOD
E
node-app-1-9pqtx   1/1       Running       0          5s        10.129.2.245   osa
102.dev.arctiq.ca
node-app-1-crf5f   0/1       Terminating   0          3m        <none>         osa
101.dev.arctiq.ca
node-app-1-cw6gm   1/1       Terminating   0          2m        10.128.2.172   osa
101.dev.arctiq.ca
node-app-1-k5fpn   1/1       Running       0          36s       10.131.2.161   osa
103.dev.arctiq.ca
...
[root@osjump01 ~]# oc get pods -o wide
NAME               READY     STATUS        RESTARTS   AGE       IP             NOD
E
node-app-1-9pqtx   1/1       Running       0          22s       10.129.2.245   osa
102.dev.arctiq.ca
node-app-1-cw6gm   1/1       Terminating   0          3m        10.128.2.172   osa
101.dev.arctiq.ca
node-app-1-k5fpn   1/1       Running       0          53s       10.131.2.161   osa
103.dev.arctiq.ca
...
[root@osjump01 ~]# oc get pods -o wide
NAME               READY     STATUS    RESTARTS   AGE       IP             NODE
node-app-1-9pqtx   1/1       Running   0          2m        10.129.2.245   osa102.
dev.arctiq.ca
node-app-1-k5fpn   1/1       Running   0          2m        10.131.2.161   osa103.
dev.arctiq.ca
```

Utilise Multiple Deployments with Application Replication and Pod Affinity/Anti-Affinity Rules

For applications that don't easily lend themselves to the strategy above, such as:

- Clustered applications

- Clustered Databases (mysql, postresql, cassandra, etc.)
- Any other non-stateful applications

The typical strategy is to have multiple deployments of the application (sometimes as a `StatefulSet`) and to configure each application instance to replicated with their known partners. This level of application clustering must be supported by the application, but should also have additional configurations in place to ensure that they are distributed across nodes such as affinity rules.

Pod Affinity/Anti-Affinity

- Anti-affinity rules
  - Ensure pods are not scheduled on a node that already has a matching pod running
- Affinity rules
  - Ensure pods are only scheduled on nodes that have the matching pods running

Pod affinity allows a **pod** to specify an affinity (or anti-affinity) towards a group of **pods** (for an application's latency requirements, due to security, and so forth) it can be placed with. The node does not have control over the placement. Pod affinity uses labels on nodes and label selectors on pods to create rules for pod placement. Rules can be mandatory (required) or best-effort (preferred).

To ensure higher-availability of applications during planned or unplanned maintenance, it would be recommended (for production environments) that appropriate pod **anti-affinity** configurations be applied to deployments with replicas > 2 or on applications that leverage independent `DeploymentConfigs` or `StatefulSets` with replication at the application level.

References

The following links provide additional reading material on this topic:

OpenShift Specific Documentation

- https://docs.openshift.com/container-platform/3.7/admin_guide/manage_nodes.html#evacuating-pods-on-nodes
- https://docs.openshift.org/latest/admin_guide/manage_nodes.html#evacuating-pods-on-nodes
- https://docs.openshift.com/container-platform/3.7/admin_guide/scheduling/pod_affinity.html

Kubernetes Documentation

- https://kubernetes.io/docs/tasks/administer-cluster/safely-drain-node/
- https://kubernetes.io/docs/concepts/workloads/pods/pod/#termination-of-pods
- https://kubernetes.io/docs/concepts/workloads/pods/disruptions/
- https://kubernetes.io/docs/concepts/workloads/controllers/statefulset/

Checking network connectivity of a pod to an external system

To verify tcp connection w/o requiring any tools installed

```
oc rsh <pod>

timeout 5 bash -c "</dev/tcp/google.ca/443"; echo $?
```

this returns a 0 if connetion to google.ca on port 443 could be established within timeout period, if it returns anything else connection could not be established.

Granting a user access to a project

As the owner of the project, initially you are the only one who can access it and work in it. If you need to collaborate on a project with other users, you can add additional members to the project. When adding a user to the project, they can be added in one of three primary roles.

- `admin` - A project manager. The user will have rights to view any resource in the project and modify any resource in the project except for quota. A user with this role for a project will be able to delete the project.
- `edit` - A user that can modify most objects in a project, but does not have the power to view or modify roles or bindings. A user with this role can create and delete applications in the project.
- `view` - A user who cannot make any modifications, but can see most objects in a project.

To add another user with edit role to the project, so they can create and delete applications, you need to use the `oc adm policy` command. You must be in the project when you run this command.

```
oc adm policy add-role-to-user edit <collaborator>
```

Replace `<collaborator>` with the name of the user as displayed by the `oc whoami` command when run by that user.

To remove a user from a project, run:

```
oc adm policy remove-role-from-user edit <collaborator>
```

To get a list of the users who have access to a project, and in what role, a project manager can run the `oc get rolebindings` command.

Issue:

Deployment fails due to image pull problem.

Solution:

- Ensure that appropriate image puller permissions exist.

```
  oc policy add-role-to-user system:image-puller system:serviceaccount:i<project-n
ame>-dev:default -n <project-name>-tools
```

- You may have to re-tag the image (or redo the build).

```
  oc tag <image>:lates <image:dev>
```

- Verify your image stream settings (Deployments->->edit): Images

Issue:

Container does not start. Events show errors binding to persistent storage. Glusterfs serice and endpoint do net exist.

Solution:

While in your project re-create glusterfs-cluster-app service and endpoint from yaml file with 'oc create -f'

Yaml files for endpoint and service are at

- https://github.com/BCDevOps/openshift-tools/blob/master/resources/glusterfs-cluster-app-endpoints.yml
- https://github.com/BCDevOps/openshift-tools/blob/master/resources/glusterfs-cluster-app-service.yml

Run

- oc create -f https://raw.githubusercontent.com/BCDevOps/openshift-tools/master/resources/glusterfs-cluster-app-service.yml
- oc create -f https://raw.githubusercontent.com/BCDevOps/openshift-tools/master/resources/glusterfs-cluster-app-endpoints.yml to re-create.

Verify:

```
oc get endpoints
oc get svc
```

Both should show a glusterfs-cluster-app entry.

Increasing the size of an existing PVC

Basic steps:

- shut down apps using storage
- backup current storage content (you can use oc login + oc get pods + oc rsync :/
- create a new pvc for under a new name
- update the deployment config and add the new pvc to be mounted under a new name (volume and volumeMounts)
- oc rsh into the pod and verify both mounts are there (or check in the pathfinder gui)
- oc rsh to pod and copy content to new mount
- update the deployment config and remove old mount and mount new increased pvc to old location.
- verify

See write-up by Kuan Fan at

https://github.com/bcgov/gwells/wiki/Increase-PostgreSQL-Database-storage

for a good example for a postgress pvc increase.

Author: Leo Lou (DataBC)

API Management Services.

Current gateway status

In PRODUCTION, here are APIs already published
https://catalogue.data.gov.bc.ca/dataset?tags=API

Service overview

- https://github.com/bcgov/gwa/wiki/Developer-Guide
- https://catalogue.data.gov.bc.ca/dataset/api-gateway-administration

OpenAPI specs

Published https://github.com/bcgov/api-specs

Kong (https://github.com/kong/kong) API gateway

Used for common logic like rate-limit, app2app authentication like apiKeys, token, keycloak oidc.etc Current Kong Cluster version=> kong-ce.0.12.1

Samples

- DataBC demo, this site https://data.gov.bc.ca/ is entirely driven by API https://dbcfeeds.api.gov.bc.ca/ no db backend
- external client production API https://www.workbc.ca/api, WorkBC manage and host their own API but proxy via our gateway for common features like ratelimit, SSL,

Hooks with OCP

- current Kong cluster is running parallel with OCP Kamloops cluster in Zone D plus RRDNS across from Kamloops/Calgary Datacenter
- backend API, you can host your API anywhere, e.g. OCP Kamloops cluster
- Kong Cluster Production is running on a mixed of RHEL VMs and Physical Servers across from Kamloops and Calgary datacenter.

Roadmap and future development,

- OpenAPI Specs is version 3
- UI (GWA), you are welcome to connect with us directly if you need specific features assume we don’t have already, also welcome to contribute, the source code is internal at the moment and we are working with IP office to make it OSS if possible.

Author: Roland Stens

Issue:

BDD validation process fails with "Bootstrap's JavaScript requires jQuery"

The related code is:

```
<script src="https://ajax.googleapis.com/ajax/libs/jquery/2.2.4/jquery.min.js"></s
cript>
<script src="{% static 'gwells/js/bootstrap.min.js' %}"></script>
```

bootstrap.min.js has dependency on jquery.min.js. But there is a racing condition that the load of jquery.min.js is not completed before phantomJs starts to process bootstrap.min.js.

Verification:

```
oc rsh <jenkins-pipeline-pod>
```

run the BDD test locally

```
`gradlew --stacktrace --debug phantomJsTest`
```

Solution:

Download jquery.min.js from the url and store at load. Refer to the local copy.

```
<script src="{% static 'gwells/js/jquery.min.js' %}"></script>
<script src="{% static 'gwells/js/bootstrap.min.js' %}"></script>
```

As a general guideline: Keep all your javascript files local.

Angelikas-MacBook-Pro:docs aehlers$ vi GIT/GitFlow.md

GitFlow

Two main branches:

- Master
- Dev

The Master branch is the main branch of the project and will be in the ready-for-production state. They are on the remote repository (origin). So, whenever you clone the repository on the Master branch, you will have the last stable version of the project, which is very important.

The develop branch reflects all the new features for the next release. When the code inside the dev branch is stable (this means that you have done all changes for the next releases and tested it), you reach the stable point on the dev branch. Then, you can merge the dev branch into Master.

Feature branches

A feature branch is named based on what your feature is about and will exist as long as the feature is in development. Feature branches only exist in local developer repositories; do not push them on the remote repository. When your feature is ready, you can merge your branch feature to develop and delete the branch. Execute the following steps:

1. Go back to the dev branch: $ git checkout dev
2. Merge the branch to dev by creating a new commit object: $ git merge --no-ff featureBranch
3. Delete the branch: $ git branch -d featureBranch
4. Push your changes on the remote dev repository: $ git push origin dev

Hot-fix branches

These kinds of branches are very similar to release branches. It will respond to fixing a critical bug on production. The goal is to quickly x a bug while the other team members can work on their features. For example, your website is tagged as 1.1, and you are still developing the blog feature on the blog branch. You nd a huge bug on the slider inside

the main page, so you work on the release branch to x it as soon as possible. Create a hotfix branch named: $ git checkout -b hotfix-1.1.1 master Fix the bug and merge it to master (after a commit, of course):

```
$ git checkout master
$ git merge --no-ff hotfix-1.1.1
$ git tag -a 1.1.1
Similarly, like the release branch, merge the hot x branch into the current releas
e branch (if it exists) or dev branch. Then delete it:
$ git checkout dev
$ git merge --no-ff hotfix-1.1.1
$ git branch -d hotfix-1.1.1
```

GitHub Deployment Keys

Docs: https://developer.github.com/v3/guides/managing-deploy-keys/#deploy-keys

Example of setup single user with multiple REPOS:

https://gist.github.com/gubatron/d96594d982c5043be6d4

Issue:

What the best practice when creating jenkins pipelines

Solution:

- Pipeline as code (.i.e. Jenkinsfile in repo)
- Work within stages, all non-setup work should be done in stages

```
stage 'build'
//build
stage 'test'
//test
```

- Do material work within a node

By default, the Jenkinsfile script itself runs on the Jenkins master, using a lightweight executor node is expected to use very few resources. Any material work, like cloning code from a Git server or compiling a Java application, should leverage Jenkins distributed builds capability and run an agent node.

```
stage 'build'
node{
    checkout scm
    sh 'mvn clean install'
}
```

- Aquire node within parallel steps

```
parallel 'integration-tests':{
  node('maven'){ ... }
}, 'functional-tests':{
  node('sonar'){ ... }
}
```

- DO NOT Use input within a node block

```
stage 'deployment'
input 'Do you approve deployment?'
node{
  //deploy the things
}
```

- Define timeouts for inputs

```
timeout(time:5, unit:'DAYS') {
  input message:'Approve deployment?', submitter: 'it-ops'
}
```

- Set environment variables locally not globally

```
withEnv(["PATH+MAVEN=${tool 'm3'}/bin"]) {
  sh "mvn clean verify"
}
```

Resource:

- https://reidweb.com/2017/02/01/what-ive-learnt-about-jenkins-pipelines/

Issue:

Setup Jenkins image with OAUTH

Solution:

Step 1:

Deleted ALL the jenkins objects from the config

- Route
- service
- endpoint
- replication controllers
- pvc
- role binding
- service account (Note: Leave the secrets as is)

Via commandline (oc) you can delete most of the objects:

```
oc describe dc jenkins-pipeline-svc|more (look for the label with template=)
```

```
oc get all -l template=<label-id-for-jenkins> -n <namespace>
```

```
oc delete all -l template=<label-id-for-jenkins> -n <namespace>
```

Go to UI and verify all jenkins objects are gone. Remove whats left behind.

Step 2:

Add to project Select Continuous Integration & Deployment Select BC Gov Pathfinder Jenkins (Persistent)

Step 3:

Configure Jenkins

- Add env variable JAVA_OPTS value -XX:MaxMetaspaceSize=512m to jenkins config
- In jenkins
  - Maven Kubernet node (cpu, memory, namespace)
    - Environment Vars (EnvVars) : OPENSHIFT_JENKINS_JVM_ARCH = x86_64
    - Requested Mem 1Gi; Limited Mem 4Gi
    - Request CPU: 300m; Limit CPU: 500m
  - Configure timeouts Build Verification 180 (was 60)

- Re-apply other changes you might have done (like extra env vars, etc)

NOTE:

https://github.com/BCDevOps/openshift-tools/tree/master/provisioning

Issue:

Jenkins marks a build as failed after 15 minutes, but the build succeeds in OpenShift

Solution:

By default the limit on a build is 15 minutes. Adjust the OpenShift build timeout in the Jenkins configuration to allow for a longer build.

Issue:

When logging in to JENKINS following error is displayed:

```
{"error":"server_error","error_description":"The authorization server encountered
an unexpected condition that prevented it from fulfilling the request.","state":"M
TM0MjZlMjctNWI1MS00"}
```

Solution:

Add missing OAUTH annotation to jenkins service account

See https://access.redhat.com/solutions/3250151 for deatils.

To check if the required annotation for OAUTH is configured:

```
oc get sa jenkins -o json
```

To Add the required OAUTH annotation:

```
oc annotate sa jenkins serviceaccounts.openshift.io/oauth-
redirectreference.jenkins='{"kind":"OAuthRedirectReference","apiVersion":"v1","referenc
e":{"kind":"Route","name":"jenkins"}}'
```

Issue: Jenkins Pipelines not starting - Developers using a Chrome extension

Case:

Contractors were using a chrome extension to auto complete and submit jenkins jobs through the browser. Developer renamed a few projects which caused them to recreate others, all with the same name, which caused name collision. Cleanup of this in the end required to completely remove the pipelines and re-create them.

Issue: Jenkins Pipeline not starting - Kubernet plugin upgraded

The Kubernet plugin versio 1.1.3+ causes a problem with OpenShift 3.6. RESOLUTION: Downgrade the plugin to 1.1.3 or lower.

Issue:

Slaves start but cannot communicate with the master

Solution:

Set the Kubernetes Jenkins URL and JNLP fields to an appropriate value. If the values are appropriate, then check that the platform is not having DNS issues - it can get into a state where local services doe not resolve to valid IP addresses

Issue:

Slaves do not start, following text: `Using 64 bit Java since OPENSHIFT_JENKINS_JVM_ARCH is not set

Solution:

Set Jenkins config setting - Maven Kubernetes option, new environment variable OPENSHIFT_JENKINS_JVM_ARCH with value of x86_64

Issue:

Need to archive test results or other files

Solution:

Add an archive stage to your pipeline. Example:

```
// Archive the built artifacts
```

```
archive (allowEmptyArchive: true, includes: 'report/*.html')
```

Even better wrap your archive in a try/catch block:

```
try { <your stage steps> } finally { archiveArtifacts allowEmptyArchive: true, artifacts: 'build/reports/**/*' }
```

Note:

Archive will show only after the job is finished. I.e. if the job waits for input the archive does not show. There is a trick to get around this by copying the files to be archived in a new folder in the workspace.

```
  dir ('BDDreports') {
    sh 'ls -l'
    writeFile file:'ReadMe.txt', text:'BDD Test Results Folder'
    sh 'cp reports/* BDDreports/'
    sh 'ls -l'
  }
```

Issue:

Need to verify if deployment completed before running functional test

Solution:

Add line to Jenkinsfile:

```
            openshiftVerifyDeployment depCfg: '[dev-deploy-config]', namespace: '[
dev-namespace]', replicaCount: 1, verbose: 'false', verifyReplicaCount: 'false'
```

Example:

```
            openshiftVerifyDeployment depCfg: 'platform-dev', namespace: 'devex-pl
atform-dev', replicaCount: 1, verbose: 'false', verifyReplicaCount: 'false'
```

Grant access for Jenkins to dev env:

```
oc policy add-role-to-user view -z system:serviceaccount:[tools-namespace]:jenkins
 -n [dev-namespace]
```

Example:

```
oc policy add-role-to-user view -z system:serviceaccount:devex-platform-tools:jenk
ins -n devex-platform-dev
```

Note:

openshiftVerifyDeployment doesn't really work well with Rolling deployment strategy when verifying` the replica count. This is because with a rolling there is always a pod running. The old Pod is shutdown when new one is available.