Technical Implementation Guide

Couchbase Server
On OpenShift Container Platform 3.9

Prepared for: DevOps

Date: 2018 May 17
Version: 1.0
Table of Contents

Technical Implementation Guide Overview
  Introduction
  Document Purpose
  Audience
  Solution Purpose
  Featured Technologies
  Why Certified Containers?

Technology Solution Overview
  OpenShift Technology Overview
    Key Components
  Couchbase Server Technology Overview
    Key Components

Configuration and Deployment
  Prerequisites
  Deployment Overview
  Couchbase Server Deployment
  Persistent Storage Configuration
  Validation
    Checking the health of the couchbase pod
    Configure Couchbase Server using the Web UI
    Perform a N1QL query from the command line

Troubleshooting
  The couchbase pod isn't running
  The couchbase pod is running, but appears unresponsive
  The couchbase pod is unresponsive when using persistent storage

Conclusion

Resources
  OpenShift
  Couchbase

Contact Info
Technical Implementation Guide Overview

Introduction

Red Hat Connect for Technology Partners is designed for software, hardware and cloud companies looking to test and certify their products on the Red Hat portfolio. By participating in this program, companies gain access to a wide variety of resources ranging from people, information, software, and other tools to help them with the testing and certification process of their products on the Red Hat platform.

Document Purpose

This technical reference guide describes a containerized NoSQL database solution using Couchbase Server in conjunction with Red Hat OpenShift. This guide describes the containerized solution, including core features and functionality, but does not aim to be a comprehensive guide to the application or the underlying platform. Links to more detailed platform documentation are provided at the end of this guide.

NOTE: The solution described in this guide is for demonstration purposes only, and is not supported by Red Hat or the partner.

Audience

This technical reference guide is for DevOps personnel, cloud architects and cloud operators who want to deploy Couchbase Server onto the OpenShift platform. Readers should be familiar with OpenShift, RHEL, container technologies and DevOps tools such as git.

Solution Purpose

The purpose of this solution is to pull, deploy and demonstrate a certified Couchbase Server container image (as published in the Red Hat Container Catalog) on the OpenShift platform. It is assumed that OpenShift is already deployed and that the reader has a valid login with the appropriate permissions.

Featured Technologies

This solution demonstrates a containerized application deployment that leverages the following key technologies:
• Red Hat Enterprise Linux 7 / Atomic Host platforms
• Red Hat OpenShift Container Platform 3.9
• Red Hat Container Catalog
• Couchbase Server certified container image

Why Certified Containers?

With the ever increasing popularity of container technologies, the enterprise has begun to embrace the benefits of containers versus that of conventional virtualization. There remains, however an expressed concern over the security and relative newness of containers compared to virtual machines. All concerns aside, container technology has begun to mature, and container security solutions are exploding onto the scene from numerous vendors to address the need.

Given the portability of containers, the philosophy of “build once, run everywhere” seems to have lessened the emphasis on the runtime components within the container. The layering of the container filesystem lends to the inherent re-use of an image as a base runtime environment for other container images. This eliminates the need for ISVs to “reinvent the wheel” and supply their own runtime environment when assembling a container image. However, as a side effect it can also contribute to a mashup of runtime components from multiple vendors. Thus, containers can be assembled from bits and pieces of numerous different Linux distributions, both community and enterprise alike. That is often the case for container images hosted on popular public channels.

Couple all of this with the possibility of publicly hosted container images being out of date or unmaintained, and it is indeed a cause for concern. New Common Vulnerabilities and Exposures (CVE®) are cropping up all the time for widely used open source software. This not only indicates the need for hosted container images to be scanned for vulnerabilities and kept up to date, but also the need for images to get built from single-sourced, vetted software channels. The Red Hat Container Catalog was designed to address these needs.

ISV container images hosted in the Catalog must go through a certification process before being published. The certification process is outlined below:

• The build file gets reviewed by Red Hat Engineers for best practice adherence
• Approved images get pushed to the Red Hat Connect for Partners registry
• A build of the container image gets triggered
• Successful builds get scanned automatically for vulnerabilities and compliance
After a successful scan is completed and other compliance requirements are met, only then are container images published to the Container Catalog. The overall health of an image is rated by a letter-graded health index (A-F), with grades C and below being delisted from the Container Catalog. This helps to prevent the distribution of insecure or unmaintained container images, and encourages ongoing upkeep of the images.

Technology Solution Overview

OpenShift Technology Overview

Built on Red Hat technologies and Kubernetes, OpenShift Container Platform provides a secure and scalable multi-tenant platform for today’s enterprise-class applications. It also provides integrated application runtimes and libraries.

Key Components

- **Container Technology**
  - **OpenShift** includes docker technology that provides the abstraction for packaging and creating Linux-based, lightweight containers and microservices.

- **Kubernetes**
  - **Kubernetes** manages containerized applications across a set of containers or hosts and provides mechanisms for deployment, maintenance, and application-scaling.
  - A Kubernetes cluster consists of one or more masters acting as the control plane and a set of compute nodes for running pods, which are logical groups of containers.
  - All Kubernetes APIs are supported, as they are incorporated into OpenShift unchanged from upstream.

- **Image Registry**
  - OpenShift provides an integrated Docker registry that adds the ability to provision new image repositories on the fly. This allows users to automatically have a place for their builds to push the resulting images.

- **Flexible UI Options**
  - The OpenShift web console is a user interface accessible from a web browser. Developers can use the web console to visualize, browse, and manage the contents of projects.
- The OpenShift CLI (oc) offers an even greater level of flexibility and control, and eases integration with automation tools such as Ansible.

- **Developers Tools**
  - Many of the most popular development frameworks and databases are provided in the Red Hat Container Catalog.
  - Integrate your own code into base images using Source-to-Image (S2I) builds, or more advanced builds using Jenkins pipelines for CI.
  - OpenShift includes all of the components needed to build and deploy your applications, out of the box.

Couchbase Server Technology Overview

**Key Components**

- **Couchbase OpenShift Container**
  - Certified by Redhat and production ready. Deploy your applications with peace of mind on Couchbase Containers on OpenShift

- **Couchbase Autonomous Operator for OpenShift Container Platform**
  - The Couchbase Autonomous Operator for Kubernetes enables cloud portability and automates operational best practices for deploying and managing the Couchbase Data Platform.
  - Couchbase Autonomous Operator for Kubernetes reduces operational complexity up to 95% by implementing the operational best practices that most efficiently deploy and manage the Couchbase Data Platform.
  - More details can be found in this blog.
  - Documentation regarding the Operator can be found at this link.
- **Couchbase Data Platform**
  - **N1QL Queries**
    - Build and evolve web, mobile, and IoT apps faster. N1QL combines the power and familiarity of SQL with the flexibility and agility of the JSON data model.
  - **Full Text Search (FTS)**
    - Better customer experiences result from smarter, richer applications. Quickly integrate intelligent Full Text Search into your apps.
  - **Built-in Analytics**
    - Create more insightful apps with built-in analytics. Seamlessly integrate application and ad hoc analytical queries on your Couchbase data in real time.
  - **Active-active Global Replication**
    - Ensure always-on high availability, disaster recovery, and flexible global replication with cross datacenter replication (XDCR) that can scale and evolve with your business.
  - **Multi-Dimensional Scaling (MDS)**
    - One size never fits all when it comes to scaling your business. MDS delivers revolutionary distributed architecture providing compute, storage, and processing workload partitioning to meet ever-changing requirements.
  - **Enterprise-grade Security**
    - Compliance and security are fundamental business requirements. Protect your data on the wire and in the cloud. Protect your organization with built-in auditing, role-based access control, and encrypted communications.

### Configuration and Deployment

**Prerequisites**

The following prerequisites are necessary in order to deploy Couchbase Server:

- **OpenShift Container Platform (OCP) v3.9** installed to the [minimal requirements](#)
- **Persistent Volumes** (optional) to allow for persistent storage, including root permissions on the backing storage (eg: `no_root_squash` for NFS)
- The following tools for remote access to OpenShift:
  - `oc binary`
  - `git`
● An OpenShift account with `cluster-admin` permissions (such as `system:admin`) to enable the anyuid security context constraint (see Table 2)
● A Red Hat account for access to the Red Hat Container Catalog

Deployment Overview

The deployment process is split into different sections. Deployment demonstrates creating a project in OpenShift and deploying a basic single node instance of Couchbase Server from a template (set of Kubernetes objects). Persistent Storage Configuration outlines additional steps required in order to use such storage. Validation covers checking the deployed resources and testing the functionality of Couchbase Server. Troubleshooting shows example failure scenarios and solutions.

Couchbase Server Deployment

Couchbase Server is deployed from a template file, which defines the pods and services that together enable a fully-featured database hosted on OpenShift.

Login to your oc host machine (or the master) as root:

```
$ sudo su -
```

Login to your target deployment of OpenShift, replacing `<username>` with that of your OpenShift account:

```
# oc login -u <username>
```

Create a new project:

```
# oc new-project testdb
```

Create the image pull secret for registry.connect.redhat.com. Replace items denoted by `< >` with your Red Hat account info:

```
# oc create secret docker-registry rhcc 
--docker-server=registry.connect.redhat.com 
--docker-username=<username> --docker-password=<password> 
--docker-email=<email>
```

**NOTE:** Any errors with your Red Hat login credentials will prevent the image from pulling, and thus prevent the couchbase-server pod from launching.

Link the newly created secret to the project's default user for image pulling:
# oc secrets link default rhcc --for=pull

Assign the `anyuid` Security Context Constraint to the project's `default` service account (this requires the `cluster-admin` role assigned to your OpenShift account):

```
# oc adm policy add-scc-to-user anyuid -z default
```

Clone the couchbase/kubernetes repository from GitHub:

```
# git clone https://github.com/jsm84/couchbase-kubernetes
```

Change into the repository base directory:

```
# cd kubernetes
```

Deploy the ephemeral storage template into the current project:

```
# oc create -f openshift/couchbase-server-ephemeral.yaml
```

Deploy the persistent storage template into the current project:

```
# oc create -f openshift/couchbase-server-persistent.yaml
```

There are now two templates to choose from in the current project, depending on whether or not persistent storage is desired. To launch Couchbase Server, run either of the following commands:

A. For ephemeral storage (recommended for first-time deployments):

```
# oc new-app couchbase-server-ephemeral
```

B. For persistent storage:

```
# oc new-app couchbase-server-persistent
```

The next section lists additional steps required to use persistent storage. If you’ve chosen ephemeral storage, skip ahead to Validation.

**Persistent Storage Configuration**

In order for pods to access persistent storage, then the `couchbase-server` pod(s) must be granted root permissions on the backing storage volumes. In this guide, we will use NFS for our example storage. In the case of NFS, the share that gets
consumed by the `couchbase-server` pod(s) must have the `no_root_squash` option set (as a side-effect, pods no longer require storage group membership). Since these pods run as root, NFS file/directory ownership would otherwise be squashed to the user and group of `nfsnobody` for all files written to the volume. This prevents certain processes within the container from being able to write to the storage (specifically, those processes owned by the user `couchbase` w/UID 1000).

**NOTE:** It's not considered a best-practice to enable NFS `no_root_squash` on all shares exported to OpenShift, so the remainder of this section focuses on enabling `no_root_squash` on the specific NFS share consumed by Couchbase Server.

You can see which PV is bound to Couchbase Server by checking the output of the following command (highlighted):

```bash
# oc get pvc
NAME   STATUS   VOLUME     CAPACITY  ACCESSMODES  ...  AGE
couchbase Bound  pv02      5Gi      RWO               ...  2m
```

Check with your cluster admin to obtain the name of the NFS share backing the PV. Once obtained, it can be adjusted on the NFS server as follows (NOTE: this next step is performed on the NFS server by an admin):

```bash
# vi /etc/exports
```

An example `/etc/exports` file is shown. Note in this case that the share name matches the PV name (your case may differ), and `no_root_squash` is set accordingly:

```bash
/exports/pv01 *(rw,no_root_squash)
/exports/pv02 *(rw,no_root_squash)
/exports/pv03 *(rw,root_squash)
...
```

Once `/etc/exports` is edited on the NFS server, the shares can be updated with the new settings by running the following command (on the NFS server):

```bash
# exportfs -r
```

Continue on to the Validation section, which shows how to check the pods for readiness.
Validation

In this section, you will check that the couchbase pod is running properly, and then configure the cluster using the web UI. You’ll also run queries against sample data, from both the web UI and from within the container. Connecting to the database from another pod, host or application is possible, but isn’t covered here.

Checking the health of the couchbase pod

The first step in validating the deployment is to check the health status of the pod.

The next command should show the couchbase pod with a Running status:

```
$ oc get pods
NAME         READY STATUS        RESTARTS AGE
couchbase-1-562xt 1/1    Running     0   3m
```

**NOTE:** If you chose to use persistent storage, allow a few seconds for the container to restart after editing the deploymentConfig. If you don’t see the couchbase pod listed, refer to the Troubleshooting section at the end of this guide.

If the couchbase pod is running/ready, you should now be able to access Couchbase Server from a web browser. First, you'll need to discover the public URL. The following command should return the hostname used to access the service:

```
$ oc get route
NAME     HOST/PORT
PATH     SERVICES   PORT TERMINATION WILDCARD
  couchbase  couchbase-testdb.router.default.svc.cluster.local
  couchbase  admin       None
```

If you don’t have wildcard DNS configured for your OpenShift router, then a hosts file entry is required on your local machine in order to resolve the URL. The host entry should map to the physical (or virtual) IP address of the OpenShift router endpoint, eg:

```
172.1.2.3 couchbase-testdb.router.default.svc.cluster.local
```

Configure Couchbase Server using the Web UI

In this stage of validation, you will use the web UI to configure Couchbase Server, load sample data, and run a query against the database using the N1QL syntax.
Paste the previously obtained URL into a web browser, eg:
http://couchbase-testdb.router.default.svc.cluster.local

A successful deployment should bring you to the Couchbase Server login page:

From here, you can continue setting up a new cluster. Select **Setup New Cluster**. Enter the desired cluster name and password, and then select **Next: Accept Terms**:

Check the box to accept the license agreement. If your OpenShift cluster is running on a small test environment (< 8GB RAM per node), then you must proceed to **Configure Disk, Memory, Services** and scale each memory allocation down to fit your memory limitations. Otherwise, select **Finish With Defaults**.

Below is a screenshot of the resource allocation screen:
Once the setup is complete, you will be taken to the Dashboard page. Select the **sample bucket** link, which will present a choice of data samples:

Check the box adjacent to **beer-sample**, and then select **Load Sample Data**.

Once the sample data is loaded, select **Buckets** on the left side column to see active sample data sets (beer-sample should be listed). To perform queries against the data, elect the **Query** link on the left side column, which is where you’ll run a NIQL statement against Couchbase Server:
Enter the following N1QL statement into the Query Editor field:

```sql
SELECT name FROM `beer-sample` WHERE brewery_id = "mishawaka_brewing";
```

Click **Execute** and you should see the following data in the Query Result field:

Perform a N1SQL query from the command line

You can also perform queries from the command line. Return to your previous terminal window and enter the following command (replace the example pod name below with the actual pod name from your cluster):

```
# oc rsh couchbase-1-9t49x
```

You should now be at a `sh-4.2#` prompt within the container. Run the following command to connect to the locally running query engine (replace `<pwd>` with the password provided during cluster setup):

```
# /opt/couchbase/bin/cbq -e localhost:8093 -c Administrator:<pwd>
```
You will now be at a `cbq>` prompt, where you can enter the same SQL statement from the earlier web UI example:

```
cbq> SELECT name FROM 'beer-sample' WHERE brewery_id = "mishawaka_brewing";
```

You should see the same data returned as from the previous web UI query. To exit the database/container session, type `CTRL+D` and then issue the following command:

```
# exit
```

Congrats! You’ve deployed a fully containerized edition of Couchbase Server to the OpenShift platform.

## Troubleshooting

Below are a few different possible scenarios encountered while deploying Couchbase Server to the OpenShift platform. Also, see the [Resources](#) section for any errata pertaining to this guide.

### The `couchbase` pod isn't running

In this scenario, running the `oc get pods` command shows an empty output, meaning that the `couchbase` pod isn't running. Checking the current status of the project shows:

```
# oc status
...
  dc/couchbase deploys istag/couchbase-server:latest
deployment #1 waiting on image or update
```

You can see from the previous command that the `couchbase` deployment is stuck waiting on an image. The templates used to deploy Couchbase Server define an image stream which contains the path to the RHCC image. The following command will provide a detailed description of the image stream and its current status:

```
# oc describe imagestream
...
  ~ importing latest image ...
```

Image streams created via template only probe the specified registry (RHCC) once, and will remain in standby if unable to import an image. An image stream can be triggered to update by creating an image stream tag that points to the RHCC image:
# oc tag \
registry.connect.redhat.com/couchbase/server:latest \
testdb/couchbase-server:latest

Check the image stream details again for any possible errors:

```
# oc describe imagemstream
...
! error: Import failed (InternalError): Internal error occurred: ...
...received unexpected HTTP status: 500 Internal Server Error
```

In the above output, you’ll see that the image had failed to pull from the RHCC registry with **500 Internal Server Error**. For the Red Hat Container Catalog, this usually indicates an issue with the credentials provided in the image pull secret. In such a case, you’ll need to recreate the **rhcc** secret, and link it to the default service account for image pulling. It is easiest to delete the project and start from the beginning of the Deployment section:

```
# oc delete project testdb
```

The **couchbase** pod is running, but appears unresponsive

In this scenario, the **oc get pods** command shows the **couchbase** pod with a status of **Running**:

```
# oc get pods
NAME        READY STATUS    RESTARTS AGE
couchbase-1-9t49x   1/1  Running   0 2h
```

When attempting to connect to the Couchbase Server web UI (see Validation), you get an error similar to the one below:

```javascript
Application is not available
The application is currently not accepting requests at this endpoint. It may not have been started or is still starting.
```

Checking the log output of the pod, shows no errors (there is only one line in the log output stating that Couchbase Server was started):

```
# oc logs couchbase-1-9t49x
Starting Couchbase Server -- Web UI available at http://<ip>:8091
and logs available in /opt/couchbase/var/lib/couchbase/logs
```
However, if you connect to the container and inspect the running processes, you'll find a particular process (PID 1) with an error:

$ ps -ef

<table>
<thead>
<tr>
<th>UID</th>
<th>PID</th>
<th>PPID</th>
<th>C</th>
<th>STIME</th>
<th>TTY</th>
<th>TIME</th>
<th>CMD</th>
</tr>
</thead>
<tbody>
<tr>
<td>1000130+</td>
<td>1</td>
<td>0</td>
<td>0</td>
<td>19:20</td>
<td>?</td>
<td>00:00</td>
<td>runsvdir -P -H</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td>/etc/service log: server: fatal: unable to open supervise/lock: file does not exist runsv couchbase-server: fatal: unable to open supervise/lock:</td>
</tr>
<tr>
<td>1000130+</td>
<td>80</td>
<td>0</td>
<td>0</td>
<td>19:25</td>
<td>?</td>
<td>00:00</td>
<td>/bin/sh</td>
</tr>
<tr>
<td>1000130+</td>
<td>762</td>
<td>1</td>
<td>0</td>
<td>20:21</td>
<td>?</td>
<td>00:00</td>
<td>[runsv]</td>
</tr>
<tr>
<td>1000130+</td>
<td>763</td>
<td>80</td>
<td>0</td>
<td>20:21</td>
<td>?</td>
<td>00:00</td>
<td>ps -ef</td>
</tr>
</tbody>
</table>

If you inspect the **UID** field from the previous command output, you'll notice that the processes are running with **UID 1000130+**. This is due to OpenShift launching the container using the default **restricted** security context constraint. Currently, the Couchbase Server container image requires the **anyuid** security context constraint in order to run properly. Using the **restricted scc** causes the pod to launch with an unprivileged UID that lacks the necessary permissions.

To resolve this, add the **anyuid scc** to the project's **default** service account as shown in [Deployment](#) (doing so requires **cluster-admin** privileges within OpenShift):

```
# oc adm policy add-scc-to-user anyuid -z default
```

You will then need to launch a new instance of the **couchbase** pod. The easiest way is to simply delete the pod, which will then trigger the deployment of a fresh pod instance (replace the pod name in this example with the actual name of your pod):

```
# oc delete pod couchbase-1-9t49x
```

The **couchbase** pod is unresponsive when using persistent storage.

In this scenario, Couchbase Server was launched from the persistent storage template. As in the previous scenario, the pod appears to be running normally, as shown by checking the pod status:

```
# oc get pods

<table>
<thead>
<tr>
<th>NAME</th>
<th>READY</th>
<th>STATUS</th>
<th>RESTARTS</th>
<th>AGE</th>
</tr>
</thead>
<tbody>
<tr>
<td>couchbase-1-hdlpg</td>
<td>1/1</td>
<td>Running</td>
<td>0</td>
<td>2m</td>
</tr>
</tbody>
</table>
```
When attempting to access the web UI, you see the same error page as in the previous scenario:

Application is not available

However, when checking the logs of the current pod, you'll see a number of errors for attempts to change the ownership of certain directories:

```bash
# oc logs couchbase-1-hd1pg
Starting Couchbase Server -- Web UI available at http://<ip>:8091
and logs available in /opt/couchbase/var/lib/couchbase/logs
chown: changing ownership of '/var/lib/couchbase/config':
Operation not permitted
chown: changing ownership of '/var/lib/couchbase/data': Operation not permitted ...
```

In the above log output, var/lib/couchbase/* is a relative path, which actually maps to the storage mount point of /opt/couchbase/var within the container. In this case, this is due to root_squash being set on the backing NFS share for the storage. This needs to be changed to no_root_squash for this particular share. Refer to Persistent Storage Configuration for an example on how to configure the NFS server.

**Conclusion**

Reducing deployment inflexibility while simultaneously increasing the choice of software solutions is a top priority for many companies. Cost, security, reliability and ease of use are often key considerations when enterprise customers evaluate new technology solutions.

OpenShift provides that layer of flexibility while Couchbase Server provides unmatched agility and manageability – as well as unparalleled performance at any scale – to deliver ever-richer and ever more personalized customer experiences.
Resources

OpenShift

- [OpenShift Container Platform 3.9 Documentation](#)
- [OpenShift Container Platform Reference Architectures](#)

Couchbase

- [Couchbase Home](#)
- [Couchbase Blog](#)
- [Couchbase Documentation](#)
- [Technical Implementation Guide Errata](#)
- [Couchbase Server - Red Hat Container Catalog](#)
- [Couchbase Server on Kubernetes - Github](#)

Contact Info

Couchbase:
sales@couchbase.com

Red Hat Connect:
connect@redhat.com