Technical Implementation Guide

Rocket.Chat Container Application
On OpenShift Container Platform 3.7

Prepared for: DevOps

Date: 2018 Mar 16
Version: 1.2

Table of Contents

Reference 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 open source chat solution using Rocket.Chat 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 Rocket.Chat 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 Rocket.Chat 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 3.7
- Red Hat Container Catalog
- Rocket.Chat certified container image
- MongoDB 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.

Rocket.Chat Technology Overview

Key Components

- **Open Source chat alternative**
  - Rocket.Chat is the leading free open source team chat alternative.
- **Enterprise Features**
  - Free audio and video conferencing, guest access, screen sharing, file sharing, LDAP Group Sync, two-factor authentication (2FA), E2E encryption, SSO, dozens of OAuth providers and unlimited users, channels, searches, guests, messages and files.
- **Live Chat**
  - With Rocket.Chat LiveChat you can add real-time chat widgets to any website or mobile app and get more value from your team chat.
- **Real Time Translation**
  - Rocket.Chat uses cutting edge machine learning for automatic real-time message translation between users. The UI has been translated to over thirty seven languages, more than any other team chat service.
- **Customizability**
  - Easily customize Rocket.Chat using integrations, plugins, themes, data importers, powerful APIs, and documentation.

Configuration and Deployment

Prerequisites

The following prerequisites are necessary in order to deploy Rocket.Chat:

- OpenShift Container Platform (OCP) v3.7 installed to the minimal requirements
- Persistent Volumes (optional) to allow for persistent storage
- Wildcard DNS (optional) to resolve application hostnames
- The following tools for remote access to OpenShift:
    - oc binary
    - git
    - docker
- Root access to the OpenShift CLI (oc) host to run docker commands
- Basic user account permissions in OpenShift to create projects and apps
- A Red Hat account for access to the Red Hat Container Catalog

Deployment Overview

The deployment process is split into different sections of this guide. Deployment demonstrates creating a project in OpenShift and deploying the Rocket.Chat application 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 Rocket.Chat. Troubleshooting shows an example failure case and its resolution.

Rocket.Chat Deployment

Rocket.Chat deploys from a template file which defines the pods, services, image streams and route used to create a fully-featured application 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 testapp
```

Login to the Red Hat Container Catalog registry using docker. Replace items in < > with your Red Hat account info:

```
# docker login -u <username> -p <password> -e <email> \
registry.connect.redhat.com
```

Create the image pull secret for registry.connect.redhat.com, using the newly created docker config.json file containing the login credentials:

```
# oc secrets new rhcc .dockerconfigjson=/root/.docker/config.json
```

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

Link the newly created secret to the project's default user for image pulling:

```
# oc secrets link default rhcc --for=pull
```

Clone the Rocket.Chat GitHub repository:

```
# git clone https://github.com/RocketChat/Rocket.Chat
```

Change into the repo directory:

```
# cd Rocket.Chat
```

Load the Rocket.Chat templates into your project:

```
# oc create -f .openshift/rocket-chat-ephemeral.json
# oc create -f .openshift/rocket-chat-persistent.json
```

There are two template variants to choose from, depending on whether or not persistent storage is desired for the MongoDB pod:

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

```
# oc new-app rocket-chat-ephemeral -p \
ROCKETCHAT_IMAGE=registry.connect.redhat.com/rocketchat/rocketcha
t
```

B. For persistent storage:

```
# oc new-app rocket-chat -p \
ROCKETCHAT_IMAGE=registry.connect.redhat.com/rocketchat/rocketcha
t
```

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, you'll need to edit the MongoDB deployment config and include the GID used to access the storage (assuming either NFS or GlusterFS). It is recommended that GIDs used to access NFS or GlusterFS shares are unique to each share. Otherwise, shares can be owned by the group `nfsnobody` (GID 65534) for cases where group isolation between shares is not warranted or desired. For this guide, we will use NFS as our example storage.

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

```
# oc get pvc
NAME        STATUS      VOLUME      CAPACITY    ACCESSMODES ... AGE
mongodb    Bound       pv02        5Gi         RWO          ...  12m
```

Check with your cluster admin to obtain the GID of the NFS share backing the PV (in this case the PV is `pv02`. Once this GID is obtained, you can add it to the deployment config for MongoDB:

```
# oc edit dc mongodb
```

You'll need to the declare a `supplementalGroups` entry with the proper GID under the pod-level `securityContext` (55502 is the GID used in this example):

```
spec:
  containers:
    - name:
    ...
  securityContext:
    supplementalGroups: [55502]
```

Once the deployment config is edited, a new deployment should start. If this is not the case, check that the `supplementalGroups` entry was edited under the *pod-level* `securityContext`, and *not* at the container-level. Doing this erroneously will cause your deployment config changes to be ignored/discarded by OpenShift.

Continue onto the Validation section, which shows how to check the pods for readiness. If the MongoDB pod shows any errors when attempting to mount or write to the persistent volume, then work with your cluster admin to address the NFS share permissions in relation to the provided GID. The share should have group read/write/execute (eg: rwxrwx--- or 0770) permissions and must be owned by the group associated with the provided GID.

Validation

The first step in validating the deployment is to check the status of the pods. The following command should return two pods with a status of `Running`:

```
# oc get pods
NAME                      READY       STATUS      RESTARTS    AGE
mongodb-2-s1mws           1/1         Running     0           9m
rocketchat-1-562xt       1/1         Running     0           11m
```

NOTE: If you chose to use persistent storage, allow a few minutes for both containers to restart after editing the deployment config. If you don't see the rocketchat pod listed, refer to the Troubleshooting section at the end of this guide.

If both pods are running/ready, you should now be able to access Rocket.Chat from a web browser. First, you'll need to discover the public URL. The following command should return the hostname used to access the application:

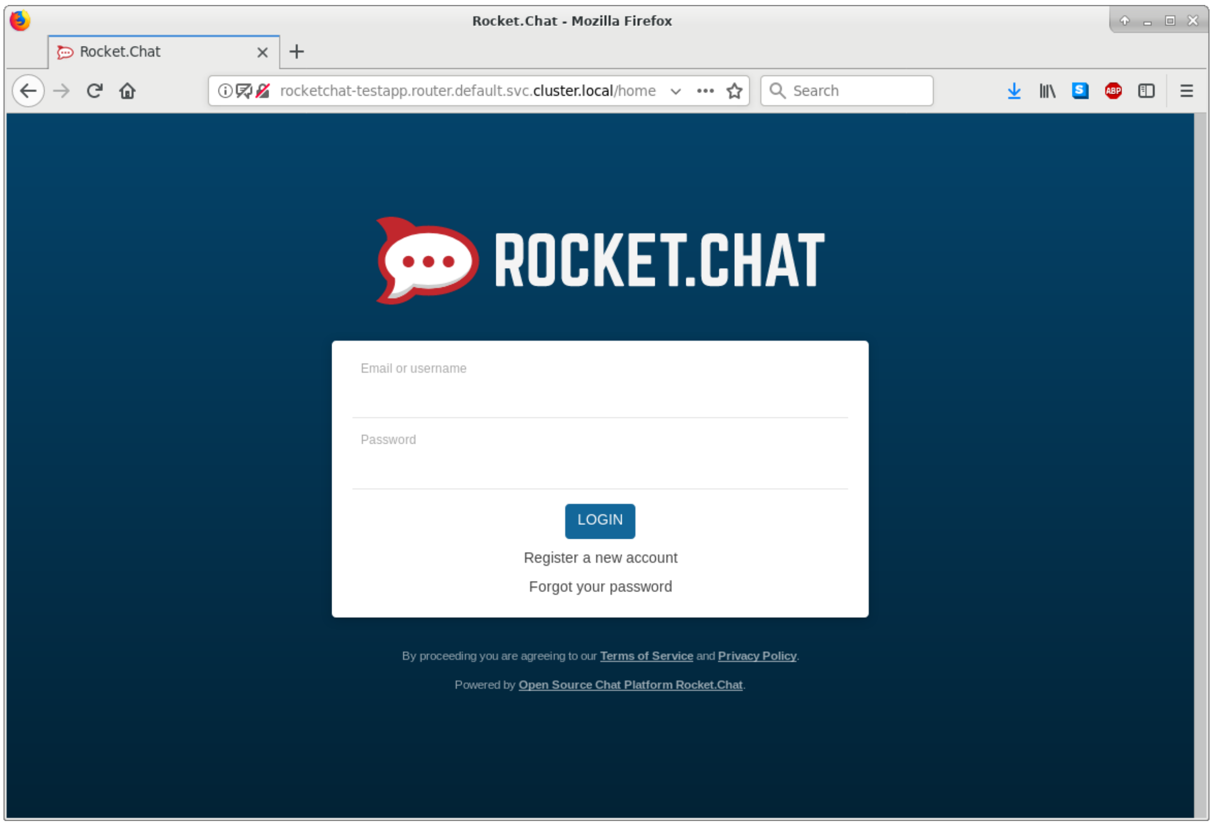
```
# oc get route
NAME          HOST/PORT
        PATH        SERVICES     PORT          TERMINATION     WILDCARD
rocketchat     rocketchat-testapp.router.default.svc.cluster.local
        rocketchat     3000-tcp                    None
```

Paste the obtained URL into a web browser, eg:
http://rocketchat-testapp.router.default.svc.cluster.local

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 rocketchat-testapp.router.default.svc.cluster.local
```

A successful deployment should bring you to the Rocket.Chat login page. From here, you can proceed with registering a user account and logging in:


Rocket.Chat - Mozilla Firefox
Rocket.Chat
rocketchat-testapp.router.default.svc.cluster.local/home
Search
ROCKET.CHAT
Email or username
Password
LOGIN
Register a new account
Forgot your password
By proceeding you are agreeing to our Terms of Service and Privacy Policy.
Powered by Open Source Chat Platform Rocket.Chat.

Congrats! You've deployed a fully containerized edition of Rocket.Chat to the OpenShift platform.

Troubleshooting

If you don't see the `rocketchat` pod listed as shown in the Validation section, or there are errors that weren't resolved by the pods restarting, there are a few steps you can take to isolate the cause. You can also refer to the Resources section for up to date errata pertaining to this guide.

An example output of `oc get pods` might show the database pod, but is missing the `rocketchat` application pod:

```
NAME                 READY    STATUS          RESTARTS   AGE
mongodb-1-ms7kb      1/1      Running         0          4m
```

Check the current status of the project, specifically the `rocketchat` deployment:

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

You can see from the previous command that the `rocketchat` deployment is stuck waiting on an image. The templates used to deploy Rocket.Chat 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/rocketchat/rocketchat:latest \
testapp/rocketchat:latest
```

Check the image stream details again for any possible errors:

```
# oc describe imagestream
...
 ! 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 for image pulling. It is easiest to delete the project and start from the beginning of the Deployment section:

```
# oc delete project testapp
```

As noted earlier on, any errors with Red Hat login credentials will prevent pods that depend on an RHCC image from deploying. However, deployment failures aren't always related to images or registry credentials. For instance, pods requiring persistent storage (such as databases) may fail to deploy when either a persistent volume isn't available or permissions to access the associated volume are lacking. In such cases, the `oc describe pod` command can be used to see pod errors in detail. Also, for templates which don't utilize an image stream, any error in regards to image pulling would be found by checking the pod details instead.

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 Rocket.Chat brings free, unlimited and open source team chat software, providing a communication platform to save time and money.

Resources

OpenShift

- OpenShift Container Platform 3.7 Documentation
- OpenShift Container Platform Reference Architectures

Rocket.Chat

- Rocket.Chat
- Rocket.Chat on OpenShift / Technical Implementation Guide Errata
- Rocket.Chat - Red Hat Container Catalog
- Rocket.Chat on GitHub

Contact Info

RocketChat:
contact@rocket.chat
sales@rocketchat.com

Red Hat Connect:
connect@redhat.com