
Table of Contents

Introduction	1.1
Setup	1.2

OpenShift 101 - Developer Operations

Overview	2.1
Network Security Policy	2.2
Builds	2.3
Registry Console	2.4
Deployments	2.5
Application Availability	2.6
Autoscaling	2.7
Persistent Storage	2.8
Persistent Configurations	2.9
Event Streams	2.10
Debugging Containers	2.11
Logging and Visualizations	2.12
Resource Requests and Limits	2.13
Pod Lifecycle	2.14

OpenShift Developer Workshop Labs

This application supports the following workshops at the BCDevOps / BC Developers Exchange / BC Gov CSI Lab:

- OpenShift 101 - Developer Operations
- OpenShift 201 - Pipeline your DevOps

Lab Setup

Prior to working on labs, the Platform Services team will have created 4 projects:

- tooling
- dev
- test
- production

One member from your group has been given administrative rights to the projects and is responsible for adding all other members to each project.

Assign yourself a unique name to be used during your application deployments. **Do not copy the application names directly from the lab guides** as you are in a shared environment with the rest of your team.

For example, if the lab says to create an application named `app1-[username]`, I would create an app named `app1-stewartshea`.

Lab Requirements

These labs will require access to the pathfinder production environment:

- [Pathfinder Web Console](#)

The environment can be accessed from a web browser such as Chrome or Firefox. For users that also leverage the CLI, the `oc` binary is available for download [here](#). The current version of OpenShift is 3.10 and should be installed into your PATH.

In addition, developers require GIT locally installed and must have a 2FA device to support access to GitHub.

- To login with the `oc` utility:
 - Login to the [Pathfinder Web Console](#) with your GitHub ID
 - Navigate to the top right corner, select the drop-down from your username, and select `Copy Login Command`
 - Paste the copied command into your terminal of choice

```
oc login https://console.pathfinder.gov.bc.ca:8443 --token=[token]
```

General OpenShift Components and Tasks

In these labs you will explore builds and deployments within the context of the Pathfinder OpenShift Platform. These labs will focus on a single 2-tier application based on the popular open source chat utility, Rocket.Chat.

The content in the labs will get `slimmer` as labs progress. The goal here is that previous learnings will help you familiarize yourself with the Web Console or CLI utilities.

Network Security Policy

By default, the new OpenShift namespace has been configured with zero network access.

- Attempt a general build

```
oc new-build https://github.com/ArctiqTeam/random-beer-giphy
```

- Notice that the build will fail
- Create the 3 basic NetworkSecurityPolicy objects
 - Obtain the sample policy

```
wget https://raw.githubusercontent.com/BCDevOps/platform-services/master/security/aporeto/docs/sample/quickstart-nsp.yaml
```

- Modify the policy with the appropriate namespace

```
---  
apiVersion: secops.pathfinder.gov.bc.ca/v1alpha1  
kind: NetworkSecurityPolicy  
metadata:  
  name: egress-internet  
spec:  
  description: |  
    allow the [INSERT NAMESPACE HERE] namespace to talk to the internet.  
  source:  
    - - $namespace=[INSERT NAMESPACE HERE]  
  destination:  
    - - ext:network=any  
---  
apiVersion: secops.pathfinder.gov.bc.ca/v1alpha1  
kind: NetworkSecurityPolicy  
metadata:  
  name: intra-namespace-comms  
spec:  
  description: |  
    allow the [INSERT NAMESPACE HERE] namespace to talk to itself  
  source:  
    - - $namespace=[INSERT NAMESPACE HERE]  
  destination:  
    - - $namespace=[INSERT NAMESPACE HERE]  
---  
apiVersion: secops.pathfinder.gov.bc.ca/v1alpha1  
kind: NetworkSecurityPolicy  
metadata:  
  name: int-cluster-k8s-api-comms  
spec:  
  description: |  
    allow [INSERT NAMESPACE HERE] pods to talk to the k8s api  
  destination:  
    - - int:network=internal-cluster-api-endpoint  
  source:  
    - - $namespace=[INSERT NAMESPACE HERE]
```

- Apply the policy again

```
oc apply -f quickstart-nsp.yaml
```

- Attempt the build again

```
oc start-build random-beer-giphy
```

- Remove the previous test build

```
oc delete build random-beer-giphy
```

Builds

In this lab, you will create a simple Docker based build for the Rocket Chat application.

Team Permissions

Once all projects have been created by the Platform Services team, the team admin must navigate to each project and assign your users the appropriate permissions.

As a team, find each project and add the rest of the team members. Feel free to experiment with the default roles.

- Once in the project, navigate to `Resources -> Membership -> Users`
- Select `Edit Membership`

- Add each user based on their GitHub id

- Select `Done Editing`

This can also be done on the CLI with the `oc` utility:

```
oc policy add-role-to-user [role] [username]
```

The Tools Project

The tools project is what will hold various support tools for the application. In this case, Jenkins will run here (later), and all builds will run in this project.

Creating a Docker-Based Build

The Rocket.Chat application build will be based off a minimal Dockerfile in a [public repository](#). Leveraging the commandline, you can use the `oc new-build` command to create all of the necessary OpenShift build components.

Ensure that all team members have edit rights into the project. Once complete, each member can create their own Rocket.Chat docker build.

- To start, switch to your new project. *Ensure that you are in the tools project in this lab*

```
oc project [project-name]
```

- With the `oc` cli, create the build

```
oc new-build https://github.com/BCDevOps/devops-platform-workshops-labs/ --context-dir=apps/rocketchat --name=rocketchat-[username]
```

- The output of the previous command should be similar to the following:

```
--> Found Docker image c71b6c4 (4 weeks old) from registry.access.redhat.com for "registry.access.redhat.com/rhscl/nodejs-8-rhel7"

Node.js 8
-----
Node.js 8 available as container is a base platform for building and running various Node.js 8 applications and frameworks. Node.js is a platform built on Chrome's JavaScript runtime for easily building fast, scalable network applications. Node.js uses an event-driven, non-blocking I/O model that makes it lightweight and efficient, perfect for data-intensive real-time applications that run across distributed devices.

Tags: builder, nodejs, nodejs8

* An image stream will be created as "nodejs-8-rhel7:latest" that will track the source image
* A Docker build using source code from https://github.com/BCDevOps/devops-platform-workshops-labs/ will be created
  * The resulting image will be pushed to image stream "rocketchat-[username]:latest"
  * Every time "nodejs-8-rhel7:latest" changes a new build will be triggered

--> Creating resources with label build=rocketchat-[username] ...
imagestream "nodejs-8-rhel7" created
imagestream "rocketchat-[username]" created
buildconfig "rocketchat-[username]" created
--> Success
Build configuration "rocketchat-[username]" created and build triggered.
Run 'oc logs -f bc/rocketchat-[username]' to stream the build progress.
```

- You can now explore the Web Console to watch the build status from `Builds -> Builds` *note* you will see multiple builds from each team member

Builds



The screenshot shows the OpenShift Container Platform interface. The top navigation bar includes 'OPENSHIFT CONTAINER PLATFORM' and user information. The left sidebar has sections for Overview, Applications, Builds, Resources, Storage, and Monitoring. The 'Builds' section is currently selected. The main content area displays a table titled 'Builds' with columns: Name, Last Build, Status, Duration, Created, Type, and Source. Three builds are listed:

Name	Last Build	Status	Duration	Created	Type	Source
rocketchat-jefkel	#1	Running	1 minute, 10 seconds	a minute ago	Docker	https://github.com/BCDevOps/devops-platform-workshops-labs/
rocketchat-stewartshea	#1	Running	5 minutes, 3 seconds	5 minutes ago	Docker	https://github.com/BCDevOps/devops-platform-workshops-labs/
rocketchat-sheaphillips	#1	Running	37 seconds	a few seconds ago	Docker	https://github.com/BCDevOps/devops-platform-workshops-labs/

- Or this can be done on the CLI

```
oc get bc  
oc status
```

- The build status can be monitored from the Web Console by selecting the `View Logs` link.

The screenshot shows the OpenShift Container Platform interface. The top navigation bar includes 'OPENSHIFT CONTAINER PLATFORM' and a user icon for 'stewartshea'. The left sidebar has sections for Overview, Applications, Builds, Resources, Storage, and Monitoring. The 'Builds' section is selected and shows a list of builds for the project 'devops-training-rc-tools'. One build, '#1 rocketchat-stewartshea', is currently running. The main content area displays the build details, including the build number, status, duration (8 minutes, 6 seconds), and creation time (8 minutes ago). A 'View Log' link is also present.

Builds > rocketchat-stewartshea

rocketchat-stewartshea created 8 minutes ago

build #1 rocketchat-stewartshea

History Configuration Environment Events

Build #1 is running. [View Log](#)

started 8 minutes ago

Filter by label	Add		
Build	Status	Duration	Created
#1	Running	8 minutes, 6 seconds	8 minutes ago

OpenShift Container Platform	
Overview	Search Catalog
Applications	Add to Project
Builds	Follow
Resources	
Storage	
Monitoring	
Catalog	

Interacting with the Registry Console

The registry console provides a web based interface to view the container images created for your projects. It can be used to obtain information on how to push/pull to the registry (from your local machine), as well as setting access rights on your project images.

Accessing the Registry Console

The registry console can be accessed at <https://registry-console-default.pathfinder.gov.bc.ca>.

- From the browser, navigate to the [registry console](#) and login if required

Viewing the Registry Push/Pull Commands

Navigate to the bottom of the [Overview](#) tab to find your specific docker login commands. These can be used if you wish to push images into your project space.

The screenshot shows the 'RED HAT CONTAINER REGISTRY' interface. On the left, there's a sidebar with 'Overview', 'Images', and 'Projects'. The main area has tabs for 'Images pushed recently' and 'All Projects'. A central panel says 'No images pushed' with a note to use commands below. Below this are sections for 'Login commands' and 'Image commands'. Under 'Login commands', there are examples for Docker and OpenShift. Under 'Image commands', there are examples for pushing and pulling images using sudo docker commands.

```

Log into the registry: $ sudo docker login -p  -c -e unused docker-registry-edge.pathfinder.gov.bc.ca
Log into OpenShift command line tools: $ oc login --token  -c console.pathfinder.gov.bc.ca:8443

Push an image: $ sudo docker tag myimage docker-registry-edge.pathfinder.gov.bc.ca/project/name:tag
$ sudo docker push docker-registry-edge.pathfinder.gov.bc.ca/project/name

Pull an image: $ sudo docker pull docker-registry-edge.pathfinder.gov.bc.ca/project/name:tag

```

Viewing Image Details

Navigate to the Images tab, select the correct project, and explore the image details. Also navigate to the Web Console and explore [Builds -> Images](#) to see similar details.

The screenshot shows the 'RED HAT CONTAINER REGISTRY' interface with the 'Images' tab selected. The sidebar shows 'Project: devops-training-rc-tools'. The main area lists images under 'Images' and shows their tags and repository details. An 'Image stream' section at the bottom contains commands for tagging and pushing images.

Name	Tags	Repository
devops-training-rc-tools/nodejs-8-rhel7	latest	docker-registry.default.svc:5000/devops-training-rc-tools/nodejs-8-rhel7
devops-training-rc-tools/rocketchat-jefkel	latest	docker-registry.default.svc:5000/devops-training-rc-tools/rocketchat-jefkel
devops-training-rc-tools/rocketchat-sheaphillips	latest	docker-registry.default.svc:5000/devops-training-rc-tools/rocketchat-sheaphillips
devops-training-rc-tools/rocketchat-stewartshea	latest	docker-registry.default.svc:5000/devops-training-rc-tools/rocketchat-stewartshea

```

Access Policy: Images may only be pulled by specific users or groups
Pulling repository: docker-registry.default.svc:5000/devops-training-rc-tools/rocketchat-stewartshea
Image count: 1
To push an image to this image stream:
$ sudo docker tag myimage docker-registry-edge.pathfinder.gov.bc.ca/devops-training-rc-tools/rocketchat-stewartshea:tag
$ sudo docker push docker-registry-edge.pathfinder.gov.bc.ca/devops-training-rc-tools/rocketchat-stewartshea:tag

```

Adding an Image Tag

In preparation for deployment to our dev environment, we will tag the latest version of our image with the tag `dev`.

- From the CLI

```
oc tag rocketchat-[username]:latest rocketchat-[username]:dev
```

- In the Registry Console, notice that the tag has now been added:

Name	Tags	Repository
devops-training-rc-tools/nodejs-8-rhel7	latest	docker-registry.default.svc:5000/devops-training-rc-tools/nodejs-8-rhel7
devops-training-rc-tools/rocketchat-jefkel	latest	docker-registry.default.svc:5000/devops-training-rc-tools/rocketchat-jefkel
devops-training-rc-tools/rocketchat-sheaphillips	latest	docker-registry.default.svc:5000/devops-training-rc-tools/rocketchat-sheaphillips
devops-training-rc-tools/rocketchat-stewartshea	dev latest	docker-registry.default.svc:5000/devops-training-rc-tools/rocketchat-stewartshea

Tags

Tag	From	Identifier	Last Updated
dev	rocketchat-stewartshea@sha256:42f0dcab7840fdde42f75e7135a0cfefea3fc24a99eb9a6660db5b529b011d12f	sha256:42f...	2 minutes ago
latest	pushed image	sha256:42f...	an hour ago

- From the CLI

```
oc get imagestreams
```

Changing Project Image Permissions

- Navigate to the Projects tab of the registry console and notice that the security for the project can be changed with the edit icon

Change project

Name: devops-training-rc-tools

Display name:

Description:

Access Policy:

- Or individual members / service account access can be added

The top screenshot shows the 'Add Member' dialog box. In the 'User or Group' field, 'watkinspd' is typed. In the 'Roles' dropdown menu, 'Select Role' is highlighted, with options like Admin, Push, and Pull listed below. The bottom screenshot shows the 'Membership' section of the project 'devops-training-rc-tools'. It lists a single member, 'watkinspd', with the role 'Pull' assigned.

- There is currently a [bug](#) if your username has upper case characters you will see "The member name contains invalid characters. Only letters, numbers, spaces and the following symbols are allowed: , = @ . _" You can skip this step as it just for informational purposes. You can also do this from the command line `oc policy add-role-to-user system:image-puller UpperUsername`
- Notice that the change is reflected in the Web Console

The screenshot shows the 'Membership' page in the OpenShift Container Platform. It displays a table of users and their assigned roles:

Name	Roles
stewartshea (you)	admin
sheaphillips	edit
watkinspd	registry-viewer
jefkel	view

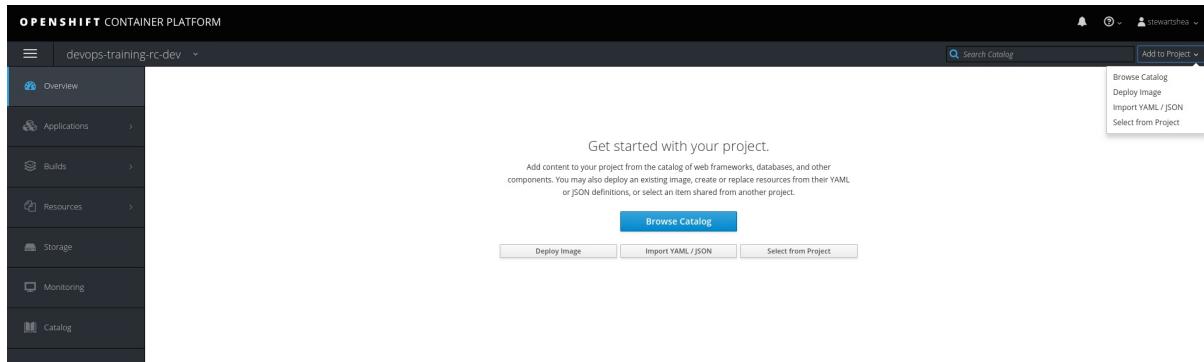
Deployment

Since the build and deploy stages are separate, and we have a built image, we can deploy this image into our remaining project spaces.

Create an Image-Based Deployment

Navigate to the configured `dev` project and deploy an image.

- From the Overview tab, select `Deploy Image`, or from the top right corner from the drop down `Add to Project`



- Select the tools project, the your specific rocketchat image, and the appropriate tag

Deploy Image

Image **Results**

1 2

Deploy an existing image from an image stream tag or image registry.

Image Stream Tag

devops-training-rc-tools / rocketchat-stewartshea : dev

⚠ Service account **default** will need image pull authority to deploy images from **devops-training-rc-tools**. You can grant authority with the command:
`oc policy add-role-to-user system:image-puller system:serviceaccount:devops-training-rc-dev:default -n devops-training-rc-tools`

Image Name

Image name or pull spec

To deploy an image from a private repository, you must [create an image pull secret](#) with your image registry credentials. [Learn More](#)

rocketchat-stewartshea:dev 2 hours ago, 450.0 MIB, 7 layers

This image will be deployed in Deployment Config [rocketchat-stewartshea](#).

Deploy Image

Image **Results**

1 2

This image declares volumes and will default to use non-persistent, host-local storage. You can add persistent storage later to the deployment config.

* **Name**

Identifies the resources created for this image.

Environment Variables [About Environment Variables](#)

<input type="text" value="Name"/>	<input type="text" value="Value"/>	<input type="button" value="X"/>
-----------------------------------	------------------------------------	----------------------------------

[Add Value](#) | [Add Value from Config Map or Secret](#)

Labels [About Labels](#)

Each label is applied to each created resource.

<input type="text" value="app"/>	<input type="text" value="rocketchat-stewartshea"/>	<input type="button" value="X"/>
----------------------------------	---	----------------------------------

[Add Label](#)

- Or do this from the CLI

```
oc new-app [devops-training-namespace]-tools/rocketchat-[username]:dev --name=rocketchat-[username]
```

- If performed with the CLI, the output should be as follows

```
--> Found image b949f08 (2 hours old) in image stream "[devops-training-namespace]-tools/rocketchat-[username]" under tag "dev" for "[devops-training-namespace]-tools/rocketchat-[username]:dev"

Node.js 8
-----
Node.js 8 available as container is a base platform for building and running various Node.js 8 applications and frameworks. Node.js is a platform built on Chrome's JavaScript runtime for easily building fast, scalable network applications. Node.js uses an event-driven, non-blocking I/O model that makes it lightweight and efficient, perfect for data-intensive real-time applications that run across distributed devices.

Tags: builder, nodejs, nodejs8

* This image will be deployed in deployment config "rocketchat-[username]"
* Ports 3000/tcp, 8080/tcp will be load balanced by service "rocketchat-[username]"
* Other containers can access this service through the hostname "rocketchat-[username]"
* This image declares volumes and will default to use non-persistent, host-local storage.
  You can add persistent volumes later by running 'volume dc/rocketchat-[username] --add ...'

--> Creating resources ...
imagestreamtag "rocketchat-[username]:dev" created
deploymentconfig "rocketchat-[username]" created
service "rocketchat-[username]" created
--> Success
Application is not exposed. You can expose services to the outside world by executing one or more of the commands below:
'oc expose svc/rocketchat-[username]'
Run 'oc status' to view your app.
```

-

Troubleshoot Image Pull Access

As the Web UI indicated, the `dev` project service accounts do not have the appropriate access to pull the image from the `tools` project.

- Navigate to `Applications -> Pods` and investigate further

Status	Template
Status: ✖ Image Pull Back-off Deployment: rocketchat-stewartshea, #1 IP: 172.51.71.26 Node: octopod-p-186.dmc (142.34.143.172) Restart Policy: Always	Containers: rocketchat-stewartshea ● Image: devops-training-rc-tools/rocketchat-stewartshea ↳ Port: 3000/HTTP, 8080/TCP ● Mount: rocketchat-stewartshea-1 -- /opt/app-root/src/uploads read-write ● Mount: default-token-bpf8s -- /var/run/secrets/kubernetes.io/serviceaccount read-only ● CPU: 100 millicores to 250 millicores ● Memory: 256 MiB to 1 GiB
Container: rocketchat-stewartshea State: Waiting (ImagePullBackOff) Ready: false Restart Count: 0	Volumes: rocketchat-stewartshea-1 Type: empty dir (temporary directory destroyed with the pod) Medium: node's default

- Navigate to the pods `Events` tab for more detail

The screenshot shows the OpenShift Container Platform Web Console. The left sidebar is for the project 'devops-training-rc-dev'. The main area shows a deleted pod named 'rocketchat-stewartshea-1-6ksk5'. The 'Events' tab is selected. The events log displays the following entries:

Time	Reason	Message
12:37:51 PM	Back-off	Back-off pulling image "172.50.0.2:5000/devops-training-rc-tools/rocketchat-stewartshea@sha256:42f06cab7840fde42f75e7135a0cfe43fc24a99eb9a6660db5b529b011d12f". 20 times in the last 12 minutes
12:34:21 PM	⚠ Failed	Error: ImagePullBackOff 6 times in the last 12 minutes
12:34:09 PM	⚠ Failed	Error: ErrImagePull 4 times in the last 12 minutes
12:34:09 PM	⚠ Failed	Failed to pull image "172.50.0.2:5000/devops-training-rc-tools/rocketchat-stewartshea@sha256:42f06cab7840fde42f75e7135a0cfe43fc24a99eb9a6660db5b529b011d12f": rpc error: code = Unknown desc = unauthorized: authentication required
12:34:09 PM	Pulling	pulling image "172.50.0.2:5000/devops-training-rc-tools/rocketchat-stewartshea@sha256:42f06cab7840fde42f75e7135a0cfe43fc24a99eb9a6660db5b529b011d12f" 4 times in the last 12 minutes
12:32:44 PM	Sandbox Changed	Pod sandbox changed, it will be killed and re-created.
12:32:40 PM	Scheduled	Successfully assigned rocketchat-stewartshea-1-6ksk5 to octopf-p-186.dmc

Create Proper Access Rights Across Projects

Any team member with admin rights on the `tools` project can create the appropriate permissions for each other project.

- From the Web Console

The screenshot shows the OpenShift Container Platform Web Console. The left sidebar is for the project 'devops-training-rc-tools'. The main area shows the 'Membership' section. The 'Service Accounts' tab is selected. The table lists service accounts and their assigned roles:

Name	Roles
devops-training-rc-tools / deployer	system:deployer
devops-training-rc-tools / builder	system:image builder
devops-training-rc-dev / default	system:image puller
devops-training-rc-test / default	system:image puller
devops-training-rc-prod / default	system:image puller

- From the CLI:

```
oc policy add-role-to-user system:image-puller system:serviceaccount:[devops-training-namespace]-dev:default -n [devops-training-namespace]-tools
oc policy add-role-to-user system:image-puller system:serviceaccount:[devops-training-namespace]-test:default -n [devops-training-namespace]-tools
oc policy add-role-to-user system:image-puller system:serviceaccount:[devops-training-namespace]-prod:default -n [devops-training-namespace]-tools
```

With the appropriate access in place, redeploy the application.

- From the Web Console

The screenshot shows the OpenShift Container Platform Web Console. The left sidebar is for the project 'devops-training-rc-dev'. The main area shows the 'Deployments' section for the application 'rocketchat-stewartshea'. The 'History' tab is selected. A red exclamation mark indicates a failure. The table shows the deployment history:

Deployment	Status	Created	Trigger
#1 (latest)	✗ Failed	23 minutes ago	Config change

- OR from the CLI

```
oc rollout latest rocketchat-[username]
```

- Validate that the image is able to be pulled

Deploying the Database

Before going into further deployment configuration options, review the current status of the application container.

Troubleshoot Deployment Issues

Navigate to the pod and review the logs to determine why we the container will not start.

- From the Web Console navigate to `Applications -> Pods -> rocketchat-[username]-[randomid]` and select Logs

```

1 /opt/app-root/src/bundle/programs/server/node_modules/fibers/future.js:313
2   throw(ex);
3 ^
4 MongoError: failed to connect to server [mongo:27017] on first connect [MongoError: getaddrinfo ENOTFOUND mongo mongo:27017]
5   at Error.ctor (domain.js:169:10)
6   at Pool. (events.js:211:7)
7   at Connection. (/opt/app-root/src/bundle/programs/server/npm/node_modules/meteor/npm-mongo/node_modules/mongodb-core/lib/connection/pool.js:200:12)
8   at Object.onceWrapper (events.js:317:30)
9   at Connection.emit (events.js:214:7)
10  at Socket.<anonymous> (/opt/app-root/src/bundle/programs/server/npm/node_modules/meteor/npm-mongo/node_modules/mongodb-core/lib/connection/connection.js:189:49)
11  at Object.onceWrapper (events.js:315:30)
12  at emitOne (events.js:94:13)
13  at Socket.emit (events.js:194:7)
14  at ontimeout (timers.js:430:11)
15  at tryOnTimeout (timers.js:283:5)
16  at listOnTimeout (internal/timers/destroy.js:64:8)
17  at ontimeout (internal/timers/destroy.js:64:8)
18  at _combinedTickCallback (internal/process/next_tick.js:138:11)
19  at process._tickCallback (internal/process/next_tick.js:180:9)

```

- Or from the CLI

```
oc get pods | grep rocketchat-[username]
oc logs rocketchat-[username]-[randomid]
```

note you can follow the logs with `oc -f`

Create Mongo Database with Ephemeral Storage

Having identified that the application is trying to connect to a mongo database, add a mongo database to the project for your application.

- From the Web Console
 - From the `Add to Project` dropdown, select `Browse Catalog`
 - In the search catalog area, type `mongo` and select `mongodb-ephemeral`
 - Ensure to customize the details with a service name such as `mongodb-[username]`, `username/password` and default database such as `rocketchat`

MongoDB (Ephemeral)



MongoDB (Ephemeral)

Red Hat, Inc.

DATABASE MONGODB

[View Documentation](#) [Get Support](#)

Default plan

MongoDB database service, without persistent storage. For more information about using this template, including OpenShift considerations, see <https://github.com/sclorg/mongodb-container/blob/master/3.2/README.md>.

WARNING: Any data stored will be lost upon pod destruction. Only use this template for testing

This template provides a standalone MongoDB server with a database created. The database is not stored on persistent storage, so any restart of the service will result in all data being lost. The database name, username, and password are chosen via parameters when provisioning this service.

[Cancel](#) [< Back](#) [Next >](#)

MongoDB (Ephemeral)

*** Memory Limit**

 Maximum amount of memory the container can use.

Namespace

 The OpenShift Namespace where the ImageStream resides.

*** Database Service Name**

 The name of the OpenShift Service exposed for the database.

MongoDB Connection Username

Username for MongoDB user that will be used for accessing the database.

MongoDB Connection Password

Password for the MongoDB connection user.

MongoDB (Ephemeral)

Cancel < Back Next >

Information	Configuration	Binding	Results
1	2	3	4

MongoDB Connection Username

Username for MongoDB user that will be used for accessing the database.

MongoDB Connection Password

Password for the MongoDB connection user.

*** MongoDB Database Name**

Name of the MongoDB database accessed.

MongoDB Admin Password

Password for the database admin user.

*** Version of MongoDB Image**

Version of MongoDB image to be used (2.4, 2.6, 3.2 or latest).

Cancel < Back Next >

MongoDB (Ephemeral)

Create a binding for **MongoDB (Ephemeral)**

Bindings create a secret containing the necessary information for an application to use this service.

Create a secret in **devops-training-rc-dev** to be used later
Secrets can be referenced later from an application.

Do not bind at this time
Bindings can be created later from within a project.

Cancel < Back Create

- Bonus: determine how to create this from the CLI

Deployment Configuration Options

As a result of using a generic `new-app` style deployment, as opposed to openshift specific templates, a lot of defaults are leveraged.

Environment Variables

By default we currently have no environment variables attached to our deployment configuration. So, while the app trying to start, and a database has been deployed, the app does not know how or where to connect to. Add an environment variable to the deployment configuration.

- In the Web Console, navigate to `Applications -> Deployments`, and select your deployment
- Select the `Environment Tab`
- Add the following environment variable with the connection string details configured for mongodb

OPENSHIFT CONTAINER PLATFORM

devops-training-rc-dev

Deployments > rocketchat-stewartshea

rocketchat-stewartshea created an hour ago

app rocketchat-stewartshea

History Configuration Environment Events

Name Value

MONGO_URL mongodb://buser:dbpass@mongodb-stewartshea:27017/rocketchat

Add Value | Add Value from Config Map or Secret

Environment From

Config Map/Secret

Select a resource

Prefix

Add ALL Values from Config Map or Secret

Save Clear Changes

- Click save and take note of what happens next
- Navigate to Applications -> Pods and Applications -> Deployments to notice the changes

Name	Status	Containers Ready	Container Restarts	Age
rocketchat-stewartshea-3-vpgjg	Running	1/1	1	a minute
rocketchat-stewartshea-2-vx6m5	Terminating	0/1	9	26 minutes
rocketchat-stewartshea-1-deploy	Error	0/1	0	an hour

Deployment	Status	Created	Trigger
#3 (test)	Active, 1 replica	2 minutes ago	Config change
#2	✓ Complete	27 minutes ago	Manual
#1	✗ Failed	an hour ago	Config change

While you are waiting for the application to redeploy, expose the route to the public internet.

- From the Web Console, navigate to Applications -> Routes
- Select create Route
 - Customize the name of the route, such as `rocketchat-[username]`
 - Ensure the service it points to is your particular service

Exploring Health Checks

With the new deployment running, monitor the readiness of the pod.

- Navigate to Applications -> Pods
- Notice that 1/1 containers are ready

Name	Status	Containers Ready	Container Restarts	Age
rocketchat-stewartshea-5-xhw5	Running	1/1	0	a minute
rocketchat-stewartshea-4-zjgn	Terminating	1/1	0	6 minutes
mongodb-stewartshea-1-x22dw	Running	1/1	0	8 minutes
rocketchat-stewartshea-1-deploy	Error	0/1	0	an hour

- Visit the application route, however, and notice that the application is not ready

Application is not available

The application is currently not serving requests at this endpoint. It may not have been started or is still starting.

- Possible reasons you are seeing this page:
- The host doesn't exist. Make sure the hostname was typed correctly and that a route matching this hostname exists.
 - The host exists, but doesn't have a matching path. Check if the URL path was typed correctly and that the route was created using the desired path.
 - Route and path matches, but all pods are down. Make sure that the resources exposed by this route (pods, services, deployment configs, etc) have at least one pod running.

Adding a Healthcheck

A container that is marked `ready` when it is not is an indication of a lack of (or misconfigured) healthcheck. Let's add a healthcheck.

- Navigate to `Applications -> Deployments`
- Select the appropriate deployment
- Select `Actions` and then `Edit Health Checks`

The screenshot shows the OpenShift Container Platform interface. On the left, there is a sidebar with navigation links: Overview, Applications, Builds, Resources, Storage, and Monitoring. The main area shows a deployment named 'rocketchat-stewartshea' under the 'rocketchat-stewartshea' application. The deployment table lists two entries: '#5 (latest)' with status 'Active, 1 replica' and 'Created 5 minutes ago', and '#4' with status 'Complete' and 'Created 10 minutes ago'. On the right, there is a context menu with options like Deploy, Edit, Pause Rollouts, Add Storage, Add Autoscaler, Edit Resource Limits, Edit Health Checks, Edit YAML, and Delete. The 'Edit Health Checks' option is highlighted.

- Select `Add Readiness Probe` and leverage the HTTP GET defaults for this application, setting an initial delay of 15 seconds

The screenshot shows the 'Edit Health Checks' dialog for the 'rocketchat-stewartshea' deployment. It has sections for 'Readiness Probe' and 'Liveness Probe'. Under 'Readiness Probe', the 'Type' is set to 'HTTP GET', 'Path' is empty, 'Port' is set to '3000', 'Initial Delay' is '15' seconds, and 'Timeout' is '1' second. There are buttons for 'Save' and 'Cancel' at the bottom.

- While the new deployment rolls out, continue to refresh the public route and validate that it stays up
- At the same time, monitor the pod `Containers Ready` column from `Applications -> Pods` and notice what happens when it becomes ready

Exploring Deployment Configuration Options

Additional actions are available to edit your deployment configuration. Review and explore;

- Resource Limits
- Healthcheck liveness probes
- YAML

Versioning a Deployment Configuration

At this point in time, your deployment configuration has undergone many changes, such as adding environment variables and adding health checks. Review the deployment configuration `History` tab:

- Select Deployment #1, right-click, and open in a new tab
- Select your latest deployment version, right-click, and open in a new tab
- Compare the differences - this can be done through the UI or by comparing the YAML

Changing Deployment Configuration Triggers

While reviewing the different deployment versions, take note of the `Trigger` column.

Deployment	Status	Created	Trigger
#6 [latest]	Active, 1 replica	11 minutes ago	Config change
#5	✓ Complete	19 minutes ago	Manual
#4	✓ Complete	24 minutes ago	Manual
#3	✓ Complete	an hour ago	Config change
#2	✓ Complete	an hour ago	Manual
#1	✗ Failed	2 hours ago	Config change

- Navigate to the `Configuration` tab of the deployment and review the currently configured Triggers

Explore how an Image can also trigger a deployment

- Navigate to your original build and investigate the available triggers

- Edit the buildconfig to change the output image to the `dev` tag



- Start a new build and monitor your the `dev` deployment when the build completes

The screenshot shows the OpenShift Container Platform interface. The left sidebar has sections for Overview, Applications, Builds, Resources, Storage, Monitoring, and Catalog. The main area shows the 'rocketchat-stewartshea' application under 'Deployments'. A message says 'Deployment #7 is running, View Log'. Below is a table of deployment history:

Deployment	Status	Created	Trigger
#7 (latest)	Running, 1 replica	a few seconds ago	Image change
#6	Active, 1 replica	20 minutes ago	Config change
#5	✓ Complete	28 minutes ago	Manual
#4	✓ Complete	33 minutes ago	Manual

Changing the Deployment Strategy Option

The default deployment configuration provides a `Rolling Update` style deployment, which waits for the container to be ready prior to cutting over traffic and terminating the previous container.

The screenshot shows the 'Edit Deployment Config' page for the 'rocketchat-stewartshea' application. The left sidebar is similar to the previous screenshot. The main area shows the 'Deployment Strategy' section:

- Strategy Type:** Rolling
- Timeout:** 600 seconds
- Maximum Number of Unavailable Pods:** 25%
- Maximum Number of Surge Pods:** 25%

- Change the strategy to a `Recreate` and redeploy a couple of times
- Refresh the browser URL right after a new deployment and observe the behavior
- Change the strategy back to `Rolling`

Application Availability

Prior to beginning this lab, navigate to the public route you created and finish the initial setup of your application. Ensure to select `Keep standalone` on the last page of the initial Rocket.Chat application setup.

Single Pod Applications

Single pod applications are not highly-available and can be abruptly terminated if the pod crashes or due to regularly scheduled platform maintenance. In order to simulate the effect on an application:

- Navigate to the pod, select `Actions` and `Delete`
- Select `Delete pod immediately`

- Refresh URL of application

Application is not available

The application is currently not serving requests at this endpoint. It may not have been started or is still starting.

- ⓘ Possible reasons you are seeing this page:
- The host doesn't exist. Make sure the hostname was typed correctly and that a route matching this hostname exists.
 - The host exists, but doesn't have a matching path. Check if the URL path was typed correctly and that the route was created using the desired path.
 - Route and path matches, but all pods are down. Make sure that the resources exposed by this route (pods, services, deployment configs, etc) have at least one pod running.

Scaling Pods

To increase the availability of an application and defend against unplanned outages or planned maintenance tasks, an application must have multiple pods/instance running. For this reason, stateless applications are desirable as they do not require custom clustering configurations.

Note Stateful applications do not support "scaling pods" as a form of high availability. Such a stateful example would be the mongodb database. For this reason, this lab focuses on the rocketchat application which will function with multiple pods. Please refer to specific application documentation for details on scalability support.

- Navigate to `Overview` or `Deployments` in the Web Console and increase the pod count to 2

APPLICATION
rocketchat-stewartshea

DEPLOYMENT CONFIG
rocketchat-stewartshea, #7

CONTAINERS
rocketchat-stewartshea
Image: devops-training-rc-tools/rocketchat-stewartshea
Ports: 3000/TCP and 1 other

NETWORKING
Service - Internal Traffic
rocketchat-stewartshea
3000/TCP (3000-tcp) → 3000 and 1 other

Average Usage Last 15 Minutes

Routes - External Traffic
<http://rocketchat-stewartshea-devops-training-rc-dev.pathfinder.gov.bc.ca>
Route rocketchat-stewartshea, target port 3000-tcp

2 pods

Deployments > rocketchat-stewartshea

rocketchat-stewartshea created 3 hours ago

Configuration

Details

Selectors: app=rocketchat-stewartshea, deploymentconfig=rocketchat-stewartshea

Replicas: 2

Strategy: Rolling

Timeout: 600 sec

Update Period: 1 sec

Interval: 1 sec

Max Unavailable: 25%

Max Surge: 25%

Autoscaling

Add Autoscaler

Hooks Learn More

none

- Or from the CLI

```
oc scale dc/rocketchat-[username] --replicas=2
```

- Notice the balancing across nodes by exploring the details of each pod

The image contains two side-by-side screenshots of the OpenShift Container Platform web interface. Both screenshots show the 'Details' tab for a specific pod.

Screenshot 1 (Top): The URL is 'devops-training-rc-dev'. The pod name is 'rocketchat-stewartshea-7-k6kcc'. Status: Running. Deployment: rocketchat-stewartshea, #7. IP: 172.51.68.135. Node: ociopf-p-186.dmz (142.34.143.172). Restart Policy: Always. State: Running since Sep 26, 2018 2:59:20 PM. Ready: true. Restart Count: 0.

Screenshot 2 (Bottom): The URL is 'devops-training-rc-dev'. The pod name is 'rocketchat-stewartshea-7-k82w2'. Status: Running. Deployment: rocketchat-stewartshea, #7. IP: 172.51.76.32. Node: ociopf-p-187.dmz (142.34.143.173). Restart Policy: Always. State: Running since Sep 26, 2018 3:14:26 PM. Ready: false. Restart Count: 0. There is a 'Debug in Terminal' link at the bottom.

- Or from the CLI notice the hosts the pod runs on (in the last field)

```
oc get pods -o wide | grep rocketchat-[username]
```

```
- The output should look similar to this:
```
$ oc get pods -o wide | grep rocketchat-stewartshea
rocketchat-stewartshea-7-k6kcc 1/1 Running 0 16m 172.51.68.135 ociopf-p-186.dmz
rocketchat-stewartshea-7-k82w2 0/1 Running 0 1m 172.51.76.32 ociopf-p-187.dmz
```

```

- Delete single pod, refresh the URL of application and notice that the application is served by the surviving pods
- Perform deployment, refresh the URL of application and notice that the application is served by the surviving pods

Autoscaling

Autoscaling can be configured on pods to enable OpenShift to add or remove pods as load varies. In general, the Horizontal Pod Autoscaler sets:

- Upper limit of pods
- Lower limit of pods
- Metric threshold to use for scaling tasks

Currently CPU and memory metrics are supported, with additional arbitrary metrics support intended for the future.

Autoscaling Pods

This lab will provide a simple demonstration of autoscaling based on CPU, as this is configurable in the Web Console.

- Navigate to your rocketchat deployment and select the `Actions` dropdown
- Select `Add Autoscaler`
- Configure an upper and lower limit of pods
- Configure a very low CPU Request Target (such as 1 or 2%) so that you can test it easily by browsing the web application
 - In a production environment you would taget something like 70-80%

Autoscale Deployment Config rocketchat-stewartshea

Scale replicas automatically based on CPU usage.
Learn More ↗

Autoscaler Name
rocketchat-stewartshea

A unique name for the horizontal pod autoscaler within the project.

Min Pods
1

The lower limit for the number of pods that can be set by the autoscaler. If not specified, defaults to 1.

Max Pods
3

The upper limit for the number of pods that can be set by the autoscaler.

CPU Request Target
10

The percentage of the CPU request that each pod should ideally be using. Pods will be added or removed periodically when CPU usage exceeds or drops below this target value.
Learn More ↗

Labels

app, rocketchat-stewartshea

Save **Cancel**

- Browse to the application to generate some load and monitor the behavior of the pods
 - Generate some activity such as creating messages, channels, etc.

APPLICATION
rocketchat-stewartshea

DEPLOYMENT CONFIGS
rocketchat-stewartshea, #7

CONTAINERS
rocketchat-stewartshea

Average Usage Last 15 Minutes

Max Memory	520
Core CPU	0.008
Kib/s Network	47

Autoscaled: min: 1, max: 3

NETWORKING

Service - Internal Traffic
Rocketchat-stewartshea

Routes - External Traffic
<http://rocketchat-stewartshea-devops-training-rc-dev.pathfinder.gov.bc.ca>

Route Rocketchat-stewartshea, target port 3000-tcp

- Review the deployment configuration and try to add or remove replicas

The screenshot shows the OpenShift Container Platform interface. On the left, there's a sidebar with options like Overview, Applications, Builds, Resources, Storage, Monitoring, and Catalog. The main area shows a deployment named "rocketchat-stewartshea" created 3 hours ago. The "Configuration" tab is selected. In the "Autoscaling" section, it shows the deployment config "app-rocketchat-stewartshea" with 3 replicas (autoscaled). The configuration includes:

Setting	Value
Replicas	3 replicas (autoscaled)
Strategy	Rolling
Timeout	600 sec
Update Period	1 sec
Interval	1 sec
Max Unavailable	25%
Max Surge	25%

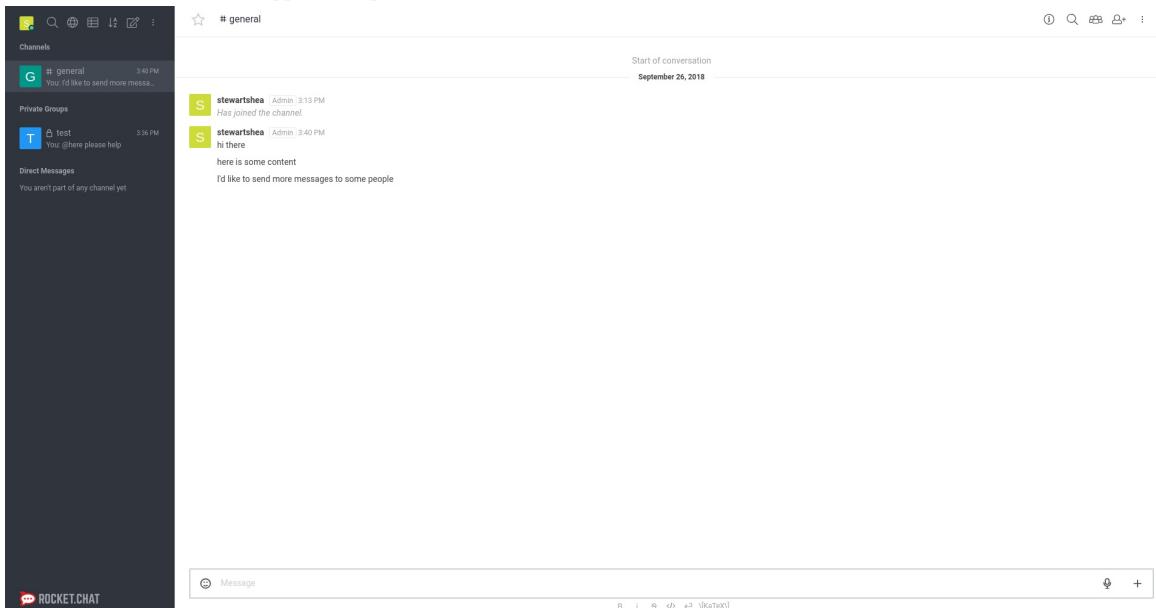
Below the configuration, there's a "Template" section. At the top right, there are "Deploy" and "Actions" buttons.

- Remove the autoscaler

Persistent Storage

Up to this point you have leveraged a single mongodb pod with ephemeral storage. In order to maintain the application data, persistent storage is required.

- Let's first take a look at our application prior to this lab



Deleting Pods with Ephemeral Storage

To understand what will happen when a pod with ephemeral storage is removed,

- Scale down both the rocketchat and mongo applications to 0 pods
- Scale back up each application pod to 1 replica

The screenshot shows the 'Setup Wizard' step 'Admin Info'. It has four fields: 'NAME' (placeholder: 'Type your name'), 'USERNAME' (placeholder: 'Type your username'), 'ORGANIZATION EMAIL' (placeholder: 'Type your email'), and 'PASSWORD' (placeholder: 'Type your password'). Below the fields is a 'Continue' button. To the left of the form, there's a sidebar with a navigation tree: '1 Admin Info' (selected), '2 Organization Info', '3 Server Info', and '4 Register Server'.

Adding Storage to Existing Deployment Configurations

Now that we notice all messages and configuration is gone, let's add persistent storage to the mongodb pod.

- Scale down both the rocketchat and mongo applications to 0 pods
- Edit the `mongodb-[username]` configuration
 - Remove the `emptyDir` volume
 - Add a new volume by selecting `Add Storage`

The screenshot shows the OpenShift Container Platform interface. The left sidebar has a dark theme with icons for Overview, Applications, Builds, Resources, Storage, Monitoring, and Catalog. The main area is titled "devops-training-rc-dev". A sub-menu bar at the top of the main area includes "Overview", "Builds", "Deployment Configs", "MongoDB", "MongoDB Shards", "MongoDB Replicasets", and "MongoDB Services". Below this, the "Deployment Configs" section lists "mongodb-stewartshea" (created 2 hours ago). The "Configuration" tab is selected. The "Details" section shows the following configuration:

- Selectors:** name=mongodb-stewartshea
- Replicas:** 0 replicas
- Strategy:** Recreate
- Timeout:** 600 sec

The "Template" section details the container configuration for "mongodb".

- Containers:** mongodb
 - Image:** openshift/mongodb
 - Ports:** 27017/TCP
 - Mount:** mongodb-stewartshea-data → /var/lib/mongodb/data (read-write)
 - Memory:** 512 MiB limit
 - Readiness Probe:** /bin/sh -i -c mongo 127.0.0.1:27017/\$MONGODB_DATABASE -u \$MONGODB_USER -p \$MONGO_... See All 3s delay, 1s timeout
 - Liveness Probe:** Open socket on port 27017 30s delay, 1s timeout

The "Volumes" section shows one volume:

- mongodb-stewartshea-data** [Remove](#)

Type: empty dir (temporary directory destroyed with the pod)
Medium: node's default

Buttons for "Add Storage" and "Add Config Files" are present.

The "Triggers" section includes:

- Manual (CLI):** oc rollout latest dc/mongodb-stewartshea -n d
- New Image For:** openshift/mongodb:3.2
- Change Of:** Config

The screenshot shows the OpenShift Container Platform interface. The left sidebar has a dark theme with icons for Overview, Applications, Builds, Resources, Storage, Monitoring, and Catalog. The main area is titled "devops-training-rc-dev". A sub-menu bar at the top of the main area includes "Overview", "Builds", "Deployment Configs", "MongoDB", "MongoDB Shards", "MongoDB Replicasets", and "MongoDB Services". Below this, the "Deployment Configs" section lists "mongodb-stewartshea". The "Add Storage" page is displayed, showing the message "No persistent volume claims." and the note "A persistent volume claim is required to attach to this deployment config, but none are loaded on this project." A "Create Storage" button is present.

- Select the `gluster-block` storage class, set the type to RWO (which is block storage), and the size to 1GB, with a name of `mongodb-[username]` **note** Each application will have a preferred storage type. This is NOT the recommended storageclass for mongo, but is useful in an upcoming lab.

- Scale up `mongodb-[username]` instance to 1 pod
- When mongo is running, scale `rocketchat-[username]` to 1 pod
- Configure RocketChat again
- Scale down and scale back up both the database and the rocketchat app
- Verify that data was persisted

RWO Storage

RWO storage (which was selected above) can only be attached to a single pod at a time, which causes issues in certain deployment strategies.

- Ensure your `mongodb-[username]` deployment is set to rolling

The screenshot shows the 'Edit Deployment Config' page for 'mongodb-stewartshea'. Under 'Deployment Strategy', 'Strategy Type' is set to 'Rolling'. The 'Timeout' is set to 600 seconds. The 'Maximum Number of Unavailable Pods' is set to 25%, and the 'Maximum Number of Surge Pods' is also set to 25%. There is a note about setting additional parameters or editing lifecycle hooks.

- Redeploy with Rolling Deployment
- Notice and investigate the issue

Name	Status	Containers Ready	Container Restarts	Age
mongodb-stewartshea-5-dvz25	Container Creating	0/1	0	a few seconds
mongodb-stewartshea-5-deploy	Running	1/1	0	a few seconds
mongodb-stewartshea-49b67	Running	1/1	0	2 minutes
rocketchat-stewartshea-7-rm4g5	Running	1/1	0	9 minutes

- Switch to recreate

RWX Storage

RWX storage allows multiple pods to access the same PV at the same time.

- Scale down `mongodb-[username]` to 0 pods

The screenshot shows the 'Storage' details page for 'mongodb-stewartshea'. It displays the following information:

- Status: Bound to volume `pvc-6f275f6f-c19c-11e8-a0a7-0050568348cc`
- Capacity: 1 GiB
- Requested Capacity: 1 GiB
- Access Modes: RWO (Read-Write-Once)

- Remove the previous storage volume and recreate as `gluster-file-db` with type RWX

The screenshot shows the OpenShift Container Platform interface. The left sidebar has a dark theme with the following navigation items:

- Overview
- Applications
- Builds
- Resources
- Storage
- Monitoring
- Catalog

The main content area is titled "Create Storage" under the "Storage" section. It contains the following fields:

- Storage Class:** A dropdown menu set to "gluster-file-db".
A note: "Storage classes are set by the administrator to define types of storage the users can select." with a "Learn More" link.
- Name:** An input field containing "mongodb-stewartshea".
A note: "A unique name for the storage claim within the project." with a "Learn More" link.
- Access Mode:** A radio button group where "Shared Access (RWX)" is selected.
A note: "Permissions to the mounted volume." with a "Learn More" link.
- Size:** An input field with "1" and a dropdown menu set to "GiB".
A note: "Desired storage capacity." with a "What are GiB?" link and a "Learn More" link.
- Use label selectors to request storage:** A checkbox that is unchecked.

At the bottom are two buttons: "Create" (highlighted in blue) and "Cancel".

- Scale down `mongodb-[username]` to 1 pods
- Redeploy with Rolling deployment

Persistent Configurations

In cases where configurations need to change frequently or common configurations should be shared across deployments or pods, it is not ideal to build said configurations into the container image or maintain multiple copies of the configuration. OpenShift supports `configMaps` which can be a standalone object that is easily mounted into pods. In cases where the configuration file or data is sensitive in nature, OpenShift supports `secrets` to handle this sensitive data.

ConfigMaps

Creating a Config Map and Adding it to a Deployment

Create a configMap with arbitrary data and mount it inside of your `rocketchat-[username]` pod:

- In the Web Console, create a `configMap` by navigating to `Resources -> Config Maps`. Note this can be performed directly from the deployment, or independently as these steps illustrate

The screenshot shows the OpenShift Container Platform interface. The top navigation bar includes a bell icon, a search bar labeled 'Search Catalog', and a user dropdown. The left sidebar has sections for Overview, Applications, Builds, Resources (selected), Storage, Monitoring, and Catalog. The main content area is titled 'Config Maps' with a sub-link 'Learn More'. A message 'No config maps.' is displayed, followed by a note 'No config maps have been added to project devops-training-rc-dev.' Below this is a blue 'Create Config Map' button.

- Name your configmap `rocketchat-[username]-configmap`, add an arbitrary key (ie. file name) and content; add as many files as you like

The screenshot shows the 'Create Config Map' dialog. The title bar says 'Config Maps > Create Config Map'. The dialog has fields for 'Name' (set to 'rocketchat-sheastewart-configmap') and 'Key' (set to 'myfile.txt'). The 'Value' field is empty. Below it is a 'Content' section with a text area containing the following text: 'We know there are tons of brilliant tech professionals like you who never get an opportunity to apply their skills to public service. We want to change that! Code With Us makes it easy to get paid for contributing to government's digital services by providing a process that allows you to focus on writing code, not contract paperwork.' At the bottom are 'Create' and 'Cancel' buttons.

- Attach the `configMap` to your `rocketchat-[username]` deployment by navigating to the `Configuration` tab and selecting `Add Configuration Files`



- This change will trigger a new deployment of your `rocketchat-[username]` pod
- Using the pod terminal in the Web Console or `oc rsh`, explore the path of the mounted configMap

```
sh-4.2$ cd /opt/config
sh-4.2$ ls -lha
total 8.0K
drwxr-xr-x  3 root 10030409000 4.0K Oct 24 20:21 .
drwxr-xr-x  2 root 10030409000 4.0K Oct 24 20:21 ..
-rw-r--r--  1 root 10030409000 1.0K Oct 24 20:21 myfile.txt
-rw-r--r--  1 root 10030409000 1.0K Oct 24 20:21 ..data
-rw-r--r--  1 root 10030409000 1.0K Oct 24 20:21 ..data->..myfile.txt
-rw-r--r--  1 root 10030409000 1.0K Oct 24 20:21 ..data/myfile.txt
-rw-r--r--  1 root 10030409000 1.0K Oct 24 20:21 ..data/myfile.txt
sh-4.2$
```

Changing Config Map Content

The content in your `configMap` can be changed and is dynamically updated in the pod. With that said, if the application does not support live reload of its configuration, a new deployment will be required for the changes to be picked up.

- Edit your `configMap` to change the content of the file or add a new file

* Key	<code>myfile.txt</code>
Value	<pre>We know there are tons of brilliant tech professionals like you who never get an opportunity to apply their skills to public service. We want to change that! Code With Us makes it easy to get paid for contributing to government's digital services by providing a process that allows you to focus on writing code, not contract paperwork.</pre>
* Key	<code>new-dynamic-file.txt</code>
Value	<pre>The Government of B.C. is procuring Agile software development teams.</pre>

- Using the pod terminal in the Web Console or `oc rsh`, explore the path of the mounted configMap

```

sh-4.2$ cd /opt/configs/
sh-4.2$ ls -l
total 8.0K
drwxrwxrwx  3 root 10939409009 4.0K Oct 24 20:21 .
drwxrwxrwx  3 root 10939409009 4.0K Oct 24 20:21 ..
drwxr-sr-x  2 root 10939409009 4.0K Oct 24 20:21 .2018_10_24_20_21_14.651506695
drwxrwxrwx  1 root 10939409009 4.0K Oct 24 20:21 ..data -> .2018_10_24_20_21_14.651506695
lrwxrwxrwx  1 root 10939409009 31 Oct 24 20:21 myFile.txt -> ..data/myFile.txt
lrwxrwxrwx  1 root 10939409009 31 Oct 24 20:21 myFile.txt -> ..data/myFile.txt
sh-4.2$ ls -l
drwxrwxrwx  3 root 10939409009 4.0K Oct 24 20:28 .
drwxr-sr-x  5 root 10939409009 4.0K Oct 24 20:28 ..
drwxrwxrwx  1 root 10939409009 4.0K Oct 24 20:28 ..data -> .2018_10_24_20_28_40.522769753
lrwxrwxrwx  1 root 10939409009 31 Oct 24 20:28 ..data -> .2018_10_24_20_28_40.522769753
lrwxrwxrwx  1 root 10939409009 31 Oct 24 20:28 myFile.txt -> ..data/myFile.txt
lrwxrwxrwx  1 root 10939409009 31 Oct 24 20:28 myFile.txt -> ..data/myFile.txt
sh-4.2$ ls
drwxrwxrwx  3 root 10939409009 4.0K Oct 24 20:28 ..
drwxrwxrwx  1 root 10939409009 4.0K Oct 24 20:28 ..data -> .2018_10_24_20_28_40.522769753
lrwxrwxrwx  1 root 10939409009 31 Oct 24 20:28 ..data -> .2018_10_24_20_28_40.522769753
lrwxrwxrwx  1 root 10939409009 31 Oct 24 20:28 myFile.txt -> ..data/myFile.txt
lrwxrwxrwx  1 root 10939409009 31 Oct 24 20:28 myFile.txt -> ..data/myFile.txt
sh-4.2$ 

```

Secrets

Secrets can be added in a similar way as config maps but are geared towards the management of sensitive information. In OpenShift, these are base64 encoded, and encrypted on disk when stored in the cluster. In Pods, they never live on disk (unlike configmaps) and are only in memory. Secrets, from the Web Console, are focused on supporting:

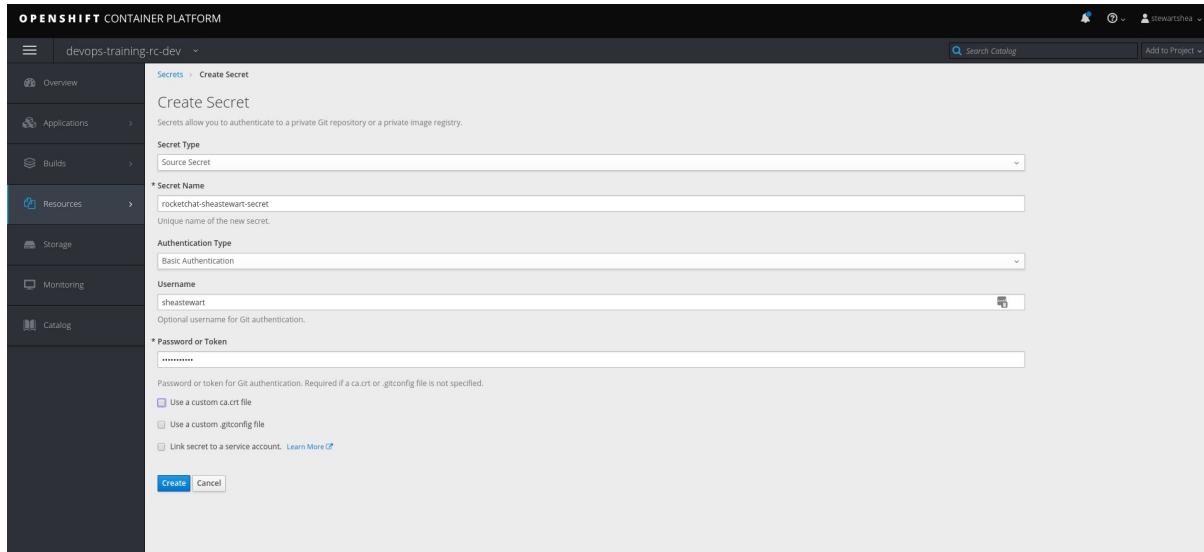
- Username/Passwords
- SSH Keys
- SSL Certificates
- Git config files

"Opaque" secrets are supported and can contain any type of data, however, these must be configured on the commandline with the `oc cli`.

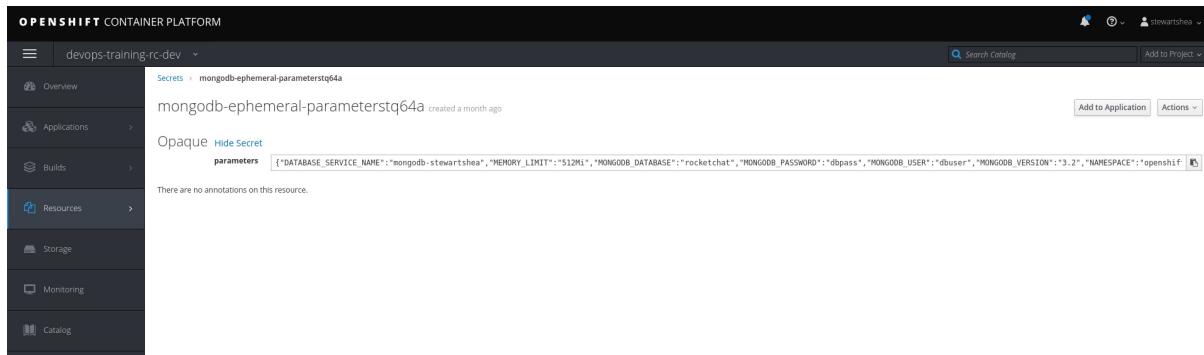
- In the Web Console, create a `secret` by navigating to `Resources -> Secrets -> Create Secret` Note this can be performed directly from the deployment, or independently as these steps illustrate

Name	Type	Created
c0d92ffd-c185-11e8-9419-0a58ac330c7e	Opaque	a month ago
mongodt-ephemeral-parameterstq64a	Opaque	a month ago
mongodt-stewartshea	Opaque	a month ago
builder-dockercfg-vrf6f	kubernetes.io/dockercfg	a month ago
default-dockercfg-v84rq	kubernetes.io/dockercfg	a month ago
deployer-dockercfg-cpq6v	kubernetes.io/dockercfg	a month ago
builder-token-pw7x2	kubernetes.io/service-account-token	a month ago
builder-token-vxzd4	kubernetes.io/service-account-token	a month ago
default-token-bpf8s	kubernetes.io/service-account-token	a month ago
default-token-vmb64	kubernetes.io/service-account-token	a month ago
deployer-token-pkkgm	kubernetes.io/service-account-token	a month ago
deployer-token-smth	kubernetes.io/service-account-token	a month ago

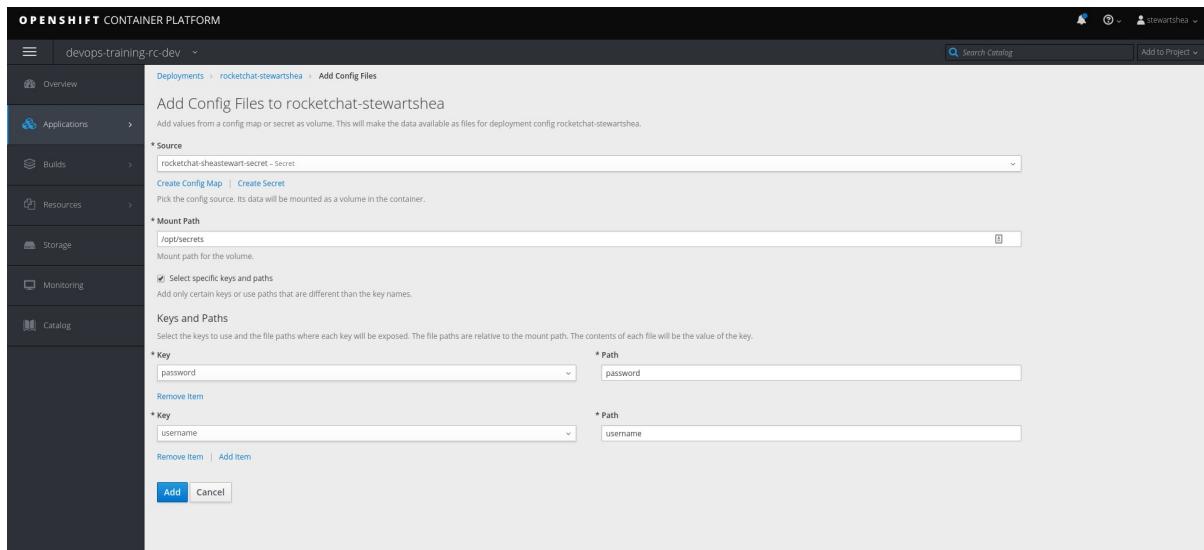
- Name your secret `rocketchat-[username]-secret`, add an arbitrary username/data or SSH key



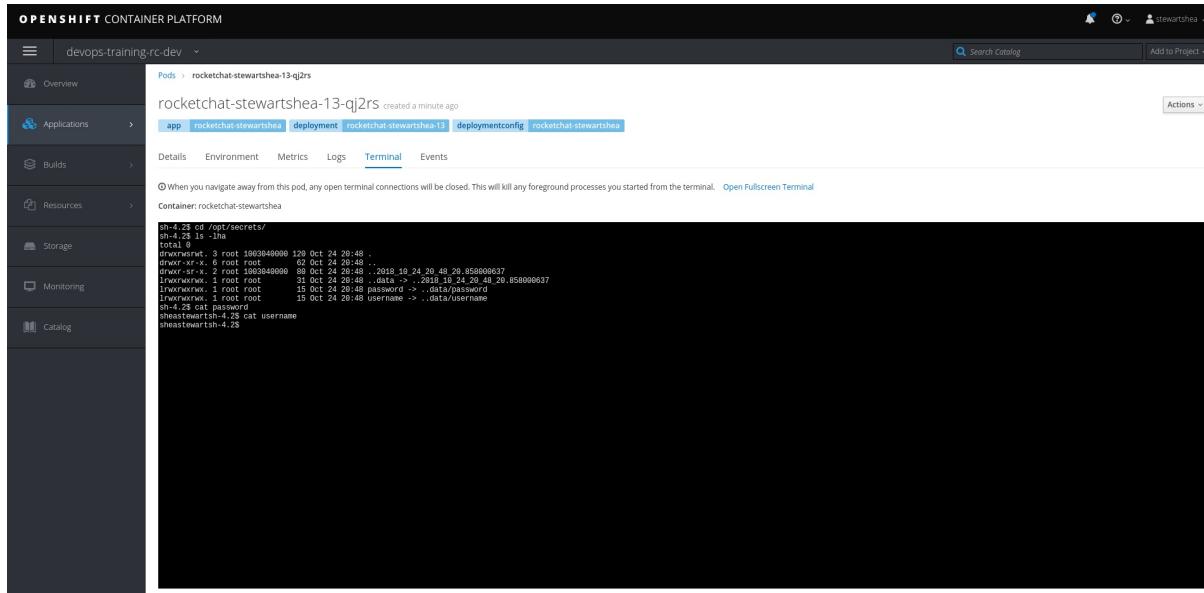
- Explore the other mongo secrets to see different variations of secret data



- Attach the `secret` to your `rocketchat-[username]` deployment by navigating to the `Configuration` tab and selecting `Add Configuration Files`
- Select the `secret` and set the mount directory; split up the keys into separate files if you like



- This change will trigger a new deployment of your `rocketchat-[username]` pod
- Using the pod terminal in the Web Console or `oc rsh`, explore the path of the mounted `secret`



- From the cli, review the secret with `oc describe secret rocketchat-[username]-secret`

```
oc describe secret rocketchat-sheastewart-secret
Name:      rocketchat-sheastewart-secret
Namespace:  devops-training-rc-dev
Labels:    <none>
Annotations: <none>

Type:  kubernetes.io/basic-auth

Data
=====
password: 11 bytes
username: 11 bytes
```

- Export the secret to view the contents with `oc get --export secret rocketchat-[username]-secret -o yaml`

```
oc get --export secret rocketchat-sheastewart-secret -o yaml
apiVersion: v1
data:
  password: c2h1YXN0ZXdhcnQ=
  username: c2h1YXN0ZXdhcnQ=
kind: Secret
metadata:
  creationTimestamp: null
  name: rocketchat-sheastewart-secret
type: kubernetes.io/basic-auth
```

- In order to reveal the contents of each key, base64 decode is required *Note* The Web Console automatically performs the base64 decode

```
echo "c2h1YXN0ZXdhcnQ=" | base64 -d
```

- To edit an existing secret, use `Edit Yaml` from the Web Console or `oc edit secret rocketchat-[username]-secret` from the cli

The screenshot shows the OpenShift Container Platform interface. In the top navigation bar, the project 'devops-training-rc-dev' is selected. Under the 'Secrets' section, a new secret named 'rocketchat-sheastewart-secret' is listed. The 'Edit YAML' button is highlighted. Below the secret, the JSON/YAML representation of the secret is shown:

```

1  apiVersion: v1
2  data:
3    password: c2hLYX08ZkdhcnQ=
4    username: cmFuZG9tbnV3Z3V5Cg==
5  kind: Secret
6  metadata:
7    creationTimestamp: '2018-10-24T20:42:33Z'
8    name: rocketchat-sheastewart-secret
9    namespace: devops-training-rc-dev
10   resourceVersion: '584936195'
11   selfLink: /api/v1/namespaces/devops-training-rc-dev/secrets/rocketchat-sheastewart-secret
12   uid: 552fa32-d7cd-11e8-b143-085056832285
13   type: kubernetes.io/basic-auth
14
15

```

- The updated content must be base64 encoded manually if updating in-place

```
echo "randomnewguy" | base64
cmFuZG9tbnV3Z3V5Cg==
```

Or visit <https://www.base64encode.org/> for encoding & decoding base64

- Similar to `configMaps`, the updated secret is automatically applied to the pod; no additional deployment is required

The screenshot shows a terminal session within a pod named 'rocketchat-sheastewart-13-qj2rs'. The user has run several commands to update the password and username, and then used 'cat username' to verify the changes. The terminal output is as follows:

```

sh-4.2$ cd /opt/secret/
sh-4.2$ ls -lH
drwxrwsrwt 3 root 10930409000 120 Oct 24 20:57 .
drwxr-xr-x  6 root root 10930409000 62 Oct 24 20:49 ..
drwxrwsrwt 1 root 10930409000 100 Oct 24 20:57 .data -> ./2018_10_24_20_57_38.917488279
lrwxrwxrwx  1 root root 10930409000 31 Oct 24 20:57 .data -> ./2018_10_24_20_57_38.917488279
drwxrwsrwt 1 root 10930409000 35 Oct 24 20:49 .data -> ./2018_10_24_20_57_38.917488279
drwxrwsrwt 1 root 10930409000 35 Oct 24 20:49 .data -> ./2018_10_24_20_57_38.917488279
sh-4.2$ cat username
randomnewguy
sh-4.2$
```

Event Streams

Event streams exist on many objects as well as at the project level. The project is the highest level that a developer can explore to see all events with that particular project.

Exploring Event Streams

The Web Console is the primary tool to visualize events sorted by time.

- Explore the events of a pod

This screenshot shows the OpenShift Container Platform Web Console. The left sidebar is collapsed. The top navigation bar shows the project 'devops-training-rc-dev' and the user 'stewartshea'. The main content area is titled 'Pods > mongodb-stewartshea-4-zhwp8'. Below this, there are tabs for 'deployment', 'deploymentconfig', 'mongodb-stewartshea', 'name', and 'mongodb-stewartshea'. The 'Events' tab is selected. A table lists events with columns for Time, Reason, and Message. The events listed are:

Time	Reason	Message
4:18:13 PM	Started	Started container
4:18:13 PM	Created	Created container
4:18:12 PM	Pulled	Container image "docker-registry.default.svc:5000/openshift/mongodb@sha256:335e1bfaea3d9c10d17d7fd4c84b7b57854d069b89e774e6825da42e52a85b0" already present on machine
4:18:09 PM	Scheduled	Successfully assigned mongodb-stewartshea-4-zhwp8 to ocioipf-p-186.dlmz

- For project wide events, navigate to Monitoring -> Events and select View Details

This screenshot shows the OpenShift Container Platform Web Console. The left sidebar is collapsed. The top navigation bar shows the project 'devops-training-rc-dev' and the user 'stewartshea'. The main content area is titled 'Monitoring > Events'. Below this, there are tabs for 'Monitoring' and 'Events'. The 'Events' tab is selected. A table lists events with columns for Time, Name, Kind, Reason, and Message. The events listed are:

Time	Name	Kind	Reason	Message
4:18:44 PM	rocketchat-stewartshea-7-rm4g5	Pod	Killing	Killing container with id docker://rocketchat-stewartsheaNeed to kill Pod
4:18:13 PM	rocketchat-stewartshea-7	Replication Controller	Successful Delete	Deleted pod: rocketchat-stewartshea-7-rm4g5
4:18:13 PM	rocketchat-stewartshea	Deployment Config	Replication Controller Scaled	Scaled replication controller "rocketchat-stewartshea-7" from 1 to 0 3 times in the last 43 minutes
4:18:13 PM	mongodb-stewartshea-4-zhwp8	Pod	Started	Started container
4:18:13 PM	mongodb-stewartshea-4-zhwp8	Pod	Created	Created container
4:18:12 PM	mongodb-stewartshea-4-zhwp8	Pod	Pulled	Container image "docker-registry.default.svc:5000/openshift/mongodb@sha256:335e1bfaea3d9c10d17d7fd4c84b7b57854d069b89e774e6825da42e52a85b0" already present on machine
4:18:09 PM	mongodb-stewartshea-4	Replication Controller	Successful Create	Created pod: mongodb-stewartshea-4-zhwp8
4:18:09 PM	mongodb-stewartshea-4-zhwp8	Pod	Scheduled	Successfully assigned mongodb-stewartshea-4-zhwp8 to ocioipf-p-186.dlmz
4:18:09 PM	mongodb-stewartshea	Deployment Config	Replication Controller Scaled	Scaled replication controller "mongodb-stewartshea-4" from 0 to 1
4:17:14 PM	mongodb-stewartshea	Persistent Volume Claim	Provisioning Succeeded	Successfully provisioned volume pvc-3689da3a-c19f-11e8-a0a7-005056834cc using kubernetes.io/glusterfs
4:16:52 PM	mongodb-stewartshea-5-dvr25	Pod	⚠️ Failed Mount	Unable to mount volumes for pod "mongodb-stewartshea-5-dvr25_develops-training-rc-dev(48fc6b0-c19f-11e8-a0a5-0050568379a2)": timeout expired waiting for volumes to attach/mount for pod "devops-training-rc-dev"/"mongodb-stewartshea-5-dvr25". list of unattached/unmounted volumes=[volume-8r9id default-token-bp8s]
4:16:26 PM	rocketchat-stewartshea-7-rm4g5	Pod	⚠️ Unhealthy	Readiness probe failed: Get http://172.51.69.33:3000/: net/http: request canceled (Client.Timeout exceeded while awaiting headers)
4:16:18 PM	mongodb-stewartshea-4-c9y67	Pod	Killing	Killing container with id docker://mongodb-Need to kill Pod

- Or on the CLI in your project

```
oc get events --sort-by='lastTimestamp'
```

- Navigate through some of the events and review some of the output that could be helpful in debugging pods

Debugging Containers

Accessing Local Logs

Logs of a running pod can be accessed from the Web Console or from the `oc` cli:

- The `Logs` tab of any running pod can be used to view active logs for the current pod

```

1 Updating process.env.MAIL_URL
2 Starting Email Interceptor...
3 Setting default file store to GridFS
4 LocalStore: store created at
5 LocalStore: store created at
6 LocalStore: store created at
7 LocalStore: store created at
8 LocalStore: store created at
9 LocalStore: store created at
10 Setting default file store to GridFS
11 Warning: connect(session) MemoryStore is not
12 designed for a production environment, as it will leak
13 memory, and will not scale past a single process.
14 Thu, 18 Oct 2018 19:49:15 GMT connect deprecated multipart: use parser (multiparty, busboy, formidable) npm module instead at npm/node_modules/connect/lib/middleware/bodyParser.js:56:20
15 Thu, 18 Oct 2018 19:49:15 GMT connect deprecated limit: Restrict request size at location of read at npm/node_modules/connect/lib/middleware/multipart.js:86:15
16 [object Object]
17 Migrating process.env.MAIL_URL
18 Using GridFS for custom sounds storage
19 Using GridFS for custom emoji storage
20 ufs: temp directory created at "/tmp/ufs"
21 -> System -> startup
22 +-----+
23 + | SERVER RUNNING
24 +-----+
25 + |
26 + | Rocket.chat Version: 0.69.2
27 + | NodeJS Version: 8.9.4 x64
28 + | Platform: linux
29 + | Process Port: 3000
30 + | Site URL: http://localhost:3000 / 
31 + | Bootstrapping: false, Disabled

```

- The `oc` command can be used to view or tail the logs:

```
oc logs -f <pod name>
```

If there is more than one container in a given pod, the `-c <container-name>` switch is used to specify the desired container logs.

Using a Debug Container

From the Web Console

In this lab, edit yaml of the `mongodb-[username]` deployment config.

- Under `spec` and `containers`, locate the line:

```
name: mongodb
```

- Insert the following line at the same indent level:

```
command: ["touch test"]
```

- Once the deployment change takes effect, notice the CrashLoopBackoff

- Click on `Debug in Terminal`
- Explore your capabilities within this container
- Once done, remove the previously added command, and notice how its placement and structure changed.

From the CLI

Using the `oc` command, start a debug container.

- Find the name of a pod you would like to debug

```
oc get pods
```

- Run the `oc debug` command to start a debug pod (your output will vary)

```
$ oc debug mongodb-1-s74g8
Defaulting container name to mongodb.
Use 'oc describe pod/mongodb-1-s74g8-debug -n crash-test' to see all of the containers in this pod.
Debugging with pod/mongodb-1-s74g8-debug, original command: container-entrypoint run-mongod
Waiting for pod to start ...
Pod IP: 10.131.22.12
If you don't see a command prompt, try pressing enter.
sh-4.2$ 
sh-4.2$ 
sh-4.2$ exit
exit

Removing debug pod ...
```

RSH and RSYNC

RSH is available to all normal pods through the web console under the `Terminal` tab, as well as through the `oc rsh` command.

- With your choice of access, rsh into one of the application pods and test access within the namespace
 - cURL internal and external resources
 - Test internal name resolution, external name resolution, etc.
 - Explore your userid

RSYNC is also available in many pods, available through the `oc rsync` command.

- On the CLI, type `oc rsync -h`

- Using this command, copy the contents of the mongo data directory to your local machine, or from your machine to the remote pod

Port Forwarding

The `oc port-forward` command enables users to forward remote ports running in the cluster into a local development machine.

- Find your pod and use the port forward command

```
oc get pods | grep rocketchat-[username]
oc port-forward [pod name from above] 8000:3000
```

- Navigate to <http://127.0.0.1:8000>

Logging and Visualizations

EFK for Aggregated Logs

The OpenShift platform provides an aggregated logging stack that is automatically configured to centralize and store logs from application pods. These logs are only retained for a short period of time, currently about 14 days, but can be used to help identify issues with application pods.

Kibana is the primary interface for viewing and querying logs.

Access the archive link from a pod

The shortcut towards accessing the Kibana is from the `Logs` tab of a running pod.

- Select the running `rocketchat-[username]` pod and select the Logs tab

```

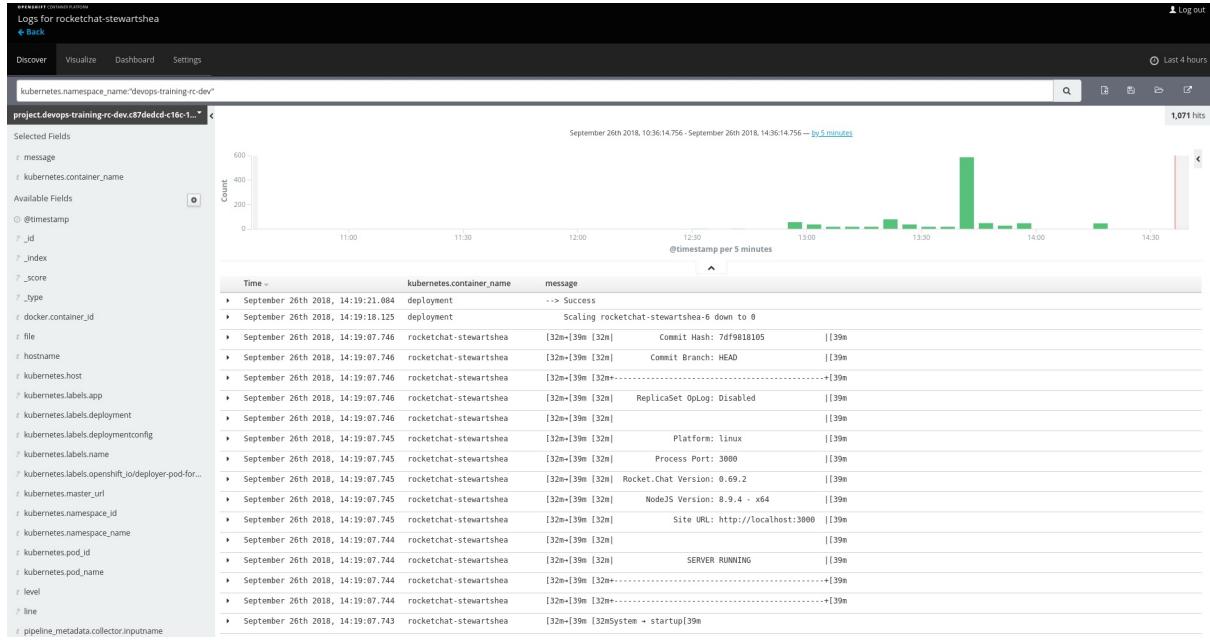
1 Updating process.env.MAIL_URL
2 Starting Email Interceptor...
3 LocalStore: store created at
4 LocalStore: store created at
5 LocalStore: store created at
6 LocalStore: store created at
7 LocalStore: store created at
8 LocalStore: store created at
9 Setting default file store to GridFS
10 Warning: connect(session) MemoryStore is not
11 designed for a production environment, as it will leak
12 memory, and will not scale past a single process.
13 Med 28 Sep 2018 16:58:26 GMT connect deprecated multipart: use parser '(multipart, busboy, formidable)' npm module instead at npv/node_modules/connect/lib/middleware/bodyParser.js:56:20
14 Med 28 Sep 2018 16:58:26 GMT connect deprecated limit: Restrict request size at location or read at npv/node_modules/connect/lib/middleware/multipart.js:86:15
15 {"tLine":158,"file":"rocketchat_migrations.js","message":"Migrations: Not migrating, already at version 151","time":"1537901168944","level":"info"}
16 Updating process.env.MAIL_URL
17 Using GridFS for custom sounds storage
18 Using GridFS for custom emoji storage
19 ufs: temp directory created at "/tmp/ufs"
20 + System + startup
21 + +-----+
22 + | SERVER RUNNING |
23 + +-----+
24 + |

```

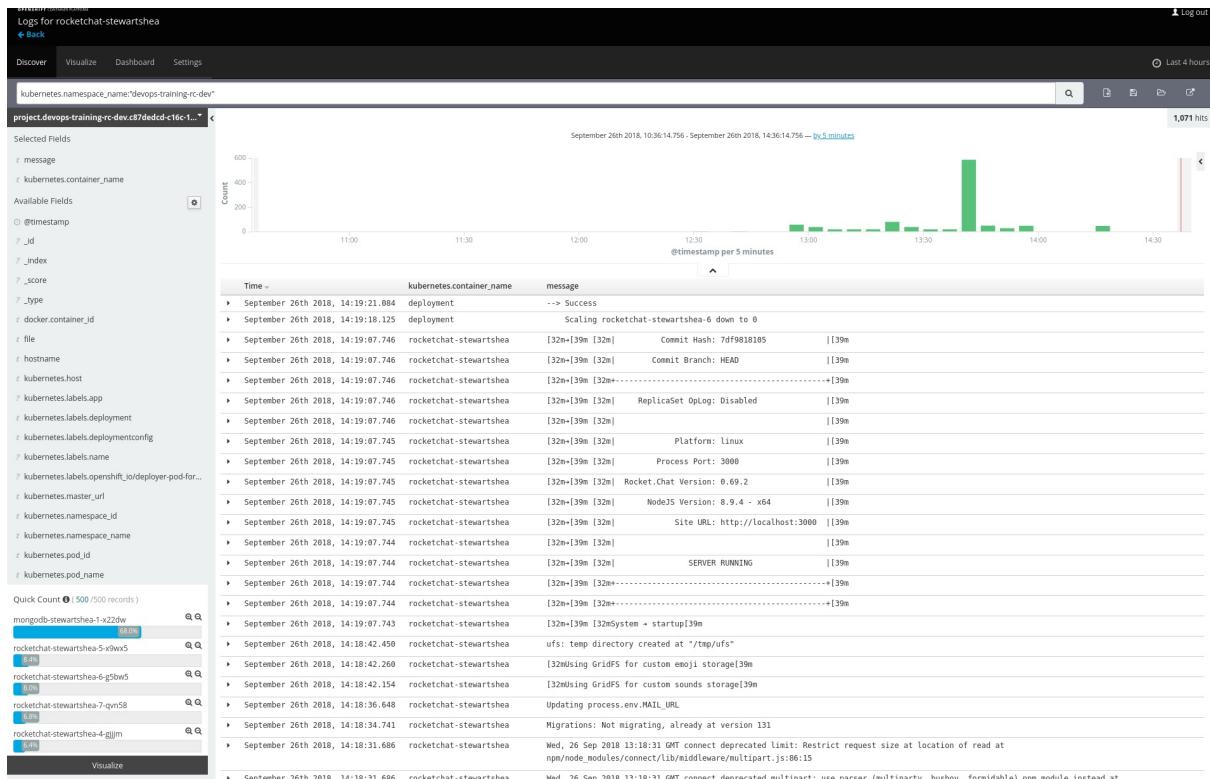
- Select the view archive link to be taken to Kibana
- Review the logging interface and the query that has been automatically populated

Time	kubernetes.container_name	message
September 26th 2018, 14:19:07.746	rocketchat-stewartshea	[32m+139m 132m] ReplicaSet Oplog: Disabled [39m
September 26th 2018, 14:19:07.746	rocketchat-stewartshea	[32m+139m 132m] Commit Hash: 7df9818105 [39m
September 26th 2018, 14:19:07.746	rocketchat-stewartshea	[32m+139m 132m] [39m
September 26th 2018, 14:19:07.746	rocketchat-stewartshea	[32m+139m 132m+-----+ [39m
September 26th 2018, 14:19:07.746	rocketchat-stewartshea	[32m+139m 132m] Commit Branch: HEAD [39m
September 26th 2018, 14:19:07.745	rocketchat-stewartshea	[32m+139m 132m] NodeJS Version: 8.9.4 - x64 [39m
September 26th 2018, 14:19:07.745	rocketchat-stewartshea	[32m+139m 132m] Site URL: http://localhost:3000 [39m
September 26th 2018, 14:19:07.745	rocketchat-stewartshea	[32m+139m 132m] Process Port: 3000 [39m
September 26th 2018, 14:19:07.745	rocketchat-stewartshea	[32m+139m 132m] Rocket.Chat Version: 0.69.2 [39m
September 26th 2018, 14:19:07.745	rocketchat-stewartshea	[32m+139m 132m] Platform: linux [39m
September 26th 2018, 14:19:07.744	rocketchat-stewartshea	[32m+139m 132m] SERVER RUNNING [39m
September 26th 2018, 14:19:07.744	rocketchat-stewartshea	[32m+139m 132m+-----+ [39m
September 26th 2018, 14:19:07.744	rocketchat-stewartshea	[32m+139m 132m] [39m
September 26th 2018, 14:19:07.744	rocketchat-stewartshea	[32m+139m 132m+-----+ [39m
September 26th 2018, 14:18:42.450	rocketchat-stewartshea	[32m+139m 132m] System + startup [39m
		ufs: temp directory created at "/tmp/ufs"

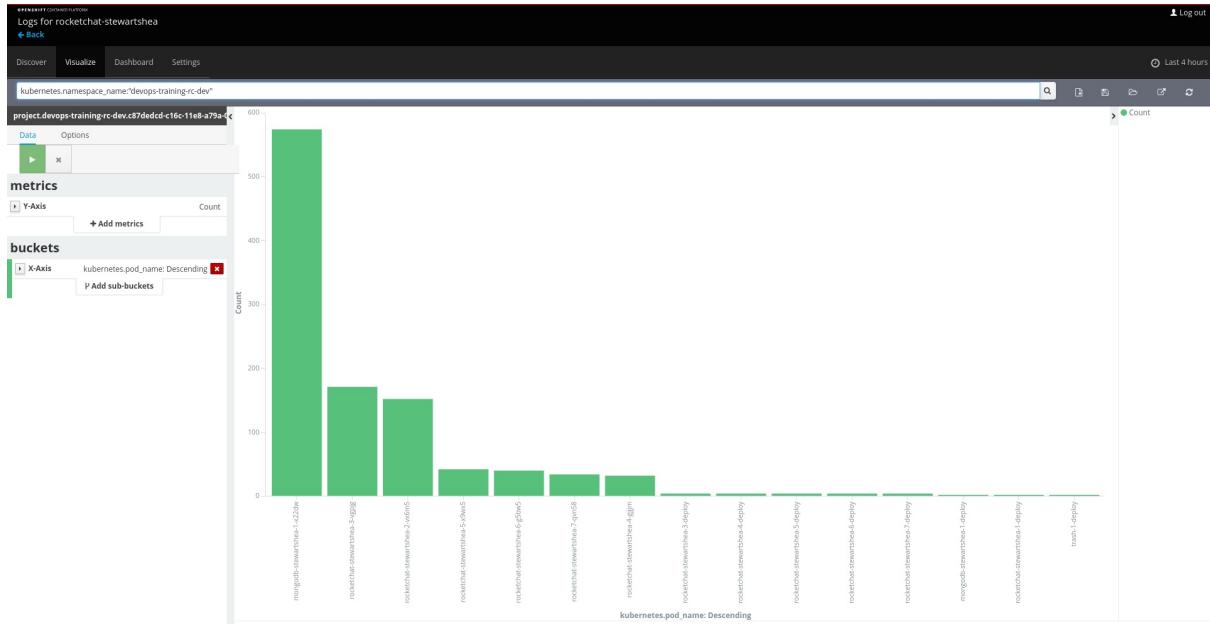
- Modify the query and time picker to select the entire namespace within the last few hours



- Review how Kibana surfaces key information about the log sources in the left panel



- Create a simple visualization of the information surfaced by Kibana



Access the kibana interface directly

Kibana can also be accessed directly at the url:

- <https://kibana.pathfinder.gov.bc.ca>

The namespaces you have access to view will be directly related to your project permissions.

Resource Requests and Limits

Tuning the resources assigned to a pod will have a direct effect on the performance of the application. Many templates include reasonable CPU and Memory resource configurations, however, new apps simply are deployed with the platform default.

Explore Default Resource Configurations

Since the Rocket Chat application was built from scratch and not deployed from a template, explore the current resources allocated to the pods:

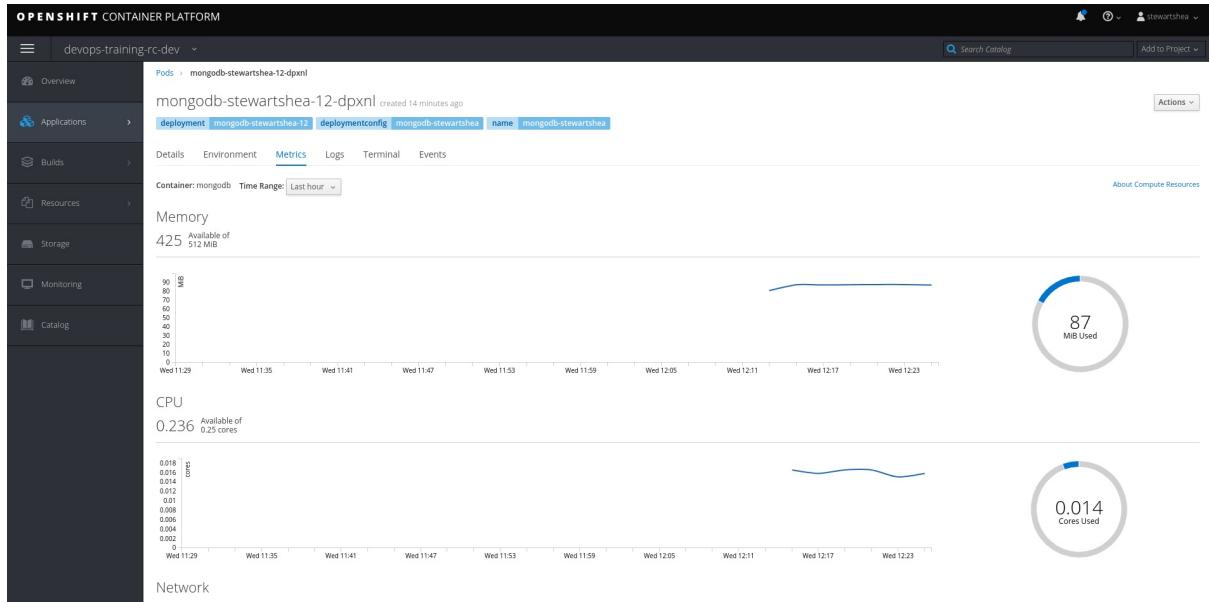
- Navigate to your rocketchat deployment and select `Actions -> Edit Resource Limits`

The screenshot shows the OpenShift Container Platform interface. The top navigation bar includes a bell icon, user profile, search catalog, and add to project options. The main area shows a deployment named 'rocketchat-stewartshea' created a month ago. On the left, there's a sidebar with 'Overview', 'Applications' (selected), 'Builds', 'Resources' (selected), and 'Storage'. On the right, a modal window titled 'Edit' is open, showing options like Deploy, Pause Rollouts, Add Storage, Add Autoscaler, Edit Resource Limits (which is highlighted in blue), Edit Health Checks, Edit YAML, and Delete. A sub-menu for 'Edit Resource Limits' is also visible.

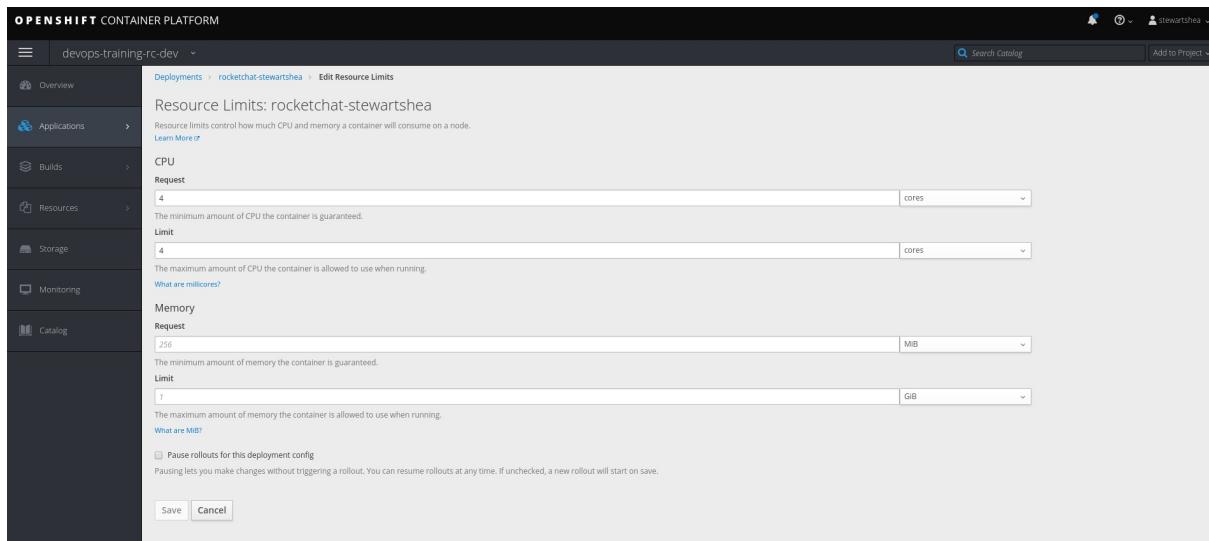
- Notice the defaults that are applied (in a light grey to indicate they are automatically set)

The screenshot shows the 'Edit Resource Limits' configuration page for the 'rocketchat-stewartshea' deployment. The left sidebar includes 'Overview', 'Applications' (selected), 'Builds', 'Resources' (selected), 'Storage', 'Monitoring', and 'Catalog'. The main content area is titled 'Resource Limits: rocketchat-stewartshea' and contains sections for CPU and Memory. Under CPU, 'Request' is set to 100 (millicores) and 'Limit' is set to 250 (millicores). Under Memory, 'Request' is set to 256 (MiB) and 'Limit' is set to 1 (GiB). A checkbox for 'Pause rollouts for this deployment config' is present. At the bottom are 'Save' and 'Cancel' buttons.

- Review the current metrics of your `rocketchat-<username>` pod



- Reduce the CPU request and limit to `50 millicores` and `100 Megabytes` and monitor the startup time of the pod
- Monitor the startup events of your pod and measure the time it takes to start
- Remove the limits previously imposed, and set your pod to 4 cores for the request and limit



- Monitor the status and speed of the new deployment
- Work with the rest of the class to determine why some pods may have succeeded, and others are failing.
- Remove the requests/limits previously set

Pod Lifecycle Customization

A Pod can be extended beyond the normal operation of the container by allowing developers to:

- add `init` containers
- add `pre` and `post` lifecycle hooks
- modify the default `entrypoint` of a container

Init Containers

Init containers are best used to prepare the pod for normal operation. In this lab, you will add a simple init container that posts a message to rocketchat with your pod hostname.

- From the Web Console, navigate to `Applications -> Deployments` and select your `rocketchat-[username]` deploymentconfig
 - If you wish to perform this from the cli with the `oc` tool, type `oc edit dc/rocketchat-[username]`
- Select `Actions -> Edit YAML`

The screenshot shows the OpenShift Container Platform interface. The top navigation bar includes 'OPENSHIFT CONTAINER PLATFORM', a user dropdown for 'stewartshea', and a search bar. The left sidebar has sections for Overview, Applications, Builds, Resources, Storage, and Monitoring. Under 'Applications', 'Deployments' is selected, and 'rocketchat-stewartshea' is shown. The main content area displays the deployment configuration. A 'History' tab is active, showing a single entry: 'Deployment #25 is active. View Log' (created 11 hours ago). To the right, there's a 'Actions' dropdown menu with options like Deploy, Edit, Pause Rollouts, Add Storage, Add Autoscaler, Edit Resource Limits, Edit Health Checks, Edit YAML, and Delete. The 'Edit' option is highlighted.

- Add the following section of code under `spec: -> template: -> spec:`

```
initContainers:
- name: init
  image: giantswarm/tiny-tools
  command: ["/bin/sh", "-c", "c=$(curl -X POST -H 'Content-Type: application/json' --data '{\"text\":\\\"Say Hello\\\"}' https://chat.pathfinder.gov.bc.ca/hooks/xxx/xxx)"]
```

- Select Save
- Ask the instructor to ensure the rocketchat instance is displayed to the class
- Explore the `Pod Details` to notice the differente with the Init Container

The screenshot shows the OpenShift Container Platform Web Console. In the left sidebar, under 'Applications', 'Deployments' is selected. Under 'Deployments', 'rocketchat-stewartshea-27-tf7ct' is selected. The main content area shows the pod details. The 'Status' section indicates the pod is running, deployed by 'rocketchat-stewartshea, #27' with IP '172.51.93.60' and node 'ociofp-p-196.dmz (142.34.143.154)'. The 'Init container init' section shows it completed successfully. The 'Template' section shows the container configuration, including command: '/bin/sh -c \$(curl -X POST -H "Content-Type: application/json" --data "{\"text\": \"\\\"\\\"\$HOSTNAME\\\"\\\" is at the postStart phase, huzzah! \"}" https://chat.pathfinder.gov.bc.ca/hooks/xxx/xxx)', ports: 3000/TCP, 8080/TCP, and various mounts and resource requirements.

In order to obtain logs from the init container, the `oc` command can be used by specifying `-c init`:

```
oc logs rocketchat-[username]-[pod-id] -c init
```

Lifecycle Hooks

Lifecycle hooks can be configured to start and stop a container properly. The lifecycle hook is tied directly to each container. Add a similar pre and post hook as the `initContainer` to demonstrate when it executes in your rocketchat deployment.

- From the Web Console, navigate to Applications -> Deployments and select your `rocketchat-[username]` deploymentconfig
 - If you wish to perform this from the cli with the `oc` tool, type `oc edit dc/rocketchat-[username]`
- Select Actions -> Edit YAML
- Add the following section of code under `spec: -> template: -> spec: -> containers`

```
lifecycle:
  postStart:
    exec:
      command: ["/bin/sh", "-c", "c=$(curl -X POST -H 'Content-Type: application/json' --data '{\"text\": \\"\\\"\\\"$HOSTNAME\\\"\\\" is at the postStart phase, huzzah! \"}' https://chat.pathfinder.gov.bc.ca/hooks/xxx/xxx)"]
  preStop:
    exec:
      command: ["/bin/sh", "-c", "c=$(curl -X POST -H 'Content-Type: application/json' --data '{\"text\": \\"\\\"\\\"$HOSTNAME\\\"\\\" is just about to STOPPPPPP! \\\"}' https://chat.pathfinder.gov.bc.ca/hooks/xxx/xxx)"]
```

- Select Save
- Ask the instructor to ensure the rocketchat instance is displayed to the class

Overriding the Entrypoint

It may be necessary, from time to time, to override the initial command/entrypoint of a container image. Generally this is used for troubleshooting purposes, or to override a vendor provided image.

- From the Web Console, navigate to Applications -> Deployments and select your `rocketchat-[username]` deploymentconfig
 - If you wish to perform this from the cli with the `oc` tool, type `oc edit dc/rocketchat-[username]`
- Select Actions -> Edit YAML

- Add the following section of code under `spec: -> template: -> spec: -> containers`

```
command:  ["/bin/sh", "-c", "c=$(curl -X POST -H 'Content-Type: application/json' --data '{"text": "'$HOSTNAME'" is AN OVERRIDING COMMAND! }' https://chat.pathfinder.gov.bc.ca/hooks/xxx/xxx)"]
```

- Take note of the pattern that will happen in the rocketchat notification screen
- Remove the previous command to enable the rocketchat instance to start properly again

References

- <https://blog.openshift.com/kubernetes-pods-life/>