***Using the oc command-line application***

In appendix A, you install oc on your laptop or workstation. This is the tool you’ll use

to manage OpenShift on the command line. If you’re using an OSX or Linux system,

you can open your favorite terminal application. On Windows, open your command

prompt. From your command line, run the oc login command, using dev for the

username and password and the URL for your master server’s API server:

$ oc login -u dev -p dev https://ocp-1.192.168.122.100.nip.io:8443

Login successful.

You don't have any projects. You can try to create a new project, by running

oc new-project <projectname>

***(continued)***

Also, please use the issue tracker and other tools in the GitHub organization to communicate

with us. Our plan is to continue to work with and update these projects even

after the book is published. So file issues, submit pull requests, and contribute to

the *OpenShift in Action* community.

**The syntax for logging in to an OpenShift**

**cluster, including the username, password,**

**and URL for your OpenShift master’s API server**

The default project and working with multiple projects

oc new-project image-uploader --display-name='Image Uploader Project'

***Deploying an application***

$ oc new-app \

> --image-stream=php \

> --code=https://github.com/OpenShiftInAction/image-uploader.git \

> --name=app-cli

$ oc describe svc/app-cli

***Exposing services to the outside world with routes***

oc expose svc/app-cli

You can get information about the route you just created by running the oc describe

route/app-cli command:

$ oc describe route/app-cli

scale the app-cli

application to two pods with the following command:

oc scale dc/app-cli --replicas=2

$ oc get pods --show-all=false

NAME READY STATUS

single replica of the app-cli

application with the following command:

oc scale dc/app-cli --replicas=1

The same functionality is available on the command line. To scale the number of pods

back down to one, run the following oc scale command:

$ oc scale dc app-cli-1 --replicas=1

At this

To create the readiness probe, use the command line and run the oc set probe

command:

$ oc set probe dc/app-cli \

--readiness \

--get-url=http://:8080/notreal \

--initial-delay-seconds=5

deploymentconfig "app-cli" updated

The output includes a message that the deployment configuration was updated. Just

like a liveness probe, creating a readiness probe triggers the creation of a new app-cli

deployment. Check to see whether the new pods were deployed by running the oc

get pods command:

In the OpenShift CLI, run the following commands to create a new test environment

and deploy a new instance of MongoDB:

$ oc new-project test --display-name="ToDo App - Test"

...

$ oc new-app \

-e MONGODB\_USER=oiatestuser \

-e MONGODB\_PASSWORD=password \

-e MONGODB\_DATABASE=tododb \

-e MONGODB\_ADMIN\_PASSWORD=password mongodb:3.2

The test project can now access images in the dev project. Next, deploy the image

tagged promoteToTest in the dev project into your test project:

$ oc new-app dev/todo-app-flask-mongo:promoteToTest

By default, the Python image that you used to create your application is configured for

applications listening on port 8080—but the actual application listens on port 5000.

Run this command to update to the correct port:

$ oc patch svc todo-app-flask-mongo --type merge \

--patch '{"spec":{"ports":[{"port": 8080, "targetPort": 5000 }]}}'

service "todo-app-flask-mongo" patched

Now that the service has been correctly updated, create a route for the ToDo application:

$ oc expose svc todo-app-flask-mongo

route "todo-app-flask-mongo" exposed

Normally, everything would be good to go at this point. But if you double-check your

pods, you’ll notice that the pod isn’t running as expected:

$ oc get pods

NAME READY STATUS