Lab 2: Zero Downtime Deployments (blue-green deployments)

Description: During this lab we will push an application, then simulate updating it with a new version and manage the application routes for seamless upgrading without any downtime.

Note that commands are in *italics* and the X should be replaced with a unique number or participant’s initials.

1. Change to the odca-paas-workshop directory.
2. Create a Mongo DB service using the following command. We will bind our application to this service.

*cf create-service mongolab sandbox mymongo-userX*

*cf bind-service PaaSTutorial-userX mymongo-userX*

*cf create-service mongolab sandbox cf-workshop-mongo*

We will explaining binding services in more detail on the next lab but for now it is required for the PaaSTutorial to launch correctly.

1. Push the PaaSTutorial application. The information for the app is captured in the manifest.yml file:  
     
    *---*

*applications:*

*- name: PaaSTutorial-userX*

*memory: 256M*

*services:*

*- mymongo-userX*

and the instructions for how to package and start the node.js app are found within package.json:

*{*

*"name": "PaaS-Workshop",*

*"version": "0.0.1",*

*"private": true,*

*"scripts": {*

*"start"* ***: "node Index.js****"*

*},*

*"dependencies" : {*

*"express": "3.3.4",*

*"mongodb":"1.3.23",*

*"jade": "1.1.5"*

*}*

*}*

For that reason the push is as simple as the following line:

*cf push –i 2*

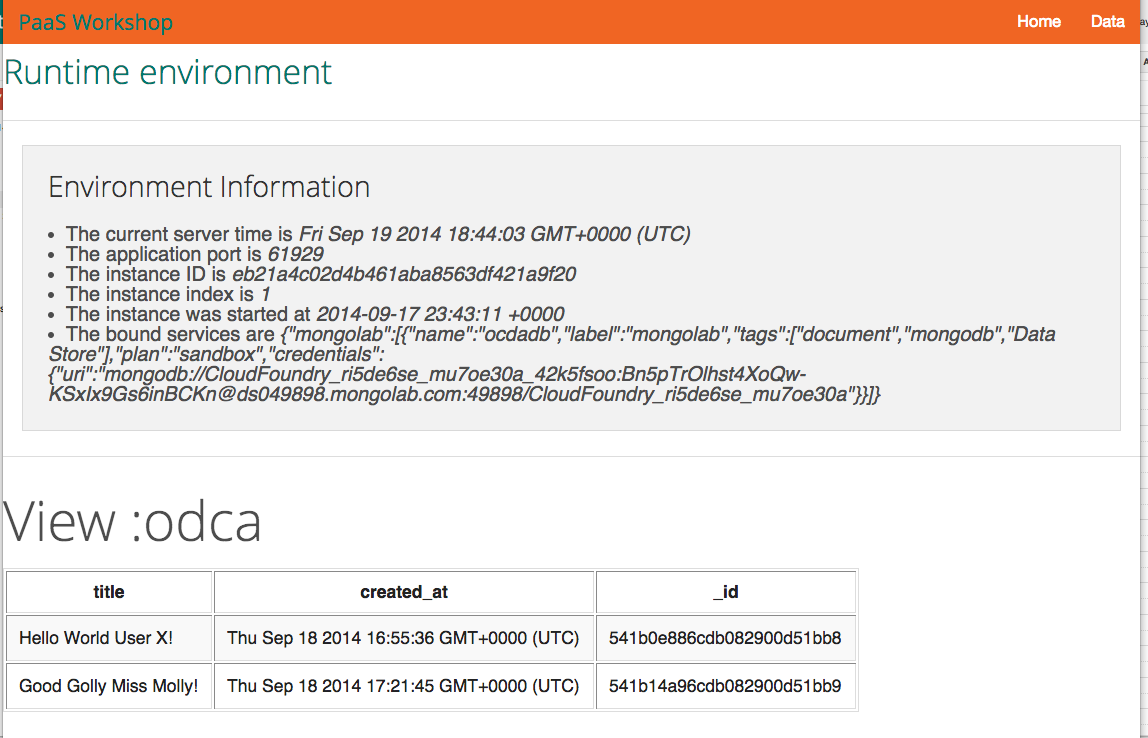
This will start the app with 2 instances of the application running bound to the mymongo-userX service.

Open a web browser and visit <http://paastutorial-userX.cfapps.io/>



* Notice the port displayed at the top in the “Environment information” section. This is the port from the application that is serving your customer.
* Also notice the mongo service is bound to the application. We’ll complete this in the next lab.
* You can populate the date with the curl command:  
    
  *curl -H "Content-Type: application/json" -X POST -d '{"title":"Hello World  ODCA!"}'* [*http://paastutorial-userX.cfapps.io/collections/odca*](http://paastutorial-userX.cfapps.io/collections/odca)

Passing in the db name of your choice (e.g. odca) and confirm with viewing in the browser <http://paastutorial-userX.cfapps.io/collections/odca> (please adjust url for your db name if you choose not to use odca).



1. Now let’s say you made an update to that app, changing the PaasTutorial to have the functionality of wanting to track the attendees in our workshop. Let’s push the new version of our app and test it.

*cd cf-workshop-node*

*update the manifest in this directory by changing the app name cf-workshop-userX*

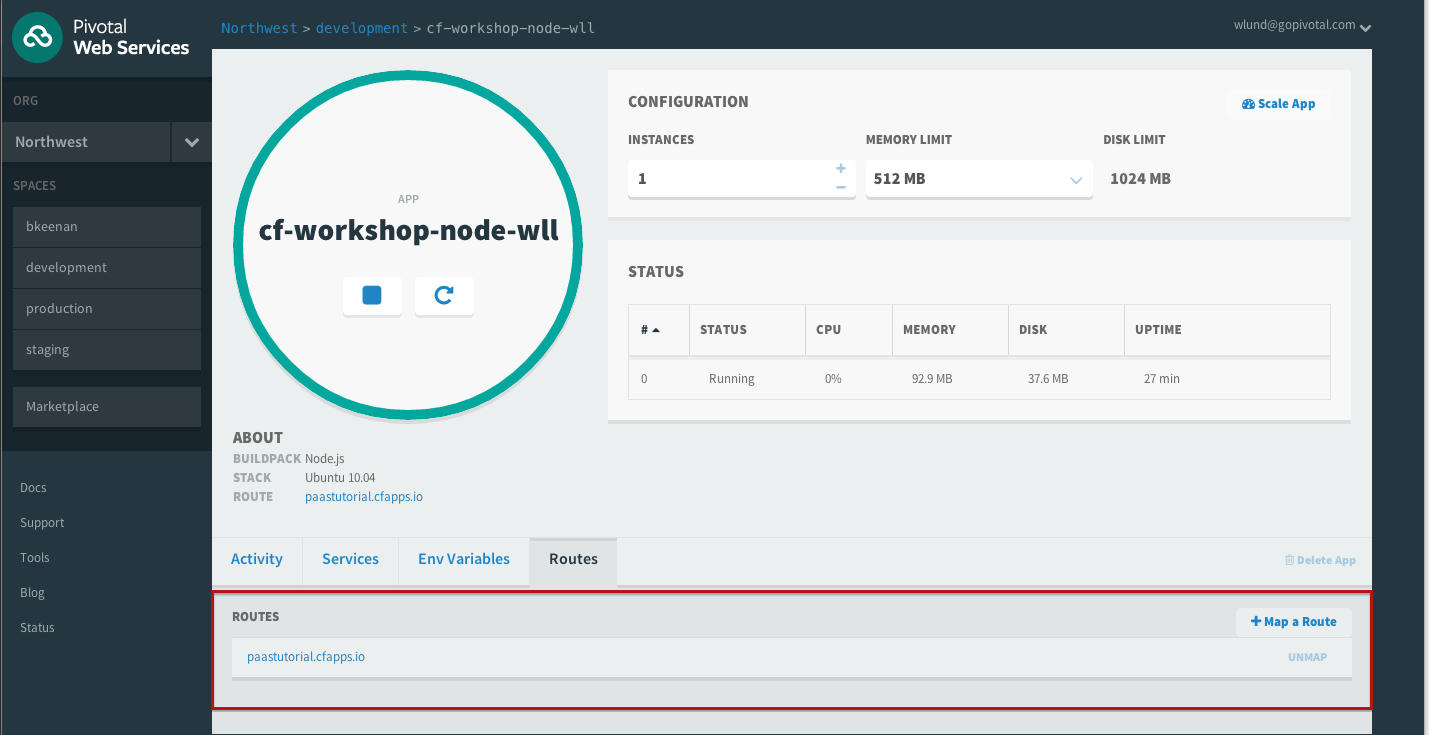
*cf push*

Visit <http://cf-workshop-node-userX.cfapps.io> to check it’s working.

1. At this point we have two different app versions – one with our PaaSTutorial and the other with cf-workshop-node, a more functional application – running and using different routes.

However, as we verified this new cf-workshop-node version is working fine, we’d like to add it to the original application route, splitting the traffic between the old and new versions.

On the Pivotal CF Web Console, select the space you’ve pushed the application to, and detail the app.



Click on “Routes” and add the original application route to it: <http://paastutorial-userX.cfapps.io/> . Click “Map a Route”.

As this route is already being used by the 1st application version (passtutorial-userX), the Cloud Foundry router will split the requests between the two applications.

Alternatively, the same can be done from the command-line:

*cf map-route* <http://cf-workshop-node-userX.cfapps.io> *-n paastutorial-userX*

After doing this, our original route will be load-balancing between 4 instances: 2 of the <http://paastutorial-userX.cfapps.io/> and 2 of the <http://cf-workshop-node-userX.cfapps.io> (50% each). We can change this proportion providing more instances of each application version if needed:

*cf scale cf-workshop-node-userX -i 4*

Now we would have 4 out of 6 requests going to the cf-workshop-node-userX version and only 2 to the paastutorial version. You can play with this hitting refresh on your browser (however browser can cache requests sometimes)

1. As we’re done testing the new version, remove the old application version from the original route, completing our rolling upgrade.

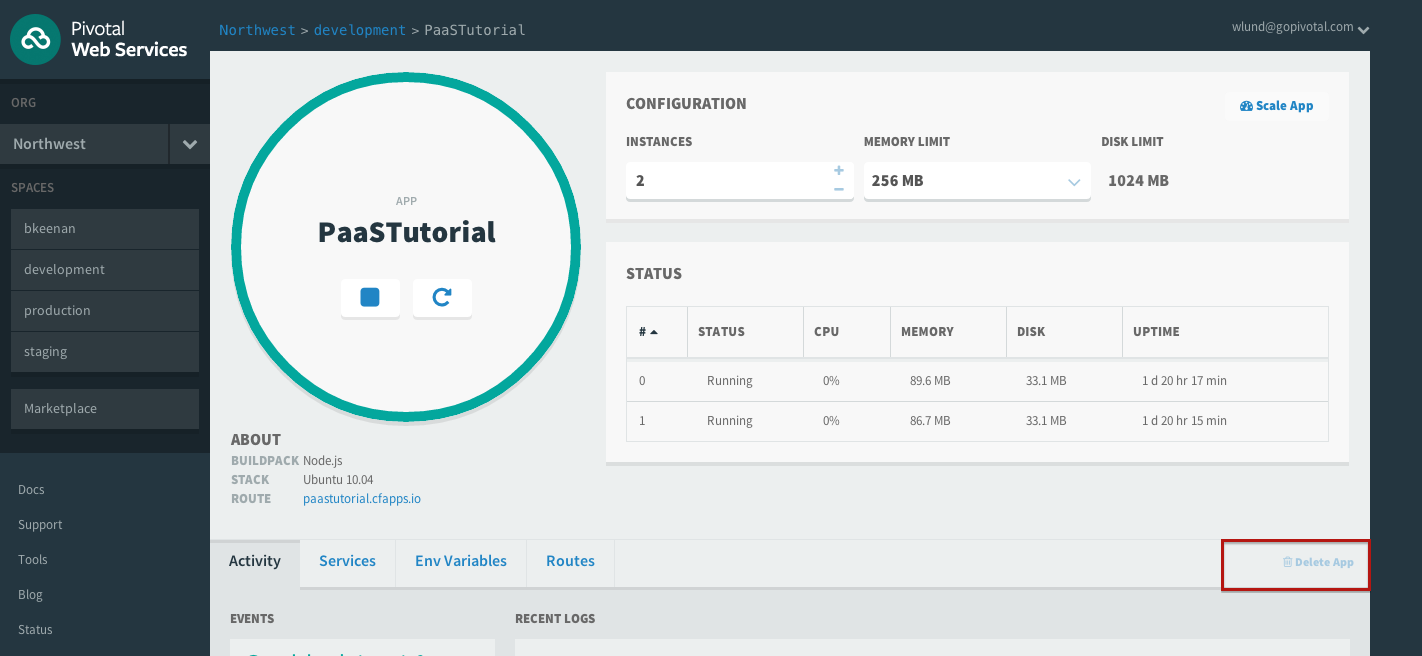
On the Web Console, detail the original PaaSTutorial application, and on the route, click “Unmap Route” from the routes tab.

Alternatively, you can do the same using the command line:

*cf unmap-route paastutorial-userX run.pivotal.io -n paastutorial-userX*

1. Now the route <http://paastutorial.cfapps.io.io> will only handle requests for the cf-workshop-node version of the app. The original version can then be deleted.

Just click “Delete App” when detailing the application through the Web Console.



You can also use the command line:

*cf delete paastutorial-userX*

**OR** you can just assign a new route for it and keep the app:

*cf map-route paastutorial-userX run.pivotal.io -n paastutorial-userX-old*