CLOUD FOUNDRY

Pivotal Cloud Foundry Developer

Lab Instructions

Application Deployment using Pivotal Cloud Foundry

Version 1.11.a

Pivotal

Copyright Notice

Pivotal

Table of Contents

Chapter 1. Blue Green

Estimated Time: 30 minutes

1.1. Preface

So you've pushed an app, and now it's time to deploy a new version. Blue-green deployments are a technique for deploying updates with zero downtime.

Cloudfoundry allows developers to manage everything about their application: from deployment to the management of routes to an application. It's thanks to this self-service philosophy that the task of achieving zero-downtime deployments becomes easy.

This lab will walk you through the steps to deploy a new version of an application with zero downtime, and provides a way to visualize how traffic gets routed to the new application.

1.2. Setup

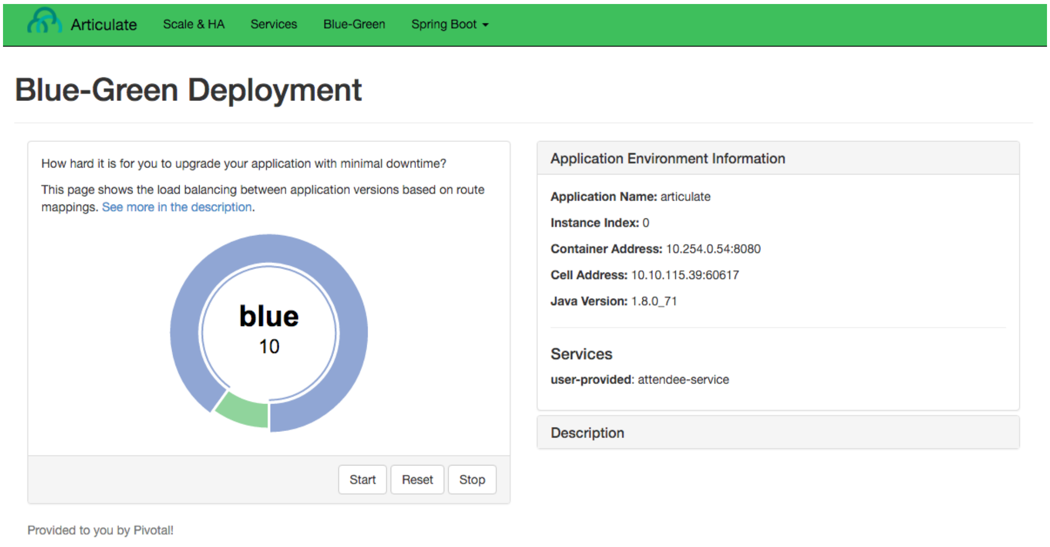
1. To simulate a blue-green deployment, first scale `articulate` to multiple instances.

```
cf scale articulate -i 2
```

1.3. Perform a Blue-Green Deployment

1. Read about using Blue-Green Deployments to reduce downtime and risk.

2. Browse to the `articulate` `Blue-Green` page.



Blue-Green Deployment

How hard it is for you to upgrade your application with minimal downtime?

This page shows the load balancing between application versions based on route mappings. See more in the description.

Application Environment Information

Application Name: articulate
Instance Index: 0
Container Address: 10.254.0.54:8080
Cell Address: 10.10.115.39:60617
Java Version: 1.8.0_71

Services

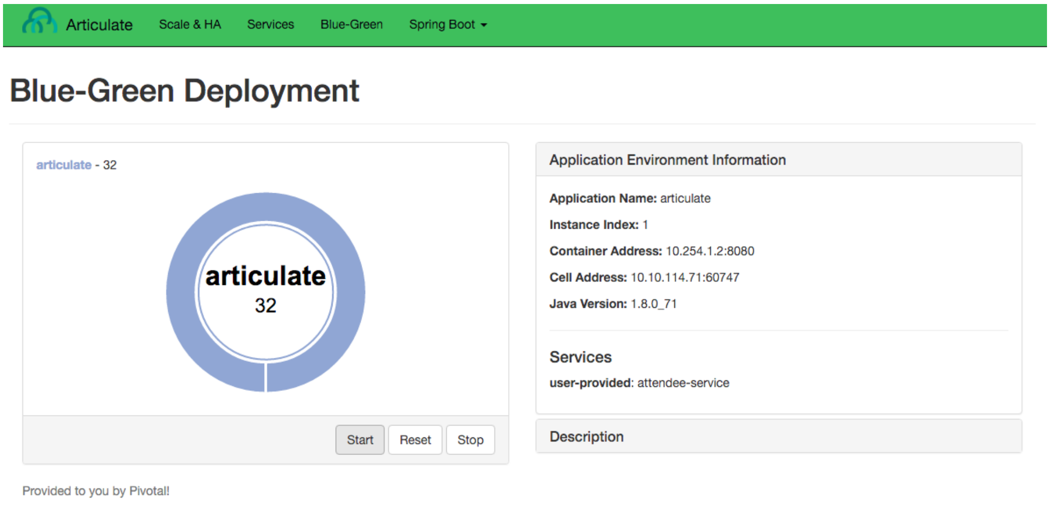
user-provided: attendee-service

Provided to you by Pivotal!

3. Lets assume that the deployed application is version 1. Let’s generate some traffic. Press the `Start` button.

   Leave this open as a dedicated tab in your browser. We will come back to this later.

4. Observe our existing application handling all the web requests.



Blue-Green Deployment

Application Environment Information

Application Name: articulate
Instance Index: 1
Container Address: 10.254.1.2:8080
Cell Address: 10.10.114.71:60747
Java Version: 1.8.0_71

Services

user-provided: attendee-service

5. Record the subdomain (`host`) for the `articulate` application.

   This is our production route. *You will use this in the next step.*

   For example:

```
cf routes

Getting routes as droberts@pivotal.io ...

space   host                                         domain                  apps
dev     articulate-heartsickening-elegance           pcfi1.fe.gopivotal.com   articulate
```

6. Now let’s `push` the next version of `articulate`.

   However, this time we will specify the subdomain by appending `-temp` to our production route.

   For example:

```
$> cd .../pivotal-cloud-foundry-developer-workshop/articulate/

$> cf push articulate-v2 -p ./articulate-0.2.jar -m 768M -n {{articulate_hostname_temp}} --no-start
```

7. Stop the `attendee-service` app to free up memory in your org.

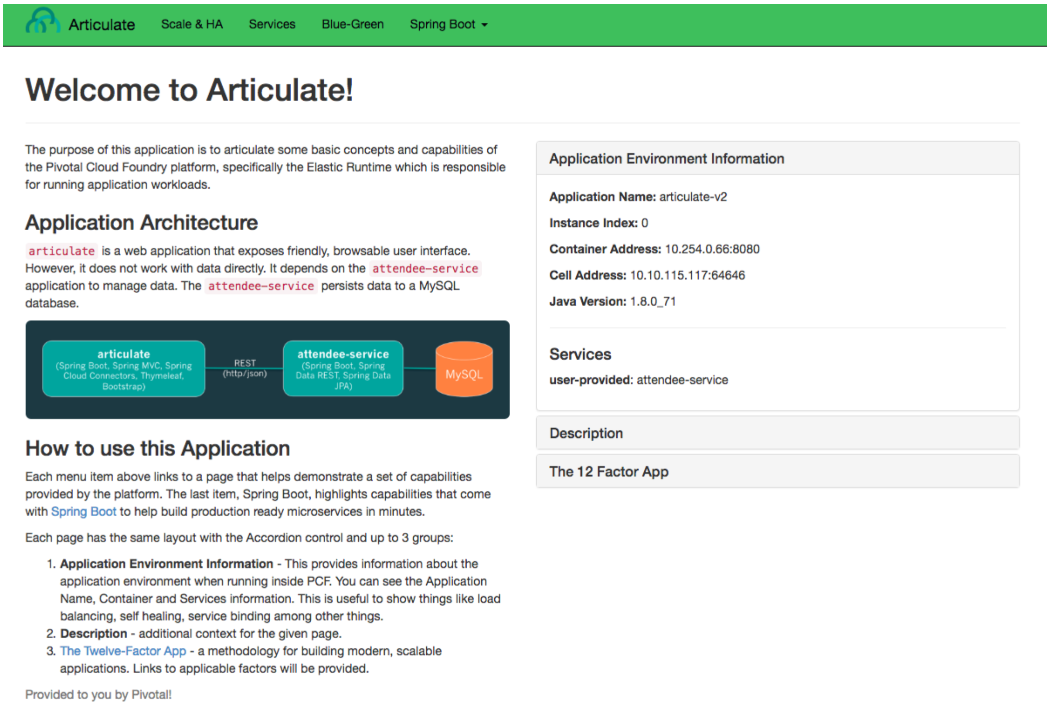
```
cf stop attendee-service
```

8. Start the new version of the `articulate-v2` app.

```
cf start articulate-v2
```

9. Now we have two versions of our app deployed.

   Open a new tab and view version 2 of `articulate` in your browser. Take note of the application name.



Welcome to Articulate!

The purpose of this application is to articulate some basic concepts and capabilities of the Pivotal Cloud Foundry platform, specifically the Elastic Runtime which is responsible for running application workloads.

Application Architecture

`articulate` is a web application that exposes friendly, browsable user interface. However, it does not work with data directly. It depends on the `attendee-service` application to manage data. The `attendee-service` persists data to a MySQL database.

How to use this Application

Each menu item above links to a page that helps demonstrate a set of capabilities provided by the platform. The last item, Spring Boot, highlights capabilities that come with Spring Boot to help build production ready microservices in minutes.

Each page has the same layout with the Accordion control and up to 3 groups:

1. **Application Environment Information** - This provides information about the application environment when running inside PCF. You can see the Application Name, Container and Services information. This is useful to show things like load balancing, self healing, service binding among other things.
2. **Description** - additional context for the given page.
3. The Twelve-Factor App - a methodology for building modern, scalable applications. Links to applicable factors will be provided.

Provided to you by Pivotal!

Application Environment Information

Application Name: articulate-v2
Instance Index: 0
Container Address: 10.254.0.66:8080
Cell Address: 10.10.115.117:64646
Java Version: 1.8.0_71

Services

user-provided: attendee-service

   At this point in the deployment process, you could do further testing of the version you are about to release before exposing customers to it.

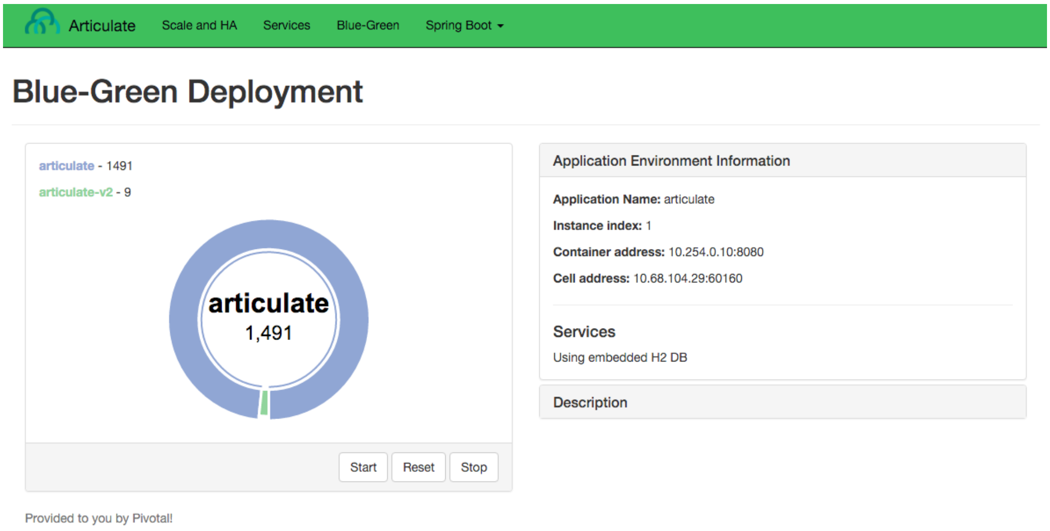
10. Let’s assume we are ready to start directing production traffic to version 2. We need to map our production route to `articulate-v2`.

    For example (your domain and subdomain will be different):

```
cf map-route articulate-v2 {{domain_name}} --hostname {{articulate_hostname}}
```

11. Return to browser tab where you started the load. You should see that it is starting to send requests to

version 2.



Blue-Green Deployment

articulate - 1491
articulate-v2 - 9

Application Environment Information

Application Name: articulate
Instance index: 1
Container address: 10.254.0.10:8080
Cell address: 10.68.104.29:60160

Services

Using embedded H2 DB

Description

Provided to you by Pivotal!

12.Press the `Reset` button, so we can see how the load get distributed across app instances.

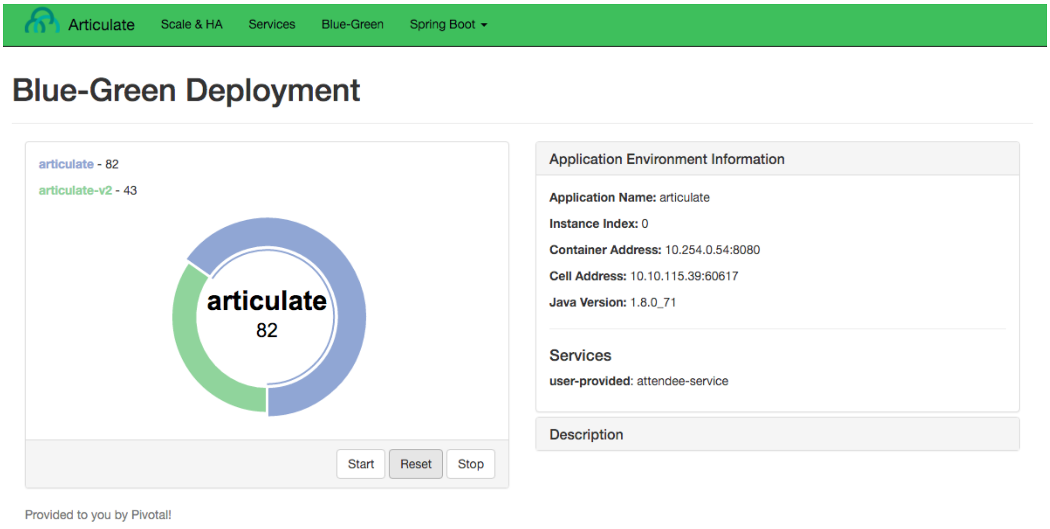
If you are running with a similar configuration to this:

```
cf apps

Getting apps in org dave / space dev as droberts@pivotal.io...
OK

name                    requested state   instances   memory   disk   urls
articulate              started           2/2         768M       1G     ...
articulate-v2           started           1/1         768M       1G     ...
```

You should see about a third of the requests going to version 2.



Blue-Green Deployment

articulate - 82
articulate-v2 - 43

Application Environment Information

Application Name: articulate
Instance Index: 0
Container Address: 10.254.0.54:8080
Cell Address: 10.10.115.39:60617
Java Version: 1.8.0_71

Services
user-provided: attendee-service

Description

Provided to you by Pivotal!

13.Move more traffic to version 2, with:

```
cf scale articulate -i 1
```

..and:

```
cf scale articulate-v2 -i 2
```

If you `Reset` the load generator, you will see 2/3 of the traffic go to `articulate-v2`.

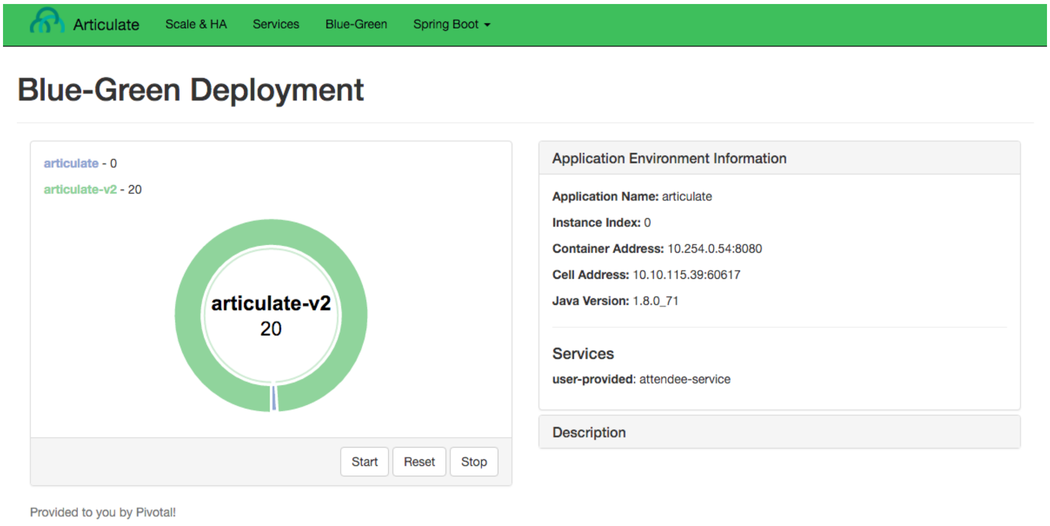
14.Move all traffic to version 2.

Remove the production route from the `articulate` application.

For example (your domain and subdomain will be different):

```
cf unmap-route articulate {{domain_name}} --hostname {{articulate_hostname}}
```

If you `Reset` the load generator, you will see all the traffic goes to `articulate-v2`.



Blue-Green Deployment

articulate - 0

articulate-v2 - 20

articulate-v2
20

Application Environment Information

Application Name: articulate

Instance Index: 0

Container Address: 10.254.0.54:8080

Cell Address: 10.10.115.39:60617

Java Version: 1.8.0_71

Services

user-provided: attendee-service

Description

Provided to you by Pivotal!

Note

Refreshing the entire page will update the application name.

15.Remove the temp route from the `articulate-v2` application.

For example (your domain and subdomain will be different):

```
cf unmap-route articulate-v2 {{domain_name}} --hostname {{articulate_hostname}}
```

Congratulations! You performed a blue-green deployment.

1.3.1. Questions

- How would a rollback situation be handled using a blue-green deployment?
- What other design implications does running at least two versions at the same time have on your applications?
- Do you do blue-green deployments today? How is this different?

1.4. Cleanup

Let’s reset our environment.

1. Delete the `articulate` application.

```
cf delete articulate
```

2. Rename `articulate-v2` to `articulate`.

```
cf rename articulate-v2 articulate
```

3. Restart `articulate`.

```
cf restart articulate
```

4. Scale down.

```
cf scale articulate -i 1
```

1.5. Explore Blue-Green Deployment Plugin

Now that we understand the mechanism by which we can perform blue-green deployments, let’s explore one of the cf cli plugins that automate some aspects of this deployment process.

1.5.1. Setup

1. Visit https://plugins.cloudfoundry.org/
2. Locate the *blue-green-deploy* plugin and follow instructions to install the plugin
3. Explore the project’s github repository README to learn how to use the plugin

1.5.2. Experiment

Let’s start again with deploying articulate in a blue-green fashion, but this time using the plugin:

1. Make sure you have a simple manifest file defined for your articulate application. Here’s an example:

manifest.yml.

```
---
applications:
- name: articulate
  path: target/articulate-0.0.2-SNAPSHOT.jar
  memory: 768M
  random-route: true
  services:
  - attendee-service
```

2. Instead of using the `push` command, deploy articulate using the blue-green-deploy command:

```
cf blue-green-deploy articulate
```

Observe what this command does:

1. it deploys articulate using a different application name and host name: `articulate-new`

Once the new version of the app is running..

1. the public route for the application is mapped to the new app
2. the previously deployed application is renamed using the '-old' suffix
3. the '-new' suffix is now dropped from the new application
4. the public route is unmapped from the old version of the application

All this is accomplished via the invocation of a single command!

We can take this a step further: by passing a smoke-test script to the `blue-green-deploy` command, the plugin will run the smoke tests and proceed to upgrade the application only on the condition that the smoke tests passed (returned with an exit code of 0). The plugin passes the fully-qualified domain name of the newly-deployed application as an argument to the smoke-test.

Here’s an updated blue-green deployment command that uses a simple health-check test for articulate:

```
cf blue-green-deploy articulate --smoke-test ./test-health.sh
```

See the articulate project source code on github for the complete details.

Congratulations! You have completed this lab.