|  |  |
| --- | --- |
| **cloudenabled** | |
| **Author** | **Sushil Kumar** |
| **Documentation Version** | **V1.0** |

**Pivotal Cloud Foundry-Developer**

**Pivotal Cloud Foundry Developer - Blue Green**

#### Table of Contents

* [What you will learn](https://content.enablement.pivotal.io/pivotal-cloud-foundry-developer/blue-green/index.html#what-you-will-learn:d680e8a854a7cbad6d490c445cba2eba)
* [Exercises](https://content.enablement.pivotal.io/pivotal-cloud-foundry-developer/blue-green/index.html#exercises:d680e8a854a7cbad6d490c445cba2eba)
  + [Setup](https://content.enablement.pivotal.io/pivotal-cloud-foundry-developer/blue-green/index.html#setup:d680e8a854a7cbad6d490c445cba2eba)
  + [Perform a Blue-Green Deployment](https://content.enablement.pivotal.io/pivotal-cloud-foundry-developer/blue-green/index.html#perform-a-blue-green-deployment:d680e8a854a7cbad6d490c445cba2eba)
    - [Questions](https://content.enablement.pivotal.io/pivotal-cloud-foundry-developer/blue-green/index.html#questions:d680e8a854a7cbad6d490c445cba2eba)
  + [Cleanup](https://content.enablement.pivotal.io/pivotal-cloud-foundry-developer/blue-green/index.html#cleanup:d680e8a854a7cbad6d490c445cba2eba)

# What you will learn

* How to manage an application upgrade with a blue-green deployment

# Exercises

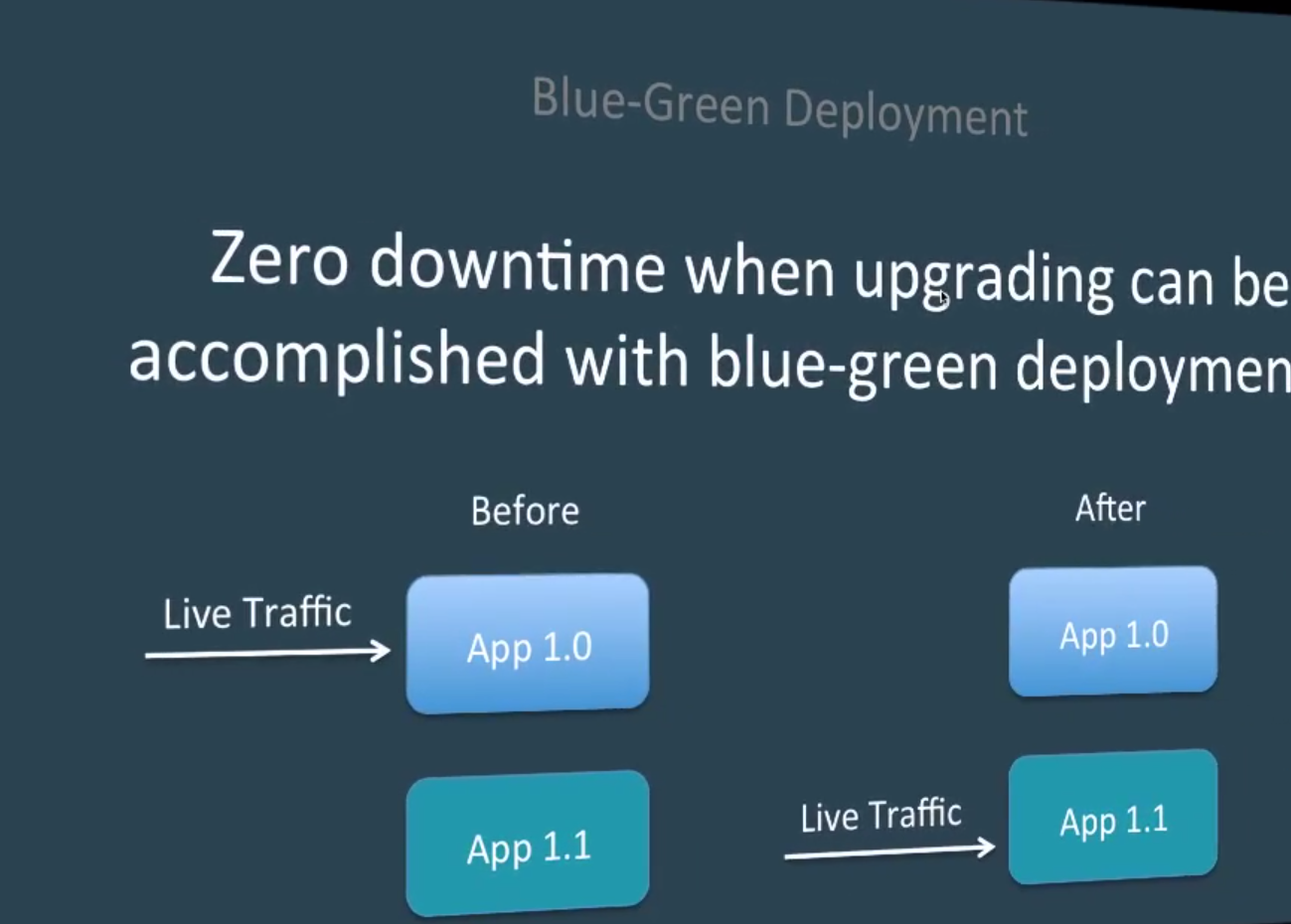
## Setup

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

$ cf scale articulate -i 2

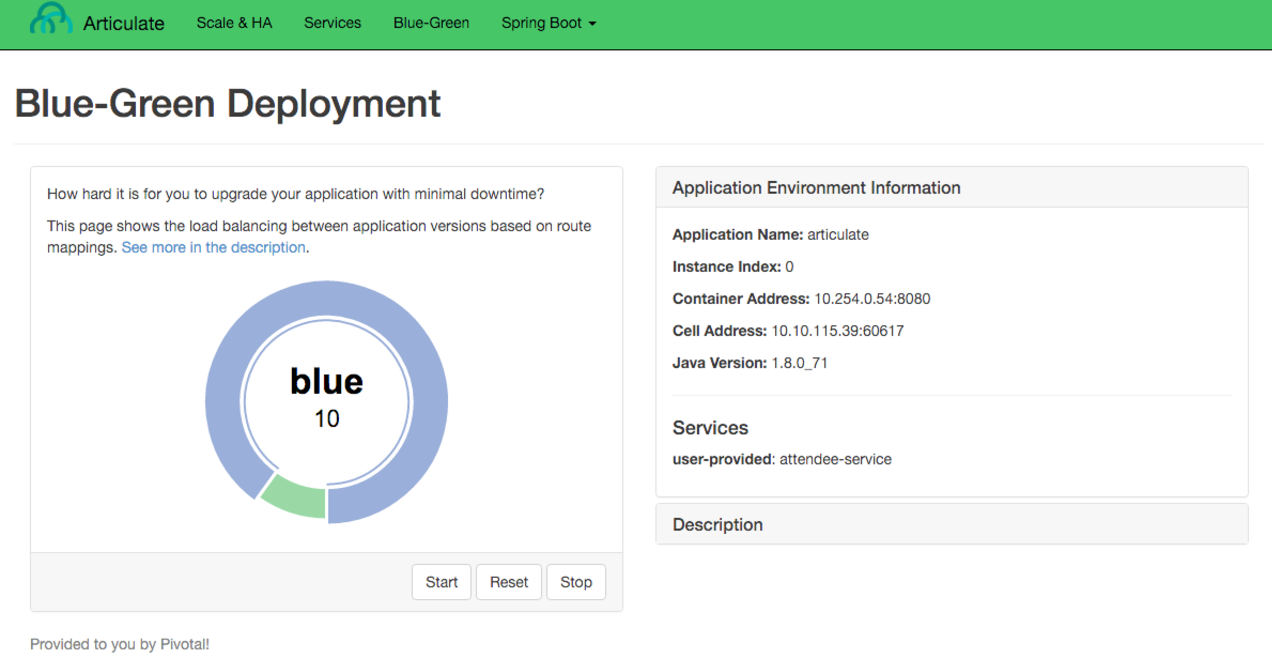
## Perform a Blue-Green Deployment

1) Read about using [Blue-Green Deployments to reduce downtime and risk](https://docs.pivotal.io/pivotalcf/devguide/deploy-apps/blue-green.html).





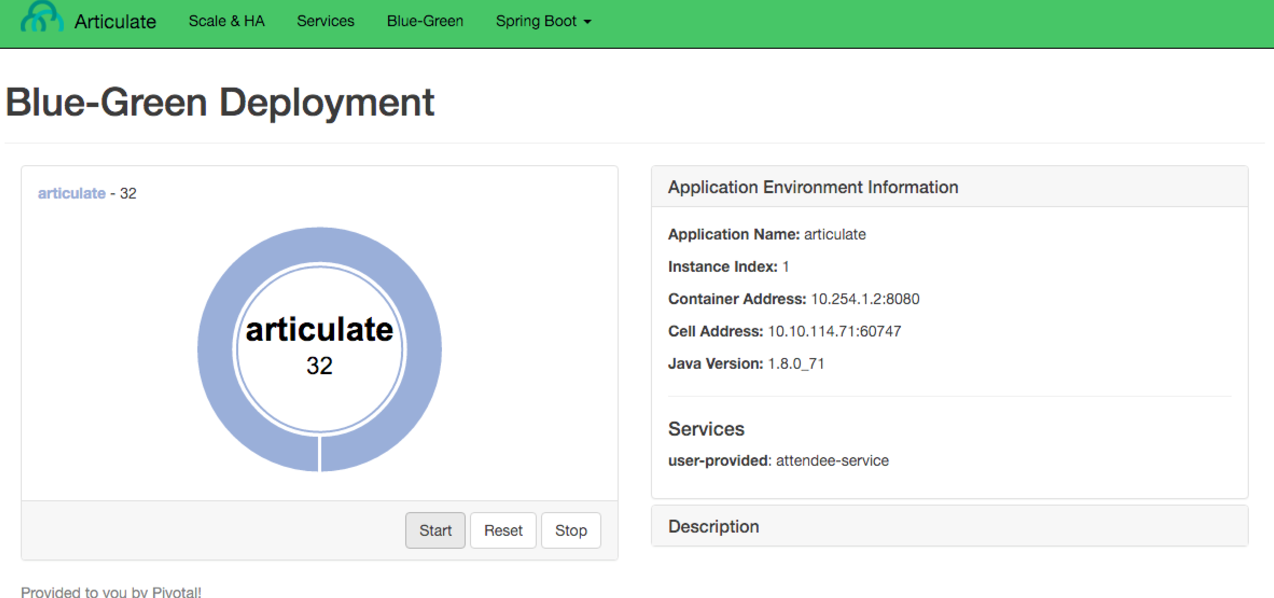
2) Browse to the articulate Blue-Green page.



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.



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 (your subdomain will be different):

$ cd ~/pivotal-cloud-foundry-developer-workshop/articulate/

$ cf push articulate-v2 -p ./articulate-0.0.1-SNAPSHOT.jar -m 512M -n articulate-heartsickening-elegance-temp --no-start

7) Bind articulate-v2 to the attendee-service user provided service.

$ cf **bind**-service articulate-v2 attendee-service

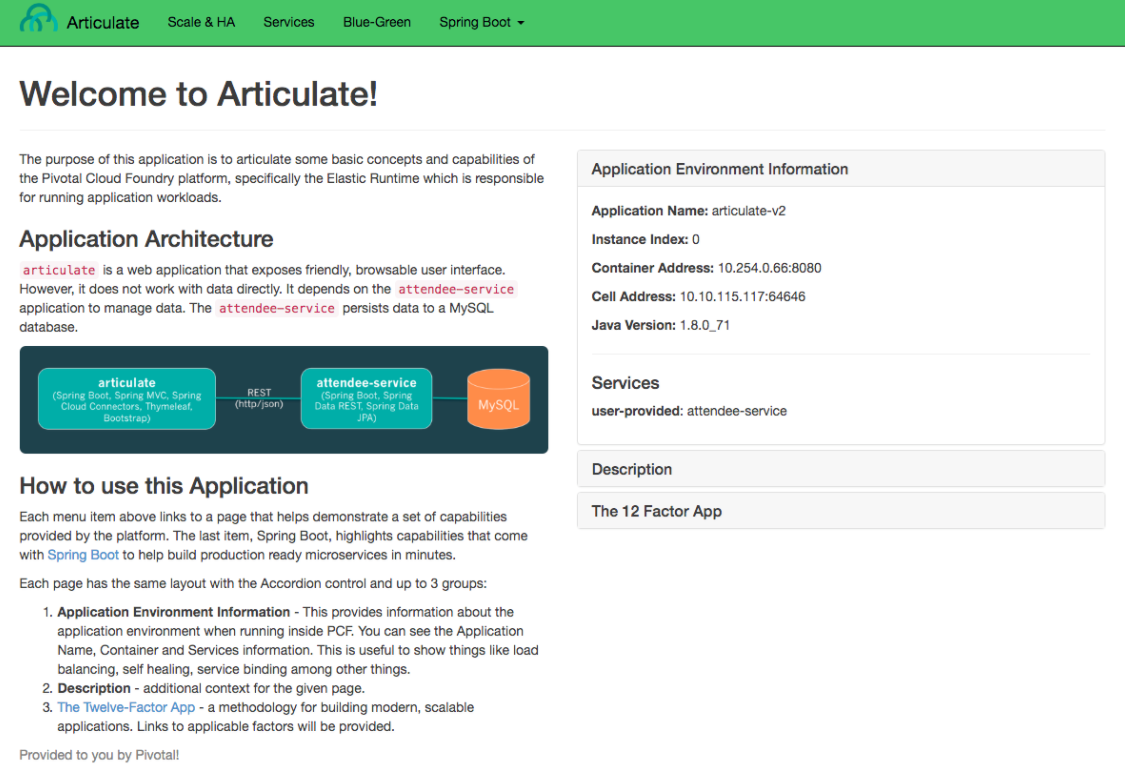
*You can ignore the “TIP: Use ‘cf restage articulate-v2’ to ensure your env variable changes take effect” message at this time.*

8) Start the application.

$ 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.



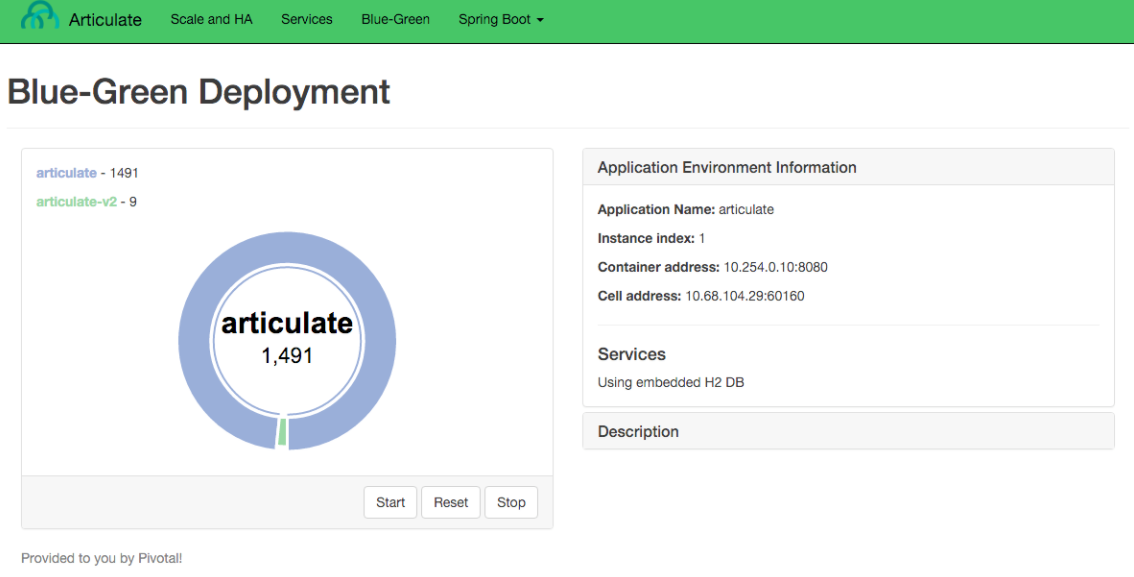
At this point in the deployment process, you could do further testing of the version you are about to release before exposing customers to it.

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 pcfi1.fe.gopivotal.com -n articulate-heartsickening-elegance

11) Return to browser tab where you started the load. You should see that it is starting to send requests to version 2.



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...

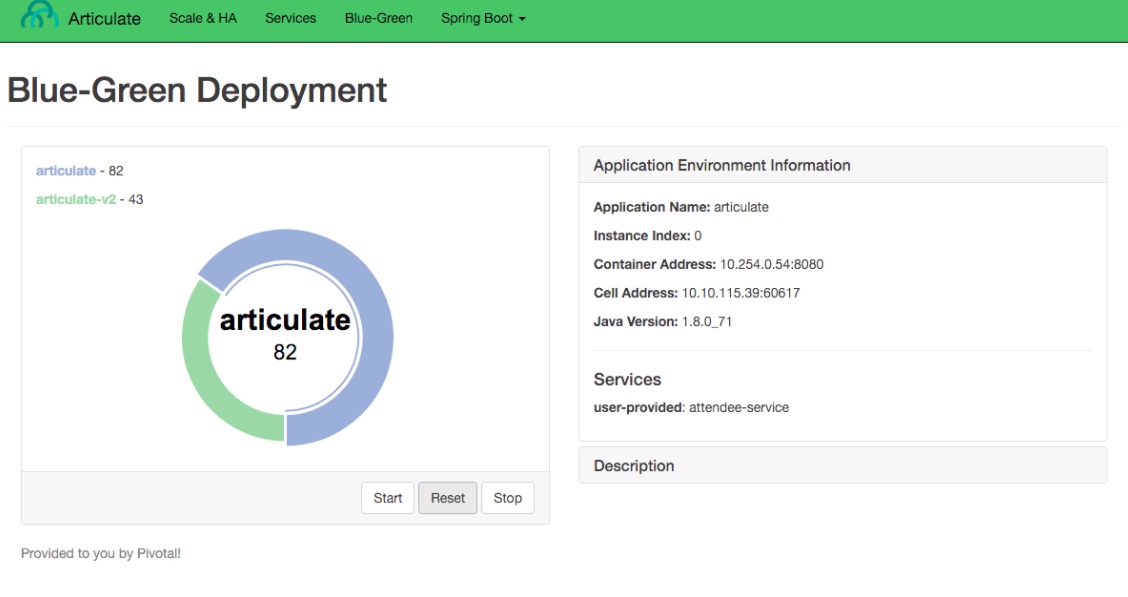
OK

name requested state instances memory disk urls

articulate started 2/2 512M 1G ...

articulate-v2 started 1/1 512M 1G ...

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



13) Move more traffic to version 2.

$ cf scale articulate -i 1

$ cf scale articulate-v2 -i 2

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

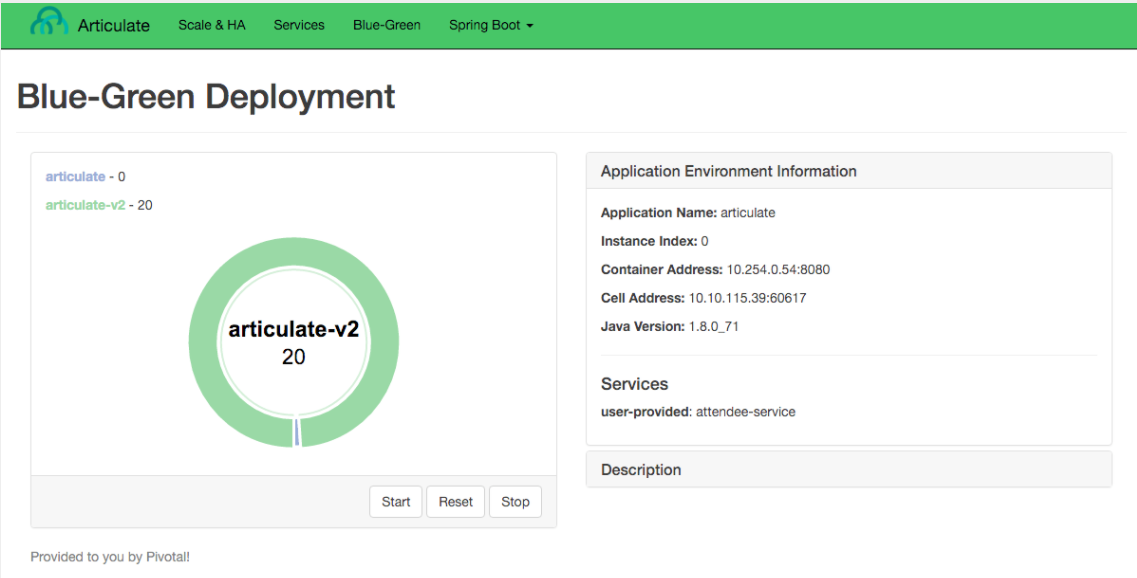
14) Move all traffic to version 2.

Remove the production route from the articulate application.

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

$ cf unmap-route articulate pcfi1.fe.gopivotal.com -n articulate-heartsickening-elegance

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



**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 pcfi1.fe.gopivotal.com -n articulate-heartsickening-elegance-temp

**Congratulations!** You performed a blue-green deployment.

### 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?

## 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