CLOUD FOUNDRY

Pivotal Cloud Foundry Developer

Lab Instructions

Application Deployment using Pivotal Cloud Foundry

Version 1.11.a

Pivotal

Copyright Notice

Table of Contents

Chapter 1. Application Autoscaler

Estimated Time: 25 minutes

1.1. Preface

So you can deploy your app, you can scale it. But what if we wish to automate scaling an application up and down during periods of higher and subsequently lower traffic. PCF allows us to automate scaling via the marketplace service named the *App Autoscaler*.

In this exercise we'll introduce a load against our application, and watch it automatically scale up to multiple instances, and then automatically scale back down to a specified minimum. Enjoy.

1.2. Setup

1. Ensure the `articulate` application has one application instance running.

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

1.3. Provision and Bind an Autoscaler Service Instance

1. Read the documentation about App Autoscaling.

2. Review what's in the marketplace.

```
cf marketplace
```

3. Create a autoscaler service instance.

```
cf create-service app-autoscaler standard autoscaler
```

4. Bind the service to `articulate`.

```
cf bind-service articulate autoscaler
```

Note

You can ignore the "TIP: Use 'cf restage articulate' to ensure your env variable changes take effect" message at this time.

5. Restart the application.

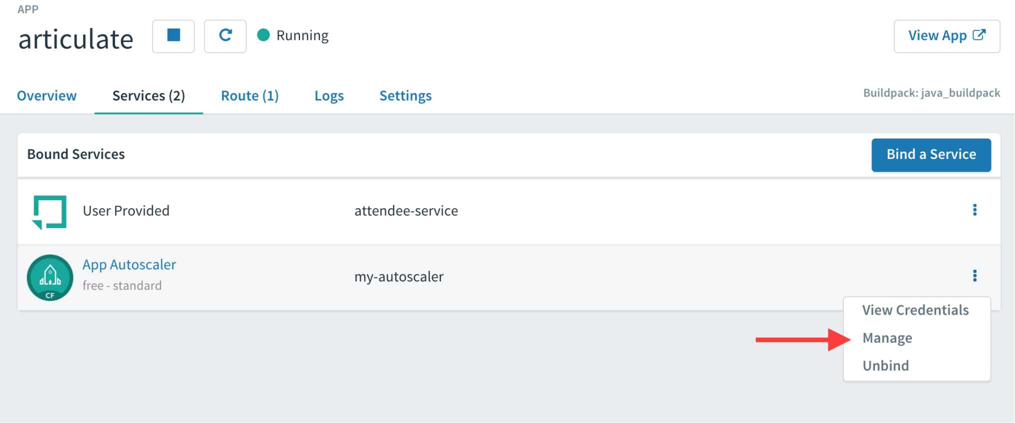
```
cf restart articulate
```

1.4. Configuring Autoscaling Service Instance

1. View `articulate` in Apps Manager.

    a. Navigate to the `Services` tab.

    b. Click the `Manage` link for the autoscaler (this will open a new tab).

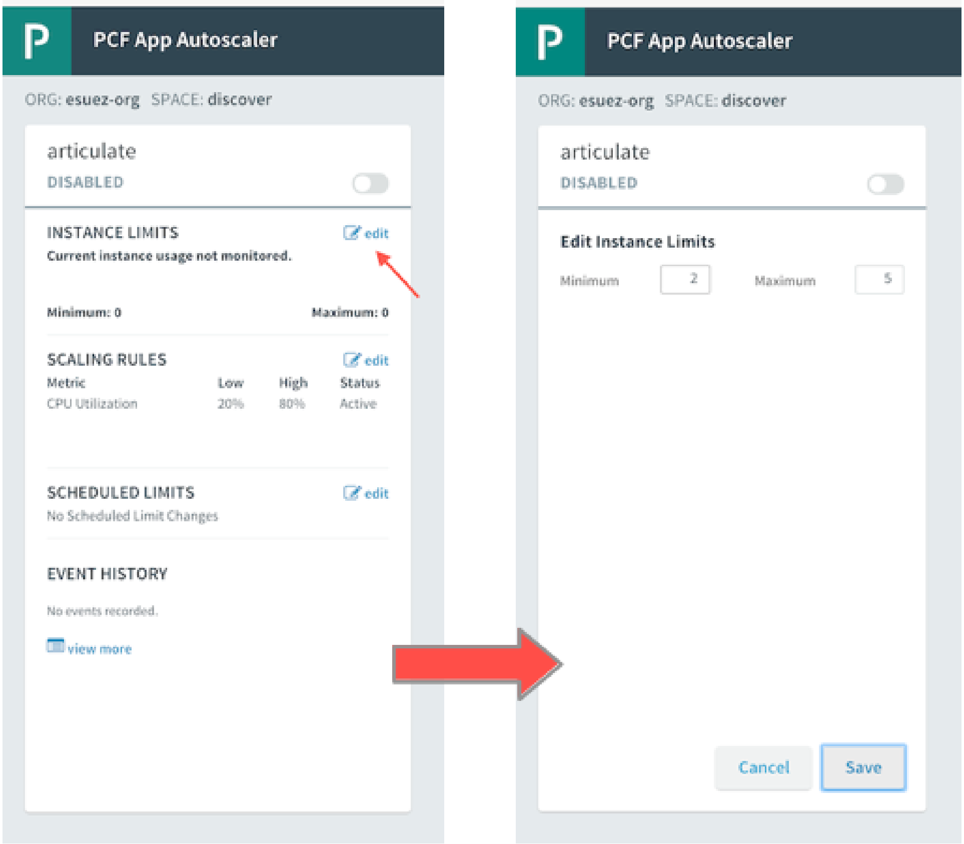


2. Edit instance limits

    a. Click 'Edit'

    b. Set the minimum number of instances to 2 and the maximum to 5

    c. Click 'Save'



PCF App Autoscaler

ORG: esuez-org SPACE: discover

articulate

DISABLED

INSTANCE LIMITS

Current instance usage not monitored.

Minimum: 0 Maximum: 0

SCALING RULES

Metric	Low	High	Status
CPU Utilization	20%	80%	Active

SCHEDULED LIMITS

No Scheduled Limit Changes

EVENT HISTORY

No events recorded.

PCF App Autoscaler

ORG: esuez-org SPACE: discover

articulate

DISABLED

Edit Instance Limits

Minimum 2 Maximum 5

3. Return to Apps Manager and observe the number of instances increase from `1` to `2`.

1.5. Generate Load and Observe the Results

One popular option for generating a load against a web site is by using the Apache load-testing tool JMeter. Instructions for JMeter are outlined below.

Note

Another option is to use a PWS marketplace service named *Load Impact*. One advantage of using Load Impact is that you don't have to download and install any software to your laptop in order to conduct a load test: it's all done via a web interface. Further, the load test itself is run from the cloud: the test specification can specify what geographic location will serve as the origin of the load test. If you feel like trying Load Impact instead of JMeter, head on over to the PWS Marketplace in your Apps Manager, and create an instance of that service, then follow the instructions for creating a test.

1.5.1. Generating a Load with JMeter

1. You should already have `apache-jmeter-XXX.zip` in your lab files, where XXX is the version number. (if you prefer you can download the latest Apache JMeter for yourself). It will be used to generate load. Review the JMeter getting started directions.

2. Do *ONE* of the following:

   Linux and Mac

   a. Add execute permissions to the JMeter script: `chmod +x {{jmeter_basedir}}/bin/jmeter.sh`

   b. Start JMeter by executing `{{jmeter_basedir}}/bin/jmeter.sh`

   Windows

   a. Start JMeter by double clicking the following file in Windows Explorer: `{{jmeter_basedir}}/bin/jmeter.bat`

3. In your lab files you will find a JMeter configuration file: `load-gen.jmx`.

4. Import the `load-gen.jmx` file with the JMeter GUI (`File` -> `Open`).

5. Expand the content of the JMeter test plan by:

   a. Selecting the `Load Generator` test plan on the left pane.

   b. Going to the menu: `Options` -> `Expand All`.

6. Under `HTTP Request`, edit the test plan as follows:

   a. Set the value of the `Service Name or IP` field to your application's fully-qualified hostname (do not include the `http://`). For example: `jsmith-articulate.cfapps.io`

b. Set the value of the HTTP Request `Path` field to `/services`

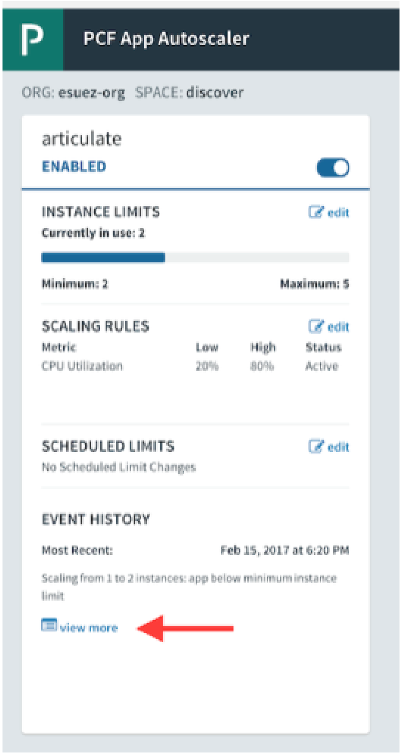
7. Save the test plan: `File` -> `Save`.

8. Run the test plan: `Run` -> `Start`.

9. Use Apps Manager and the `cf` CLI to observe your service scale up and back down based on load.

10. Stop the test plan: `Run` -> `Stop`.

11. Review Autoscale history:



PCF App Autoscaler

ORG: esuez-org SPACE: discover

articulate

ENABLED

INSTANCE LIMITS

Currently in use: 2

Minimum: 2 Maximum: 5

SCALING RULES

Metric	Low	High	Status
CPU Utilization	20%	80%	Active

SCHEDULED LIMITS

No Scheduled Limit Changes

EVENT HISTORY

Most Recent: Feb 15, 2017 at 6:20 PM

Scaling from 1 to 2 instances: app below minimum instance limit

Figure 1.1. History

Note

An even better option for monitoring the state of your application, including your application instances and their CPU usage, is to install the cf cli plugin **cf top**. Head on over to the cloud foundry plugins catalog, locate the plugin named 'top' and install it. It’s definitely worth your time.

1.6. Clean up

1. Unbind the `autoscaler` service instance.

```
cf unbind-service articulate autoscaler
```

2. Delete the `autoscaler` service instance.

```
cf delete-service autoscaler
```

3. Scale `articulate` back to original settings.

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

4. Restart `articulate`.

```
cf restart articulate
```

1.7. Questions

- How do you handle autoscaling today?
- What 12 factor principles are important when it comes to scaling?
- How do you handle scaling at the data layer?