

# PIVOTAL CLOUD FOUNDRY

## *LAB ASSIGNMENTS*

## Prerequisites:

1. Use Pivotal Web Services (PWS). Go to <https://console.run.pivotal.io/register> and go through the registration process for a trial account. If you already have a PWS account you can skip this step.
2. Install Cloud Foundry Command Line Interface (CLI) Go to <https://github.com/cloudfoundry/cli/releases> or <https://console.run.pivotal.io/tools> and download the installer for your platform. Run the installer.

## Lab 1:

### Deploy an application using the CLI (Command Line Interface)

1. Push a simple Java/Spring application to Cloud Foundry
2. Observe and scale a running application.
3. You need to know the three URLs for either your Cloud Foundry installation or for PWS. Refer back to the presentation for details on the System Domain, Apps Domain and Apps Manager URLs.

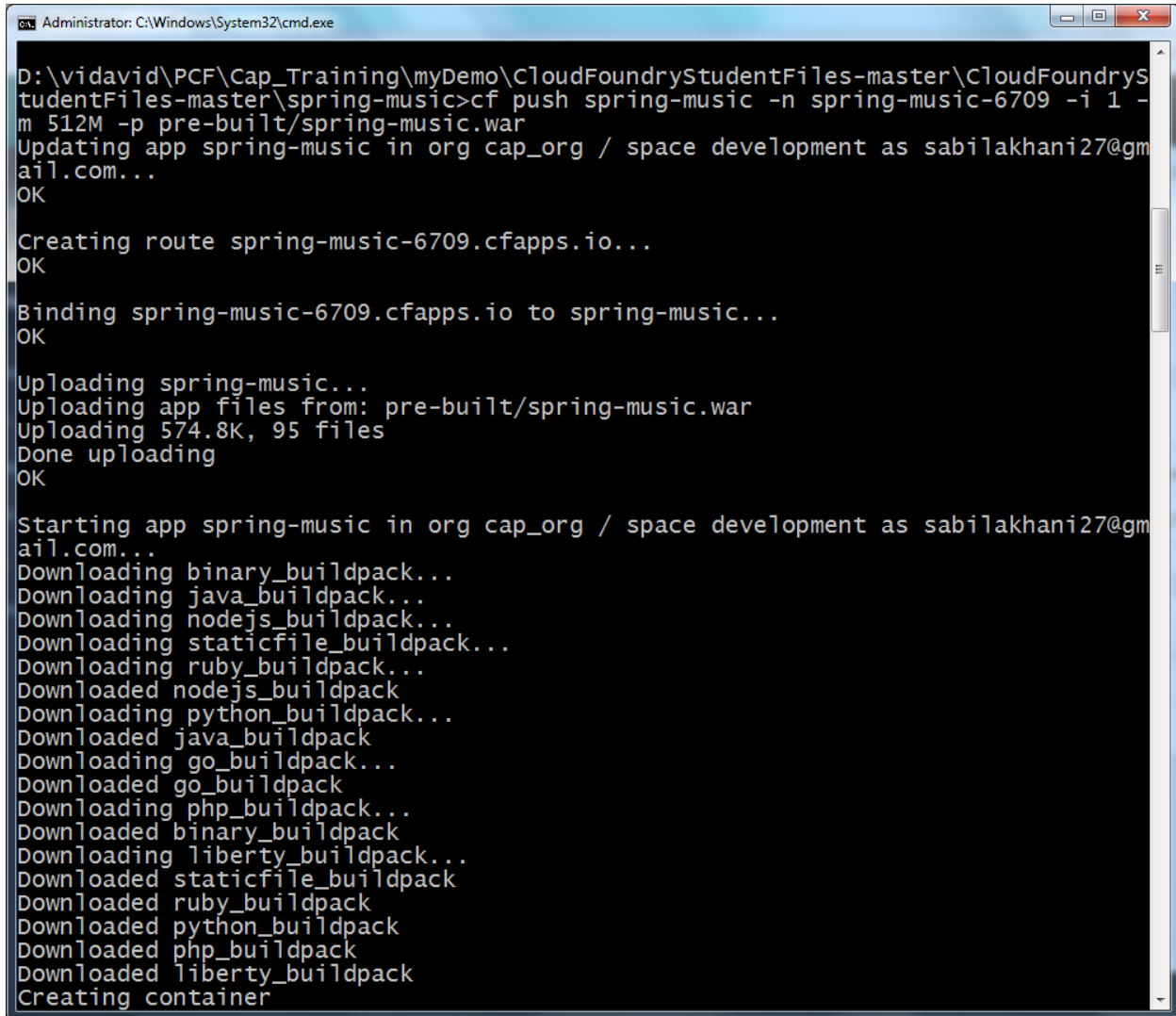
## Steps:

1. Download the Spring-Music file.
2. In command prompt, locate the unzip folder (spring-music application) location.
3. Enter below command to push your application into Pivotal.

```
cf push spring-music -n spring-music-6709 -i 1 -m 512M -p pre-built/spring-music.war
```

4. Once you enter the above command in the command prompt you will be getting the following information, finally you could see the deployed application in Pivotal.

**Note :** If you get any error related to host number, please change into any other number like 6708, 6701 , etc.



```
Administrator: C:\Windows\System32\cmd.exe
D:\vidavid\PCF\Cap_Training\myDemo\CloudFoundryStudentFiles-master\CloudFoundryStudentFiles-master\spring-music>cf push spring-music -n spring-music-6709 -i 1 -m 512M -p pre-built/spring-music.war
Updating app spring-music in org cap_org / space development as sabilakhani27@gmail.com...
OK

Creating route spring-music-6709.cfapps.io...
OK

Binding spring-music-6709.cfapps.io to spring-music...
OK

Uploading spring-music...
Uploading app files from: pre-built/spring-music.war
Uploading 574.8K, 95 files
Done uploading
OK

Starting app spring-music in org cap_org / space development as sabilakhani27@gmail.com...
Downloading binary_buildpack...
Downloading java_buildpack...
Downloading nodejs_buildpack...
Downloading staticfile_buildpack...
Downloading ruby_buildpack...
Downloaded nodejs_buildpack
Downloading python_buildpack...
Downloaded java_buildpack
Downloading go_buildpack...
Downloaded go_buildpack
Downloading php_buildpack...
Downloaded binary_buildpack
Downloading liberty_buildpack...
Downloaded staticfile_buildpack
Downloaded ruby_buildpack
Downloaded python_buildpack
Downloaded php_buildpack
Downloaded liberty_buildpack
Creating container
```

```
Administrator: C:\Windows\System32\cmd.exe
Downloaded liberty_buildpack
Creating container
Successfully created container
Downloading app package...
Downloaded app package (21M)
Staging...
-----> Java Buildpack Version: v3.8.1 (offline) | https://github.com/cloudfoundry/java-buildpack.git#29c79f2
-----> Downloading Open Jdk JRE 1.8.0_91-unlimited-crypto from https://java-buildpack.cloudfoundry.org/openjdk/trusty/x86_64/openjdk-1.8.0_91-unlimited-crypto.tar.gz (found in cache)
    Expanding Open Jdk JRE to .java-buildpack/open_jdk_jre (1.0s)
-----> Downloading Open JDK Like Memory Calculator 2.0.2_RELEASE from https://java-buildpack.cloudfoundry.org/memory-calculator/trusty/x86_64/memory-calculator-2.0.2_RELEASE.tar.gz (found in cache)
    Memory Settings: -Xss228K -Xmx317161K -XX:MaxMetaspaceSize=64M -Xms317161K -XX:MetaspaceSize=64M
-----> Downloading Spring Auto Reconfiguration 1.10.0_RELEASE from https://java-buildpack.cloudfoundry.org/auto-reconfiguration/auto-reconfiguration-1.10.0_RELEASE.jar (found in cache)
-----> Downloading Tomcat Instance 8.0.36 from https://java-buildpack.cloudfoundry.org/tomcat/tomcat-8.0.36.tar.gz (found in cache)
    Expanding Tomcat Instance to .java-buildpack/tomcat (0.1s)
-----> Downloading Tomcat Lifecycle Support 2.5.0_RELEASE from https://java-buildpack.cloudfoundry.org/tomcat-lifecycle-support/tomcat-lifecycle-support-2.5.0_RELEASE.jar (found in cache)
-----> Downloading Tomcat Logging Support 2.5.0_RELEASE from https://java-buildpack.cloudfoundry.org/tomcat-logging-support/tomcat-logging-support-2.5.0_RELEASE.jar (found in cache)
-----> Downloading Tomcat Access Logging Support 2.5.0_RELEASE from https://java-buildpack.cloudfoundry.org/tomcat-access-logging-support/tomcat-access-logging-support-2.5.0_RELEASE.jar (found in cache)
Exit status 0
Staging complete
Uploading droplet, build artifacts cache...
Uploading build artifacts cache...
Uploading droplet...
Uploaded build artifacts cache (109B)
Uploaded droplet (74M)
Uploading complete
Destroying container
```

```
Administrator: C:\Windows\System32\cmd.exe
Uploading complete
Destroying container
Successfully destroyed container

0 of 1 instances running, 1 starting
1 of 1 instances running

App started

OK

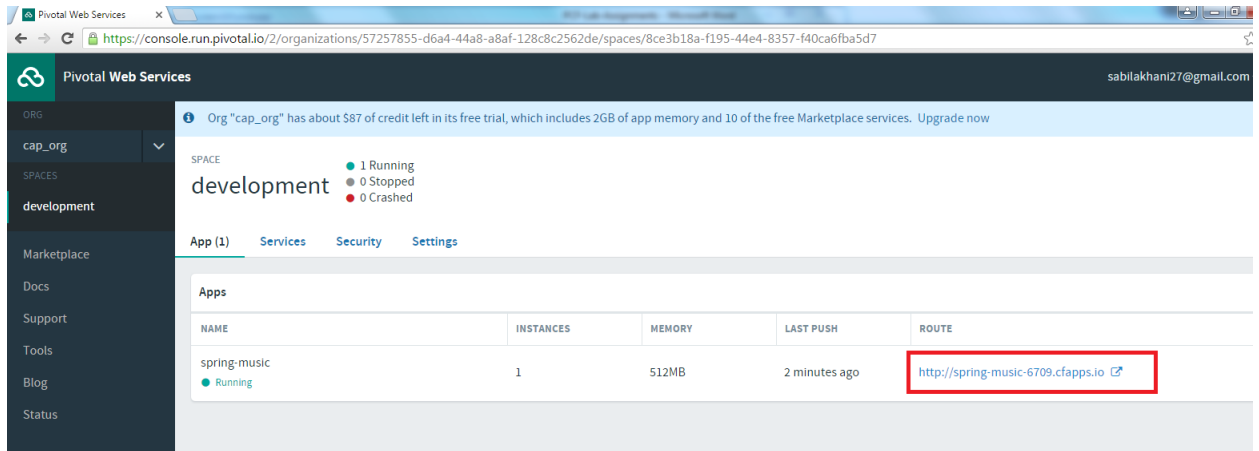
App spring-music was started using this command `CALCULATED_MEMORY=$(($PWD/.java-buildpack/open_jdk_jre/bin/java-buildpack-memory-calculator-2.0.2_RELEASE -memorySizes=metaspace:64m..,stack:228k.. -memoryWeights=heap:65,metaspace:10,native:15,stack:10 -memoryInitials=heap:100%,metaspace:100% -stackThreads=300 -totMemory=$MEMORY_LIMIT) && JAVA_HOME=$PWD/.java-buildpack/open_jdk_jre JAVA_OPTS="-Djava.io.tmpdir=$TMPDIR -XX:OnOutOfMemoryError=$PWD/.java-buildpack/open_jdk_jre/bin/killjava.sh $CALCULATED_MEMORY -Daccess.logging.enabled=false -Dhttp.port=$PORT" exec $PWD/.java-buildpack/tomcat/bin/catalina.sh run`

Showing health and status for app spring-music in org cap_org / space development as sabilakhani27@gmail.com...
OK

requested state: started
instances: 1/1
usage: 512M x 1 instances
urls: spring-music-6709.cfapps.io
last uploaded: Sun Aug 7 05:22:00 UTC 2016
stack: cflinuxfs2
buildpack: java-buildpack=v3.8.1-offline-https://github.com/cloudfoundry/java-buildpack.git#29c79f2 open-jdk-like-jre=1.8.0_91-unlimited-crypto open-jdk-like-memory-calculator=2.0.2_RELEASE spring-auto-reconfiguration=1.10.0_RELEASE tomcat-access-logging-support=...

state      since      cpu      memory      disk
details
#0  running  2016-08-07 10:52:56 AM  174.2%  347.9M of 512M  155.2M of 1G
```

- You can see the deployed application in the Pivotal Web services Browser also. As mentioned below screen shot. Your application host URL is highlighted in the below; if you click the URL it will open the Spring-music application which has been deployed in Pivotal Cloud. ☺



## Push

Push the Spring Music application.

1. Change folders to the location where you downloaded the **Spring-Music** and go into the spring-music folder.
2. Run the `cf push` command to push this application to Cloud Foundry
  - a. Use any value you like for the application name. We suggest something brief and easy to type.
  - b. Use the `-n` option to specify the host. This value must be unique within the domain used by Cloud Foundry.
  - c. Use the `-p` option to specify the package to upload. This is a Java application, and a WAR file has already been built for you. It is located in “pre-built/spring-music.war”.
3. Observe the push log output. Can you identify the different stages of the process?

## Observe

1. When the process is complete, identify the URL of the application. Open a browser and connect to this URL to make sure that the application comes up.
2. Run the `cf logs` command for your app.
3. Return to the browser, click the links, make changes to the data, then return to the command line and observe the generated log activity.

### Conclusion:

1. Open the Apps Manager associated with your Pivotal CF instance. For Pivotal Web Services, this is at <http://run.pivotal.io>. For a private CF installation you will need to find out what the System Domain URL is - this is installation dependent. Once you know the System Domain, then the Apps Manager URI is `console.<system-domain>`.
2. Login and navigate to the organization and space that your application was pushed to.
3. Open the details page associated with your application. Change the number of instances to 4 and save your changes. From the browser, refresh the application's web page repeatedly and notice that the application does not go down while it is being scaled. Return to the command prompt and observe the activity that occurs as Cloud Foundry scales your application horizontally.
4. Back on the Apps Manager, scroll down to the "instances" section to observe the running instances.
5. Scale back to 1 instance.
6. Stop application, once you observed all the changes.

**Congratulations, you have completed this exercise!**

## Prerequisites:

1. Use **Pivotal Web Services (PWS)**. Go to <https://console.run.pivotal.io/register> and go through the registration process for a trial account. If you already have a PWS account you can skip this step.
2. **Install an IDE (Eclipse or Spring Tool Suite)**  
Some of the exercises work best when run from an IDE, such as Eclipse or Spring Tool Suite (STS), but again if you are not familiar with these tools, you may prefer to simply use the CLI and your favorite editor (such as Wordpad, Notepad++, JEdit, TextWrangler, XCode, gEdit or even `vi` or `emacs`). If so you are done, please ignore the rest of these instructions.
  - a. For Eclipse, Go to <https://www.eclipse.org/downloads/> and download Eclipse. Select the latest version, download it, and install it. If you already have Eclipse 4.3 or higher, you can skip this step.
  - b. For STS Go to <http://spring.io/tools/sts> and download STS. Select the latest version. If you already have STS version 3.6.0 or higher, you can skip this step.

## Lab 2:

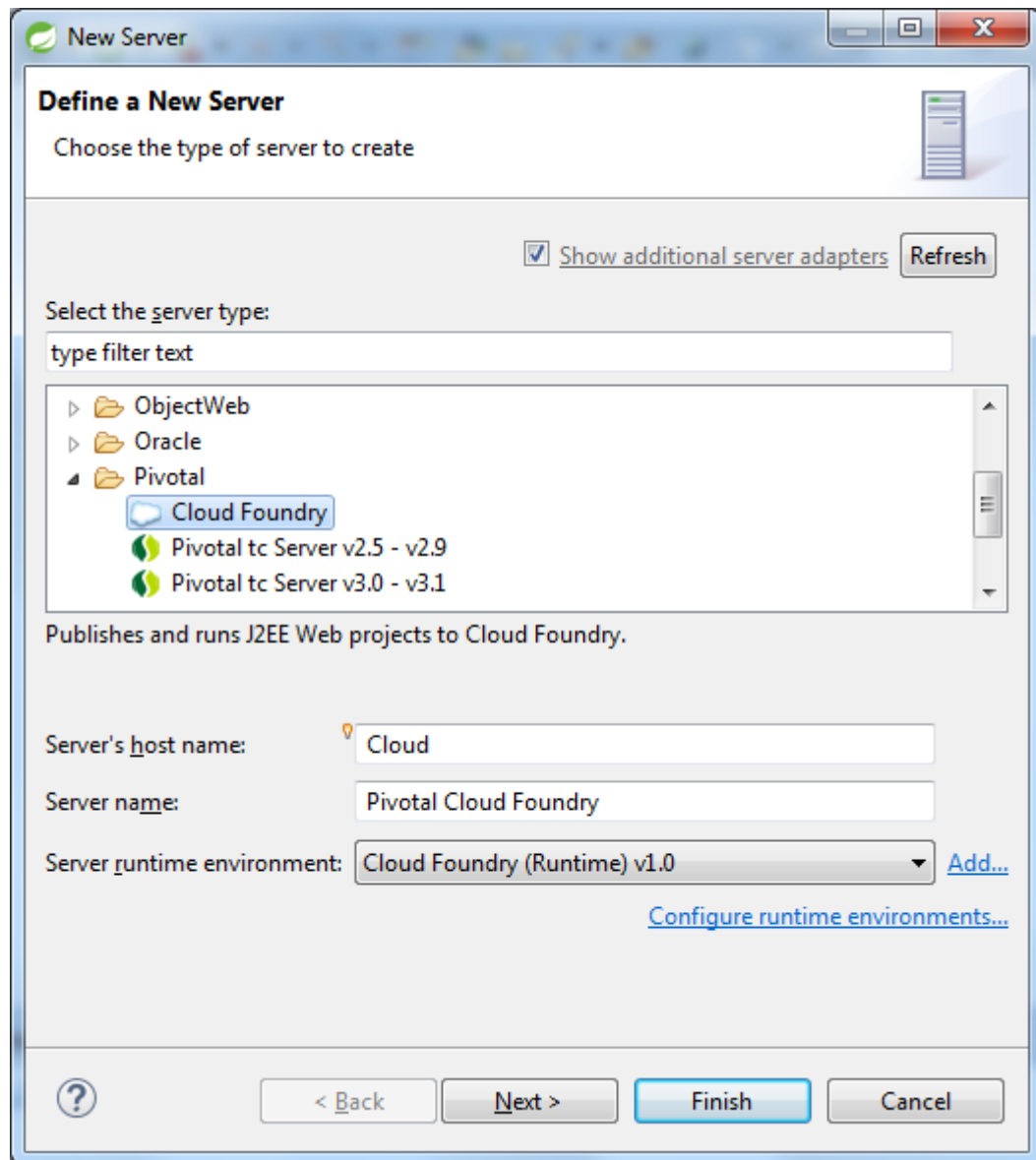
### Deploy an application using the IDE(STS / Eclipse)

#### Steps:

3. **Install IDE Support for Cloud Foundry**
  - a. Eclipse / STS: Install Cloud Foundry Integration
    - i. If using Eclipse or STS, select Eclipse Marketplace from the Help menu.
    - ii. In the marketplace dialog, enter "Cloud Foundry" in the Find field, and click Go.
    - iii. Find the entry for "Cloud Foundry Integration for Eclipse" in the search results. The most recent versions of STS have Cloud Foundry installed by default. There are three possibilities:
      - A. The Cloud Foundry plugin is not installed, click the Install button now.
      - B.If Cloud Foundry is already installed there will be two buttons Update and Uninstall. If the Update button is disabled there is nothing more to do. Go to the next section "Add a "Cloud Foundry" Server" below.
      - C.If the Update button is enabled, you may wish to update now. Click Update to continue. If you do not wish to update, go to the next section "Add a "Cloud Foundry" Server" below.
    - iv. In the next dialog just click Confirm
    - v. Finally, accept the license agreement and click Finish. The install/update process will take a bit of time, and you will be prompted to restart Eclipse to make the changes take effect.



- vi. Full details: Install to Eclipse
- 4. Add a “Cloud Foundry” Server
  - a. Eclipse or STS
    - i. Within Eclipse or STS, Select the “Servers” view; this is typically located in the lower left corner of the workspace. (If it is not present, go to Window / Show View / Other / Server / Servers.)
    - ii. Within the Servers view, right click, select New / Server and browse to Pivotal / Cloud Foundry. Enter the email and password for your account.
    - iii. By default the URL specifies PWS (run.pivotal.io). If using a different installation, click on “Manage Cloud”, then “Add” and then specify a name for your CF instance (call it whatever you want) and its API URL. If you are not sure of the API URL, you will need to obtain it from an administrator familiar with your CF setup.
    - iv. If prompted, it is fine to take the default values for Organization / Space or make a selection from the choices offered. If unsure, accept the defaults. We will explain "organization" and "space" more fully later in the course.
    - v. Click Finish. We now have a ‘server’ in Eclipse/STS that points to a Cloud Foundry environment.
    - vi. Look at the below screen shots for your guidance:



**New Server**

## Cloud Foundry Account

Press 'Validate Account', 'Next', 'Finish' to validate credentials.

**Account Information**

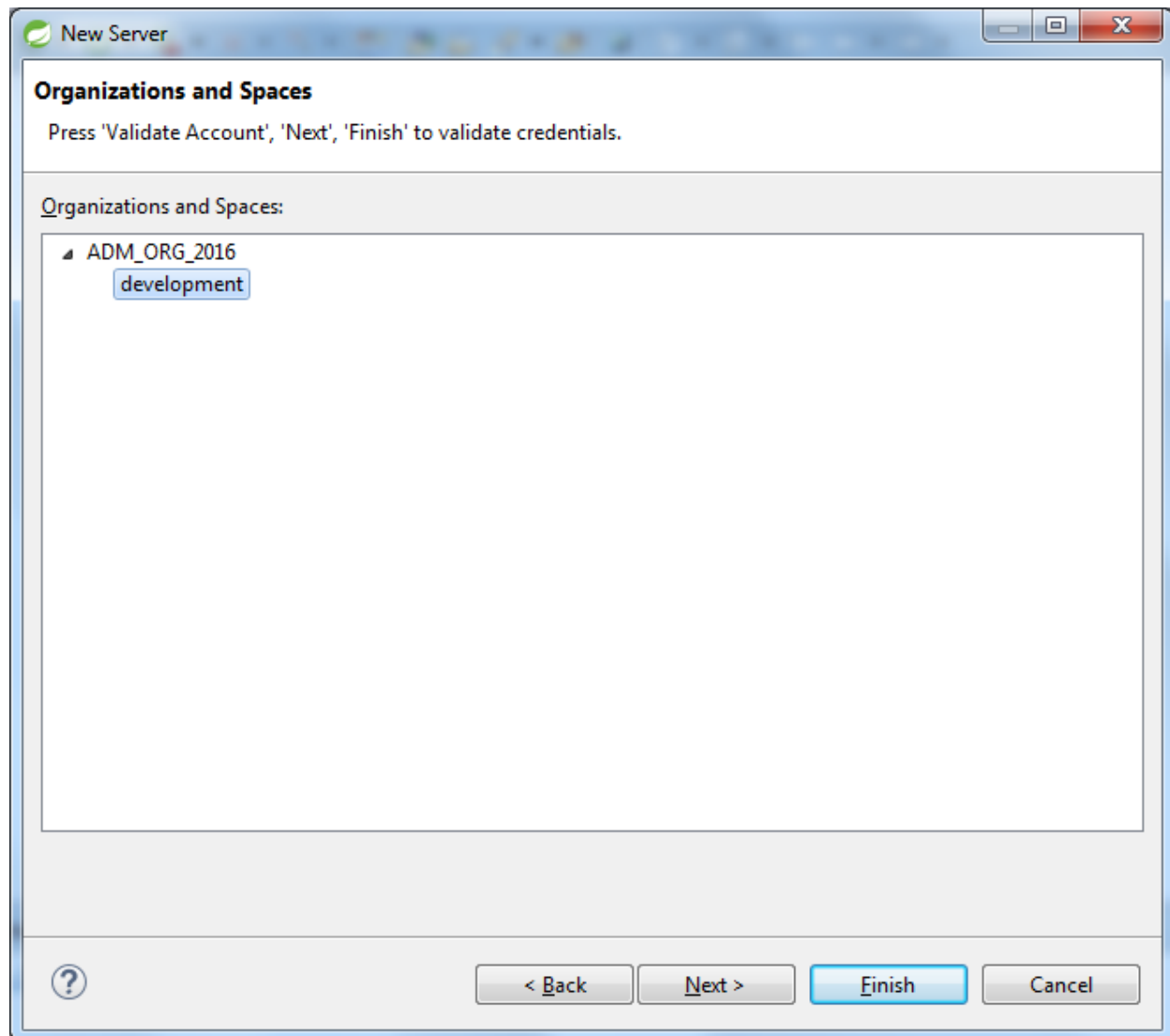
Email:

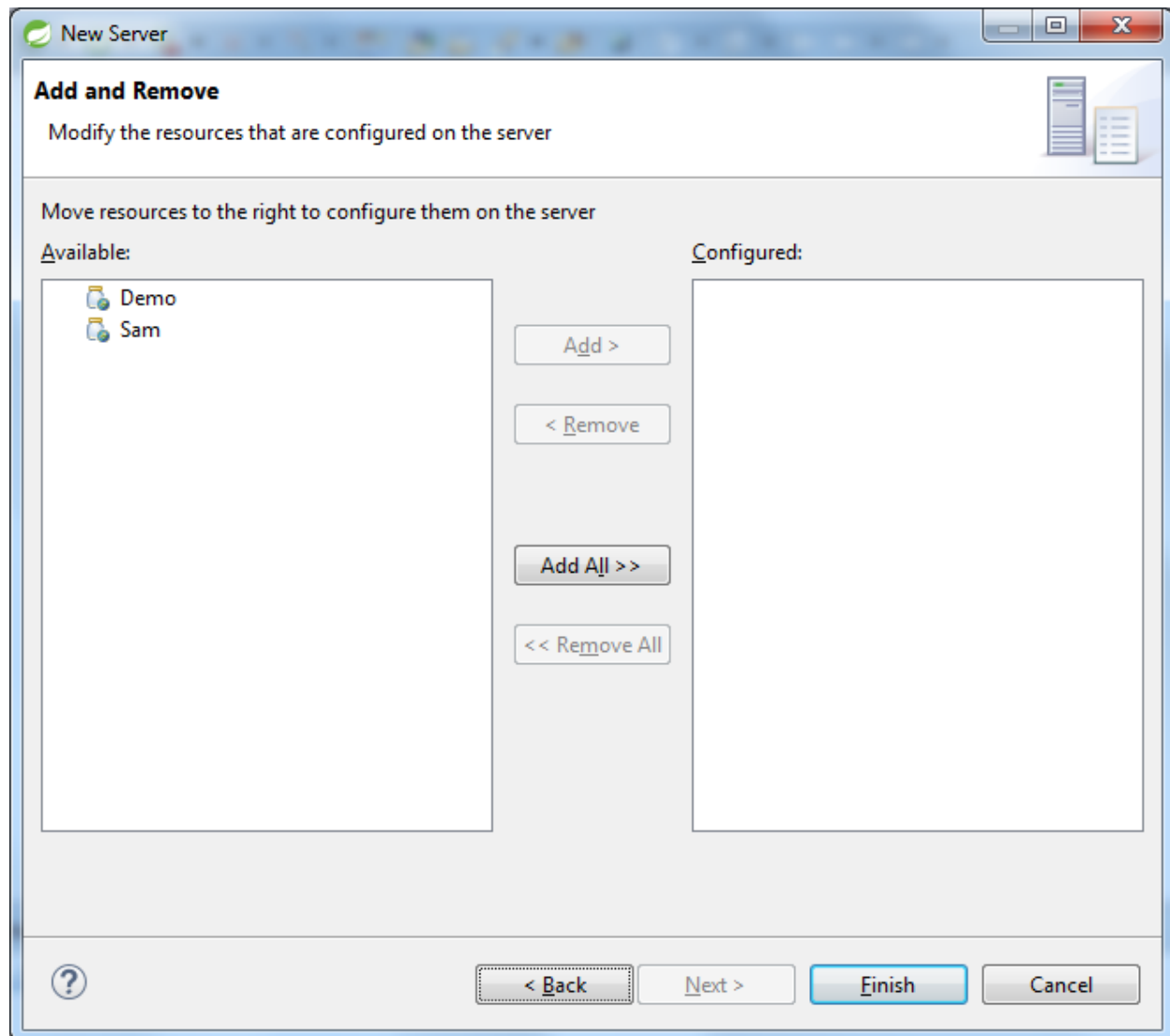
Password:

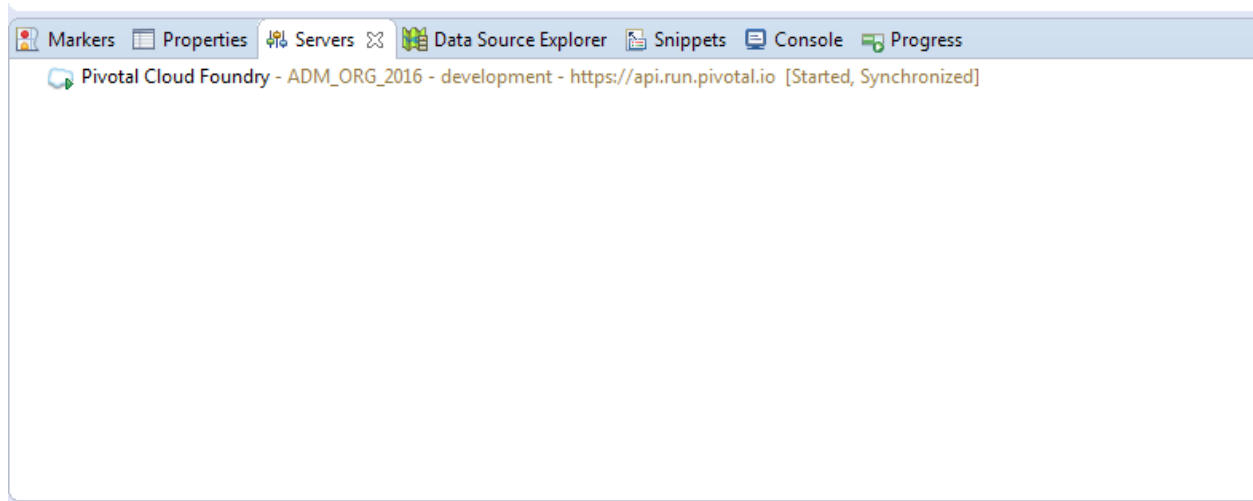
URL:  [Manage Cloud...](#)

[Validate Account](#) [Register Account...](#) [Sign Up](#)

[? Help](#) [< Back](#) [Next >](#) [Finish](#) [Cancel](#)







5. Import projects into your IDE
  - a. Eclipse or STS
    - i. From the project explorer, right click, import, Maven, existing maven projects
    - ii. Navigate to the folder and Finish.
    - iii. The projects will be imported.
    - iv. Note that depending on the version of Eclipse / STS, your version of Java, and other factors, you may encounter errors. These will need to be fixed on a case-by-case basis before proceeding.

**[Congratulations, you have completed this exercise!](#)**

## **Console Log for Lab 2:**

```
Checking application - cf-spring-mvc-demo
Generating application archive - cf-spring-mvc-demo
Pushing application - cf-spring-mvc-demo
Creating application - cf-spring-mvc-demo
Application successfully pushed
Starting application - cf-spring-mvc-demo
Updated app with guid 5051caf0-bd03-4bc4-8602-c8b0f2d8efda ({"state"=>"STARTED"})
Downloading java_buildpack...
Downloaded java_buildpack
Creating container
Successfully created container
```

```

Downloading app package...
Downloaded app package (23.6M)
Staging...
-----> Java Buildpack Version: v3.8.1 (offline) |
https://github.com/cloudfoundry/java-buildpack.git#29c79f2
    Expanding Open Jdk JRE to .java-buildpack/open_jdk_jre (1.0s)
-----> Downloading Open JDK Like Memory Calculator 2.0.2_RELEASE from https://java-
buildpack.cloudfoundry.org/memory-calculator/trusty/x86_64/memory-calculator-
2.0.2_RELEASE.tar.gz (found in cache)
    Memory Settings: -Xmx317161K -Xms317161K -XX:MetaspaceSize=64M -
XX:MaxMetaspaceSize=64M -Xss228K
-----> Downloading Container Customizer 1.0.0_RELEASE from https://java-
buildpack.cloudfoundry.org/container-customizer/container-customizer-
1.0.0_RELEASE.jar (found in cache)
-----> Downloading Spring Auto Reconfiguration 1.10.0_RELEASE from https://java-
buildpack.cloudfoundry.org/auto-reconfiguration/auto-reconfiguration-
1.10.0_RELEASE.jar (found in cache)
-----> Downloading Tomcat Instance 8.0.36 from https://java-
buildpack.cloudfoundry.org/tomcat/tomcat-8.0.36.tar.gz (found in cache)
    Expanding Tomcat Instance to .java-buildpack/tomcat (0.1s)
-----> Downloading Tomcat Lifecycle Support 2.5.0_RELEASE from https://java-
buildpack.cloudfoundry.org/tomcat-lifecycle-support/tomcat-lifecycle-support-
2.5.0_RELEASE.jar (found in cache)
-----> Downloading Tomcat Logging Support 2.5.0_RELEASE from https://java-
buildpack.cloudfoundry.org/tomcat-logging-support/tomcat-logging-support-
2.5.0_RELEASE.jar (found in cache)
-----> Downloading Tomcat Access Logging Support 2.5.0_RELEASE from https://java-
buildpack.cloudfoundry.org/tomcat-access-logging-support/tomcat-access-logging-
support-2.5.0_RELEASE.jar (found in cache)
[Application Running Check] - Checking if application is running - cf-spring-mvc-
demo. Please wait...
Staging complete
Exit status 0
Uploading build artifacts cache...
Uploading droplet, build artifacts cache...
Uploading droplet...
Uploaded build artifacts cache (108B)
Uploaded droplet (76.6M)
Uploading complete
Destroying container
Creating container
Successfully created container
Successfully destroyed container
Starting health monitoring of container
[CONTAINER] org.apache.coyote.http11.Http11NioProtocol      INFO    Initializing
ProtocolHandler ["http-nio-8080"]
[CONTAINER] org.apache.catalina.core.StandardEngine        INFO    Starting
Servlet Engine: Apache Tomcat/8.0.36
[CONTAINER] org.apache.catalina.core.StandardService      INFO    Starting
service Catalina

```

```
[CONTAINER] org.apache.catalina.startup.HostConfig      INFO    Deploying web
application directory /home/vcap/app/.java-buildpack/tomcat/webapps/ROOT
[CONTAINER] org.apache.catalina.startup.Catalina        INFO    Initialization
processed in 497 ms
[CONTAINER] org.apache.jasper.servlet.TldScanner       INFO    At least one
JAR was scanned for TLDs yet contained no TLDs. Enable debug logging for this logger
for a complete list of JARs that were scanned but no TLDs were found in them.
Skipping unneeded JARs during scanning can improve startup time and JSP compilation
time.
[CONTAINER] ing.AutoReconfigurationServletContainerInitializer INFO    Initializing
ServletContext with Auto-reconfiguration ApplicationContextInitializers
[CONTAINER] lina.core.ContainerBase.[Catalina].[localhost].[/] INFO    Spring
WebApplicationInitializers detected on classpath:
[com.gopivotal.cf.workshop.WebApplication@15dbcdcb]
2016-08-08 15:36:23.438 INFO 19 --- [ost-startStop-1]
o.c.r.s.CloudProfileApplicationListener : Adding 'cloud' to list of active profiles
/\ / _' _ _ _ ( ) _ _ _ _ \ \ \ \
.
\ / _ ) | | _ | | | | ( _ | _ ) _ )
( ( ) \ _ | ' _ | ' _ | ' _ \ / _ | \ \ \ \
' | _ | . _ | | | | | _ _ , | / / / /
=====|_|=====|_|_/=//_/_/_/
:: Spring Boot ::          (v1.1.7.RELEASE)
2016-08-08 15:36:24.189 INFO 19 --- [ost-startStop-1]
pertySourceApplicationContextInitializer : Adding 'cloud' PropertySource to
ApplicationContext
2016-08-08 15:36:24.236 INFO 19 --- [ost-startStop-1]
nfigurationApplicationContextInitializer : Adding cloud service auto-reconfiguration
Ect.....
```



### Lab 3:

## Logging and Troubleshooting

### Steps:

### CLI:

Try the following command in CLI

1. cf events <<app-name>>
2. cf logs <<app-name>>
3. cf logs <<app-name>> --recent

### Pivotal Web Console:

In pivotal web console you can see the log details under App → logs link as shown below:

```

Overview  Services  Route (1)  Env Variables  Logs  Settings
Buildpack: https://github.com/

Logs

2016-08-10T19:54:08.987+05:30 [CELL/0] [OUT] Starting health monitoring of container
2016-08-10T19:54:09.510+05:30 [CELL/0] [OUT] Container became healthy
2016-08-10T19:55:10.311+05:30 [RTR/5] [OUT] static-demo-cap-studs.cfapps.io - [10/08/2016:14:25:10.309 +0000] "GET / HTTP/1.1" 200 0 532 "-" "Mozilla/5.0 (Windows NT 6.1; Win64; x64) AppleWebKit/537.36 (KHTML, like Gecko) Chrome/51.0.2704.84 Safari/537.36" 10.10.2.195:55093 x_forwarded_for:"1.39.99.41" x_forwarded_proto:"http" vcap_request_id:9556d2cf-dec8-4df3-7185-43d99e6b9cb0 response_time:0.002442855 app_id:a5e9baab-3704-43d9-a667-d2a04f5da59f
2016-08-10T19:55:10.315+05:30 [APP/0] [OUT] 1.39.99.41, 10.10.2.195 - - [10/Aug/2016:14:25:10 +0000] "GET / HTTP/1.1" 200 544
2016-08-10T19:55:10.664+05:30 [RTR/4] [OUT] static-demo-cap-studs.cfapps.io - [10/08/2016:14:25:10.662 +0000] "GET /js/modernizr.js HTTP/1.1" 200 0 4093 "http://static-demo-cap-studs.cfapps.io/" "Mozilla/5.0 (Windows NT 6.1; Win64; x64) AppleWebKit/537.36 (KHTML, like Gecko) Chrome/51.0.2704.84 Safari/537.36" 10.10.2.195:10636 x_forwarded_for:"1.39.99.41" x_forwarded_proto:"http" vcap_request_id:e0b431ee-4677-4294-5c99-f2a617c4330c response_time:0.002675904 app_id:a5e9baab-3704-43d9-a667-d2a04f5da59f
2016-08-10T19:55:10.667+05:30 [RTR/0] [OUT] static-demo-cap-studs.cfapps.io - [10/08/2016:14:25:10.662 +0000] "GET /css/foundation.css HTTP/1.1" 200 0 18041 "http://static-demo-cap-studs.cfapps.io/" "Mozilla/5.0 (Windows NT 6.1; Win64; x64) AppleWebKit/537.36 (KHTML, like Gecko) Chrome/51.0.2704.84 Safari/537.36" 10.10.2.195:57829 x_forwarded_for:"1.39.99.41" x_forwarded_proto:"http" vcap_request_id:29ef00a9-df97-4fa0-5f00-5c583014ebca response_time:0.004970429 app_id:a5e9baab-3704-43d9-a667-d2a04f5da59f
2016-08-10T19:55:10.668+05:30 [APP/0] [OUT] 1.39.99.41, 10.10.2.195 - http://static-demo-cap-studs.cfapps.io/ - [10/Aug/2016:14:25:10 +0000] "GET /css/foundation.css HTTP/1.1" 200 18054
2016-08-10T19:55:10.669+05:30 [APP/0] [OUT] 1.39.99.41, 10.10.2.195 - http://static-demo-cap-studs.cfapps.io/ - [10/Aug/2016:14:25:10 +0000] "GET /js/modernizr.js HTTP/1.1" 200 4105
2016-08-10T19:55:10.978+05:30 [RTR/5] [OUT] static-demo-cap-studs.cfapps.io - [10/08/2016:14:25:10.963 +0000] "GET /js/jquery.js HTTP/1.1" 200 0 73062 "http://static-demo-cap-studs.cfapps.io/" "Mozilla/5.0 (Windows NT 6.1; Win64; x64) AppleWebKit/537.36 (KHTML, like Gecko) Chrome/51.0.2704.84 Safari/537.36" 10.10.2.195:54962 x_forwarded_for:"1.39.99.41" x_forwarded_proto:"http" vcap_request_id:08e06fc3-e017-4dd2-53e2-c593d33a8691 response_time:0.01458421 app_id:a5e9baab-3704-43d9-a667-d2a04f5da59f

```

### IDE (STS /Eclipse) :

IDE will show all the log reports in console view, when we push/re-push the application with pivotal cloud server.

### Conclusion:

In this lab we learnt how to view log reports in CLI and web console.

**Congratulations, you have completed this exercise!**

## Prerequisites:

1. A Cloud Foundry account (either on your company's CF installation or on Pivotal Web Services)
2. The Cloud Foundry CLI installed

## Lab 4:

### Using a Buildpack

#### Steps:

#### Pushing a Static HTML Application

In this section you will push an existing static HTML application to Cloud Foundry. CF does not have a built-in buildpack for static HTML sites, so you will need to specify an external buildpack that utilizes Nginx.

1. Locate the cf-static-demo folder on your file system.
2. Edit the manifest.yml. Give your application a unique host, and specify <https://github.com/cloudfoundry-community/staticfile-buildpack.git> as the buildpack.
3. Use CLI `cf push` to push the application to your Cloud Foundry environment.
4. When the push is complete, use a browser to access your running application. If it is not running, attempt to debug using `cf logs <app>--recent` and other techniques.

#### **Note**

Some administrators may choose to disallow external buildpacks, in which case you cannot specify an alternate selection. If this is the case in your environment, you may simply use Pivotal Web Services for this exercise.

#### Conclusion:

1. Push a simple Java/Spring application to Cloud Foundry
2. Observe and scale a running application.

**[Congratulations, you have completed this exercise!](#)**

## Prerequisites:

1. A Cloud Foundry account (either on your company's CF installation or on Pivotal Web Services)
2. Either the CCloud Foundry CLI or an IDE installed (with Cloud Foundry support)
3. **Most importantly:** GitHub account. Unlike other labs, a github account is mandatory for this lab.

## Lab 5:

### Customizing a Buildpack

Fork the Java buildpack to make a minor change.

### Steps:

Forking the Java Buildpack

In this section you will fork the Java buildpack, make a minor change to the desired Java version, then use your modified buildpack to push an application to Cloud Foundry, and observe the results.

1. Push the spring-mvc-demo application to Cloud Foundry (if needed)
  - a. Use either the CLI or Eclipse/STS, whichever you find most convenient. If you use **Eclipse/STS** it would be best to save a manifest file for the later steps.
  - b. Observe the console output during the push process. Note the version of Java being used. Our goal in this lab will be to specify a different Java version.
2. Fork the Java Buildpack.
  - a. Using a browser, go to <https://github.com/cloudfoundry/java-buildpack> (if you are not signed into GitHub, do so now).
  - b. Click the **"Fork"** button to create a fork of this repository under your own GitHub account. (You may clone a copy of this repository locally, though it is not necessary for the steps here)
3. Alter the Java version
  - a. While still in the browser viewing your forked copy of the Java buildpack, drill-down to **config/open\_jdk\_jre.yml**. Click the edit button to modify this file in place.
  - b. Find the line (near line 20) that indicates the Java version: `version: 1.8.0_+`
  - c. Alter this line to deliberately specify an older Java version: `version: 1.8.0_31`
  - d. Save your work using the 'Commit' button.
4. Re-push the application using the altered Buildpack
  - a. This step is performed differently depending on whether you have an existing manifest file or not. Assuming you do:

- i. Alter the manifest file. Add a line to indicate you new buildpack (change “yourGitId” and “java-buildpack” as needed to match). Be sure your indentation is correct:  
buildpack: `https://github.com/[yourGitId]/[java-buildpack]`
- ii. Re-push the application. If using Eclipse ensure that it is picking up the values you specified in the manifest.
- b. If you did not have a manifest file, and you used the CLI, repush using the -b option to specify the buildpack (alter “yourGitId” and “java-buildpack” as needed):  
`-b https://github.com/[yourGitId]/[java-buildpack]`
- c. Observe the console output during the push process. Note the version of Java being used. It should be the version you specified above.

### Conclusion:

We learnt the simple Customize buildpack creation.

**Congratulations, you have completed this exercise!**

## Prerequisites:

1. A Cloud Foundry account (either on your company's CF installation or on Pivotal Web Services)
2. Either the Cloud Foundry CLI or an IDE installed (with Cloud Foundry support)

## Lab 6:

### Integrating with 3rd Party Log Management Tools

#### Setup Paper Trail

1. Go to <https://papertrailapp.com> and establish a free trial account.
2. Once logged in, use "Add a Project" to create a new project. Give the project any name you like, such as "**PWSLogDrain**". Follow the instructions at <http://docs.cloudfoundry.org/devguide/services/log-management-thirdparty-svc.html#papertrail> to setup your project with the correct IP addresses.

#### Create and Bind User-Provided Service

1. Create a user-provided service using the command `cf cups log-drain -l syslog://<URL-FROM-PAPERTRAIL>`, replacing `<URL-FROM-PAPERTRAIL>` with the value provided by Papertrail. You may name the service whatever you like.
2. Bind the service to any of your existing applications used in prior labs.
3. Restart the application.
4. Access the URL of your application and generate logging activity. This varies depending on the application you are using; click on available links, refresh the web page, etc.

#### View Log Output

1. Return to Papertrail. Use the "Dashboard" button to find the main log output. Experiment with the search feature at the bottom of the page, look for terms such as "GET" or "hibernate".

**Congratulations, you have completed this exercise!**

## Prerequisites:

1. A Cloud Foundry account (either on your company's CF installation or on Pivotal Web Services)
2. Either the Cloud Foundry CLI or an IDE installed (with Cloud Foundry support)

## Lab 7:

### Integrating with APM tools

Integrate multiple instances of a running application with a third-party Application Performance Monitoring tool.

## Steps:

### Configure New Relic APM Service

1. From the PWS Apps Manager, go to the marketplace and provision a new New Relic service instance. Name the instance anything you like (suggestion: "new-relic").
2. Determine if you already have the spring-music app deployed on Cloud Foundry from an earlier lab. If not, push it now. Bind it to the "new-relic" service you just provisioned. Restage the application. Scale it to 2 instances.
3. Once the application starts, open its URL in a browser. Refresh it several times. Notice that the instance index, id, and port changes depending on which of the 4 instances you are hitting.
4. Return to the PWS Apps Manager. From the list of services in your application's space, locate the new-relic service and click on the "Manage" link.

## Note

*New Relic will use OAuth for authentication, so you will be prompted with a login screen from Pivotal Web Services. Essentially, it is verifying with PWS the account information.*

5. Agree to the New Relic terms of service, or temporarily defer.
6. You should now see a page that lists your Cloud Foundry applications (only one should be in the list at the moment, the spring-music). Click on this entry to obtain detailed monitoring information. (Note that it may take several moments for detailed logging information to appear.)
7. If you have reached this point, congratulations! You have successfully bound your application to an APM via a Cloud Foundry service. If you have time, investigate the APM's capabilities. (Note that the specific capabilities of the New Relic APM are outside the scope of this course.)

**Congratulations, you have completed this exercise!**

## Prerequisites:

1. A Cloud Foundry account (either on your company's CF installation or on Pivotal Web Services)
2. Either the Cloud Foundry CLI or an IDE installed (with Cloud Foundry support)

## Lab 8:

### Session Management

In this lab you will learn how to enable an HTTP Session to survive instance failure

#### Steps:

You will be given an application that uses the HttpSession which will lose information if restarted. Then you will fix it in Cloud Foundry by adding a Redis session replication service.

#### Push

Locate the cf-session-management application and push it to Cloud Foundry. You may use CLI or Eclipse.

1. If using the CLI, edit the manifest.yml and specify a unique host (add your name or initials).
2. Once its running, go to its URL in the browser to see it working. The application is very simple. Enter a name and description (anything you like) and click "Submit". The name and description are displayed in the results page.
3. Refresh the page several times. Notice that the values are retained. You can examine the code if you are curious how the session is used.

#### Restart Application

1. Click on the "Stop!" link to kill the application (the code behind this actually kills the JVM, simulating a real application crash).
2. Go back to the previous page in the browser and refresh that page. You will get a CF error: 404 Not Found: Requested route ('cf-session-demo2.cfapps.io') does not exist.
3. Wait a moment for Cloud Foundry's Health Manager to notice the missing instance and cause it to be restarted. You can monitor the restart via the CLI, Eclipse plugin, or Apps Manager if you like.
4. Refresh the page again, note that the previously submitted data has been lost. This is expected since the session data is stored in memory; it is lost when the application instance stops. In the next steps we will use a service to cause the session data to be persisted external to the instance.

## Bind the Redis session-replication Service

1. Using either the CLI, Apps Manager, or Eclipse, provision a new Redis service instance session-replication. You must use this name for the service instance to obtain the desired session behavior. If you are running on PWS the service is called rediscloud. Select the 25Mb plan (its free!).
2. Bind the service to the application. You can do this via CLI, Apps Manager, or Eclipse.
3. Push the application to Cloud Foundry again. Though the application code did not change, we need to force the application to be restaged. (If using Eclipse you can use the "Update and Restart" button as a shortcut).

## Note

You must run cf push again or the new service will not be used.

## Retry Killing the Application

1. Return to your browser and check the application is working again. Go to the Home page and re-enter some name and description data.
2. Redo all the steps in Section 14.2.2, "Restart Application". However, this time the name and description should be retained by the Redis-backed session even after the application instance has been stopped.

## Examine the Application

1. If you have time, take a few minutes to examine the application code. This is a Spring Boot application using Spring MVC Controllers. You might like to look at the code to see how it works. In particular: FormController and DisplayController. The FormController loads values into the session and the DisplayController pulls them out. The Spring Boot initialization class is Config, Application defines main() and WebApplication allows this application to also run as a WAR.

**Congratulations, you have completed this exercise!**



## Prerequisites:

1. A Cloud Foundry account (either on your company's CF installation or on Pivotal Web Services)
2. Either the Cloud Foundry CLI or an IDE installed (with Cloud Foundry support)

## Lab 9:

### Blue/Green Deployment

Will learn: Practice performing a zero-downtime deployment

## Steps:

### Push "Blue"

Push the `cf_bluegreen` application to Cloud Foundry using the CLI (do not use an IDE for this first step).

1. Give your app the name "blue" for the application's name. 512M should be sufficient.
2. Give your app a host name of "XXX-always-available", except replace "XXX" with some value to ensure a unique route. Remember this host for later steps.
3. For the path, specify the "pre-built/blue.war". (This WAR file has been built for you using Maven and the project's source files.)
4. After the application starts, open a browser and access the app: <http://XXX-always-available.cfapps.io> (except using the URL based on your host name). Notice the large blue box on the screen. Leave this browser window open.

## Pivotal CF

### Welcome to Pivotal CF

Blue Deployment

The application port is 64284

#### Alter the application

1. Using your IDE, open the `src/main/resources/templates/index.html` page.
2. Find the tag containing "Blue Deployment". Comment this tag out using XML comment `<!-- -->` tags.
3. Notice the tag containing "Green Deployment" that is presently commented out. Remove the comments. Save your work.
4. If using Eclipse, the application will be built for you automatically and you are ready for the next step.

#### Push "Green"

Push a separate copy of the `cf_bluegreen` application to Cloud Foundry

1. For this step, we recommend you use an IDE to avoid the need to build the Java application. If you prefer to use the CLI, you will need to build the application using Maven, which you may not have installed ("mvn clean install" is the command). If for any reason you cannot build the application, an existing WAR "green.war" is available for you in the pre-built sub folder.

2. Push the app using “green” for the application’s name. 512M should be sufficient. As always, make sure your deployed URL is unique by customizing the host.
3. After the application starts, open a separate browser window/tab and access the green app via its route. Notice the large green box on the screen.

### Alter Routes

1. Return to the original browser window/tab. Refresh it multiple times, notice it is still pointing to the old ‘blue’ deployment.
2. Using either the CLI, IDE, or Apps Manager, map the same “XXX-always-available” to the “green” app. IF using the CLI you may receive a warning that the route is already in use; this is expected.
3. Return to the original browser window/tab. Refresh this page several times. Notice that the same route alternates randomly between the ‘blue’ and ‘green’ apps; this is expected since the route is temporarily mapped to both applications.
4. Alter the blue app by removing its route.
5. Refresh the browser. The browser is now pointing to the new green instance with no downtime.

## Pivotal CF

### Welcome to Pivotal CF

Green Deployment

The application port is 61318

### Conclusion:

The conclusion: no downtime would be experienced by any user of the cf\_bluegreen application when accessed via its "always-reliable" route. Congratulations! You have conducted a zero-downtime deployment! You can stop and/or delete both applications now.

**Congratulations, you have completed this exercise!**

## Treasure Hunt

### Questions:

1. What languages / frameworks does Cloud Foundry appear to support (see “buildpacks”)?
2. What kinds of things can be specified in a “Manifest”?
3. When designing applications for the cloud, what are some practices to avoid? (Hint: see section on "App design for the cloud")
4. What are the names of some of the architectural components within Cloud Foundry?
5. What are some of the “log types” that you can expect to find in Cloud Foundry logs?  
Answer the following questions based on <http://docs.pivotal.io/pivotalcf/> - note that many of these concepts have not yet been discussed in class:
6. What kinds of things can be defined in the “Apps Manager”?
7. What is the purpose of a “Buildpack”?  
Answer the following questions based on <http://docs.run.pivotal.io> - note that many of these concepts have not yet been discussed in class:
8. What kind of services are presently available in the services marketplace?
9. BONUS When building a custom buildpack, what is the purpose of the “DETECT” script?
10. BONUS: If you were to “push” a new web application named “weeble” to Pivotal web services, what URL could you use to access it?

### Answers

Here are the answers to the questions in the Treasure Hunt (documentation) lab:

11. Java, Node.js, Ruby, Go, PHP, Python.
12. Application name, buildpack, command, domain, instances, memory, host, timeout, anything described in: <http://docs.cloudfoundry.org/devguide/deploy-apps/manifest.html>
13. Avoid local file system, avoid Http session, avoid non-standard ports, avoid pushing unneeded files.
14. Router, Cloud Controller, DEA, health manager, ... See: <http://docs.cloudfoundry.org/concepts/architecture>
15. API, STG, DEA, RTR, LGR, APP, all defined here: <http://docs.cloudfoundry.org/devguide/deploy-apps/streaming-logs.html>
16. Orgs, spaces, users, ...
17. Allows you to extend CF to handle other languages and/or frameworks.
18. See: <http://docs.run.pivotal.io/marketplace/services> - cleardb, cloudamqp elephantsql, mongolab, newrelic sendgrid, etc.
19. Detects whether or not a buildpack can be applied to a given application.
20. [weble.cfapps.io](http://weble.cfapps.io)