

Pivotal



Pivotal  
Cloud Foundry®



# Spring Boot to PCF Developer Workshop

---

Pas Apicella = {

Company: Pivotal,

Advisory Platform Architect: <http://pivotal.io>,

Email: [papicella@pivotal.io](mailto:papicella@pivotal.io)

Twitter: @PasApicella,

Blog: <http://theblasfrompas.blogspot.com.au/>

}

High Level View

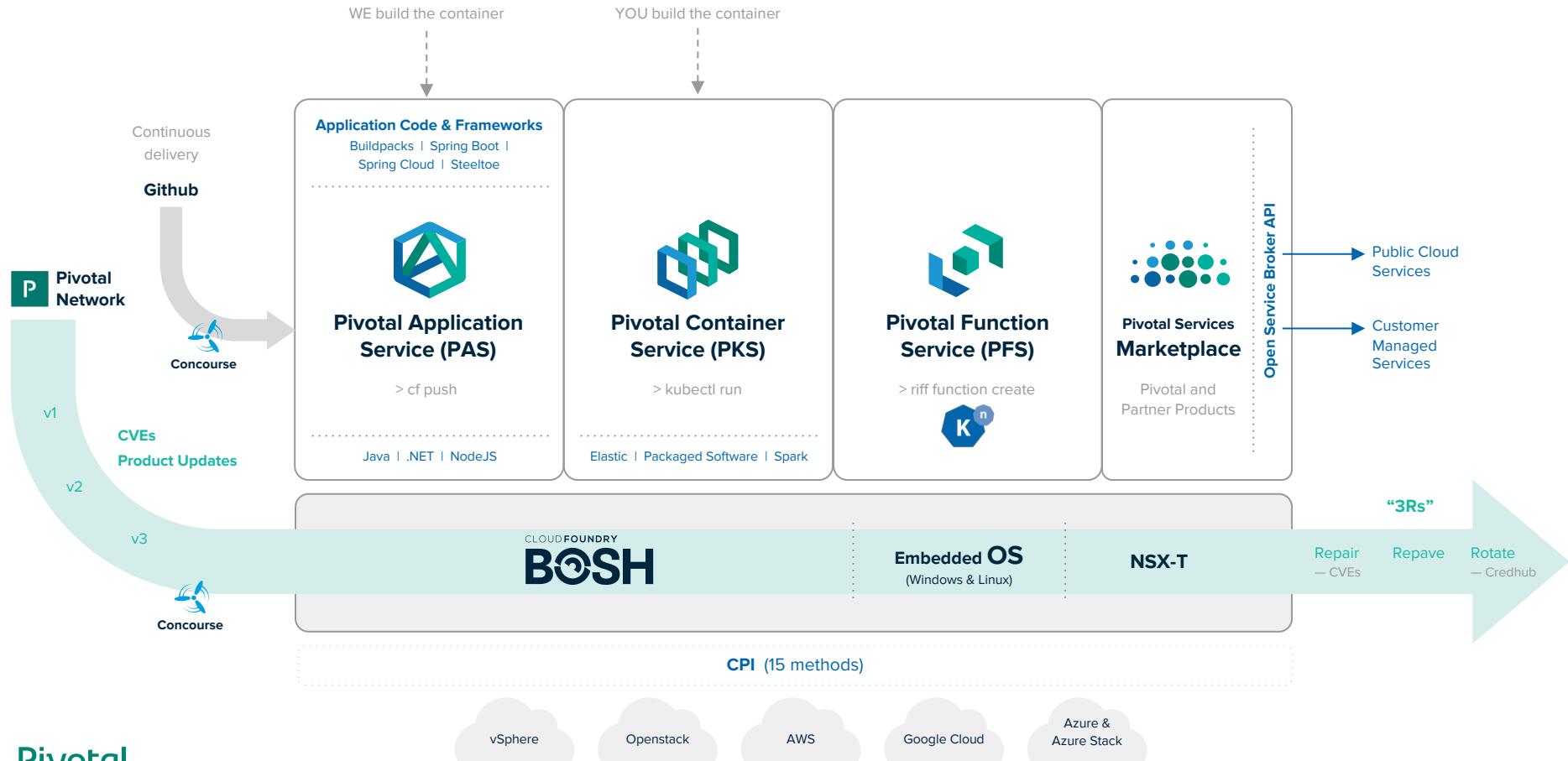
---

# What is PCF?



Pivotal  
Cloud Foundry®

# Any Workload Anywhere Anytime



# Pivotal Application Service (PAS): A Runtime for Apps

Increase speed and deploy code to production thousands of times per month. Use PAS to run Java, .NET, and Node apps.



Pivotal  
**Application Service**<sup>TM</sup>

Pivotal

**Best runtime for Spring and Spring Boot** — Spring's microservice patterns—and Spring Boot's executable jars—are ready-made for PAS.

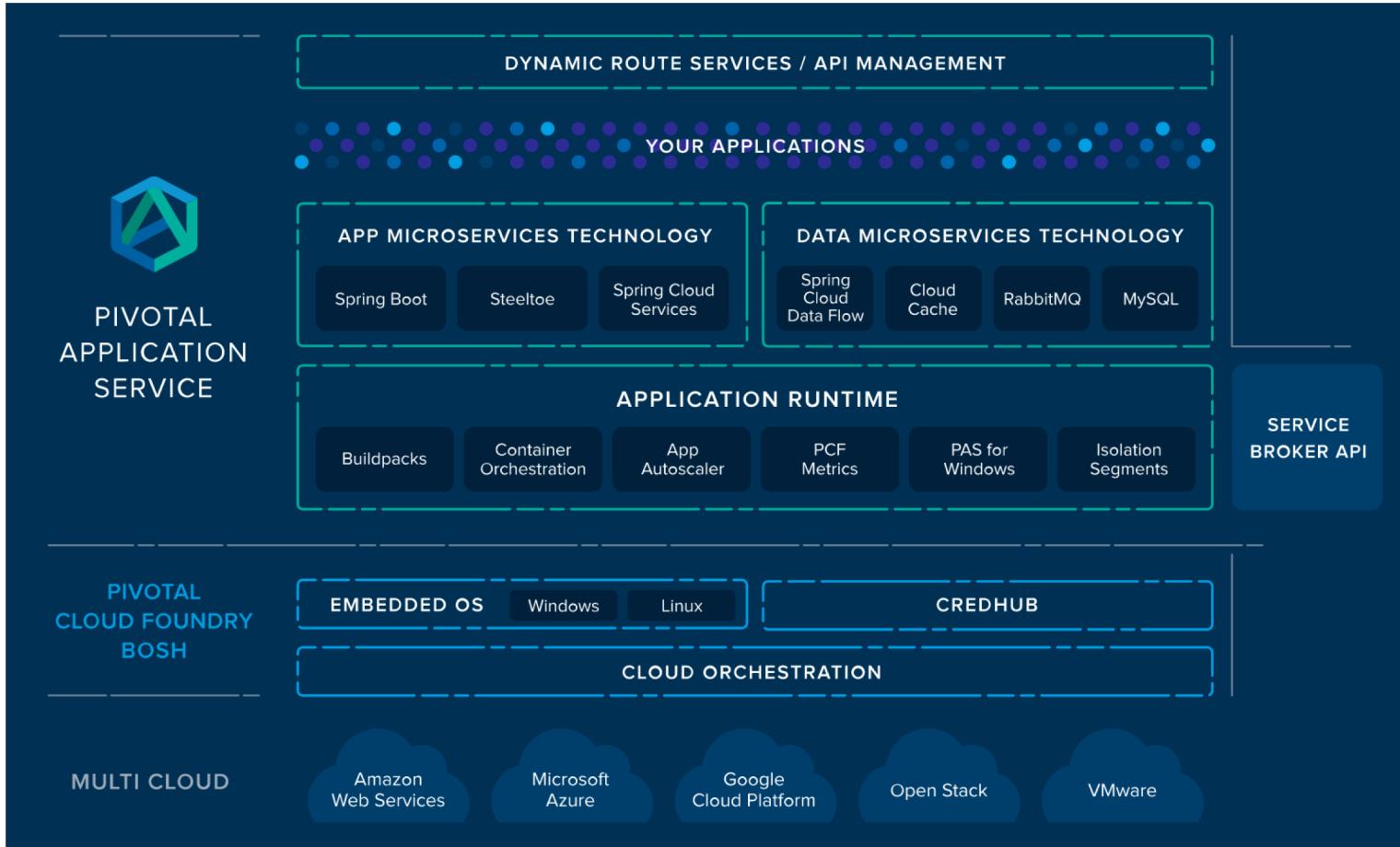
**Turnkey microservices operations and security** — Spring Cloud Services brings microservices best practices to PAS. It includes Config Server, Service Registry, and Circuit Breaker Dashboard.

**A native Windows and .NET experience** — Use PAS to run new apps built with .NET Core. Run your legacy .NET Framework apps on PAS too, using the .NET Hosted Web Core buildpack. Push applications to containers running on Windows Server 2016.

**Built for apps** — PAS has everything to need to run apps. Buildpacks manage runtime dependencies; metrics, logging, and scaling are done for you. Multitenancy, and blue/green deployment patterns are built-in. Extend apps with a rich service catalog.

**Container-ready** — PAS supports the OCI format for Docker images. Run platform-built and developer-built containers.

# Spring Boot To PCF Developer Workshop



Spring Boot to PCF Developer Workshop

---

# What Is Needed

# Spring Boot To PCF Developer Workshop

- An account with Pivotal Web Services (<https://run.pivotal.io/>)
- At least 2G of memory and 1 service spare capacity on your PWS account
- CF CLI installed (<https://github.com/cloudfoundry/cli>)
- Maven CLI installed (<https://maven.apache.org/download.cgi>), for this workshop it was tested with “Apache Maven 3.3.9”
- Java 1.8 installed, this workshop was tested with “1.8.0\_91”.
- IntelliJ IDEA , Eclipse or Spring Source Tool Suite or similar Java IDE
- CURL or HTTPie or a browser that can display RESTful JSON data through a plugin

Spring Boot to PCF Developer Workshop

---

# What Is Covered

# Spring Boot To PCF Developer Workshop

- Create a Data Bound Spring Boot Application Micro Service returning RESTful endpoints
- Tech Stack:
  - Hibernate JPA
  - Spring Boot
  - H2 In Memory Database as well as MySQL Database
  - Spring Data Repositories
  - Spring Boot Actuator
- Write some Java Code within an IDE (This is not a Spring Boot or Spring Course)
- Deploy to Pivotal Cloud Foundry
- Switch to a MySQL database in Pivotal Cloud Foundry without Code Changes
- All things application from
  - View application in “Application Manager”
  - Scale our application
  - View Logs
  - Monitor our application
  - Setup an Auto Scaling Policy
  - And more .....

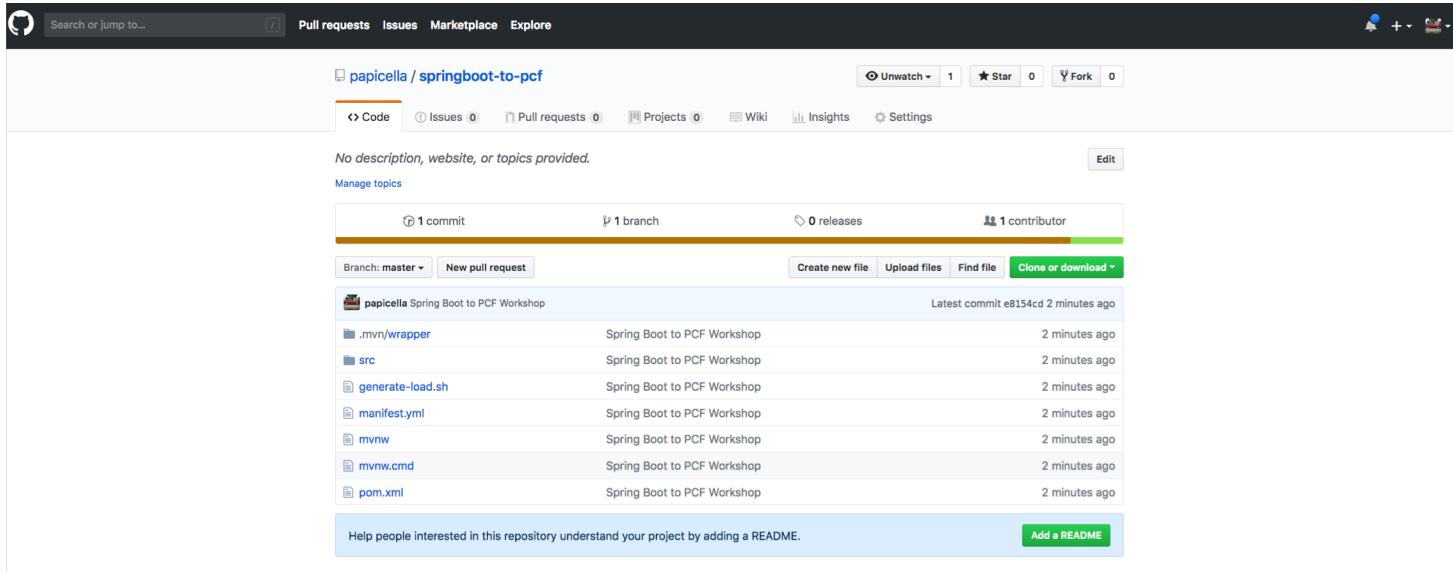
Spring Boot to PCF Developer Workshop

---

# Let's Get Started

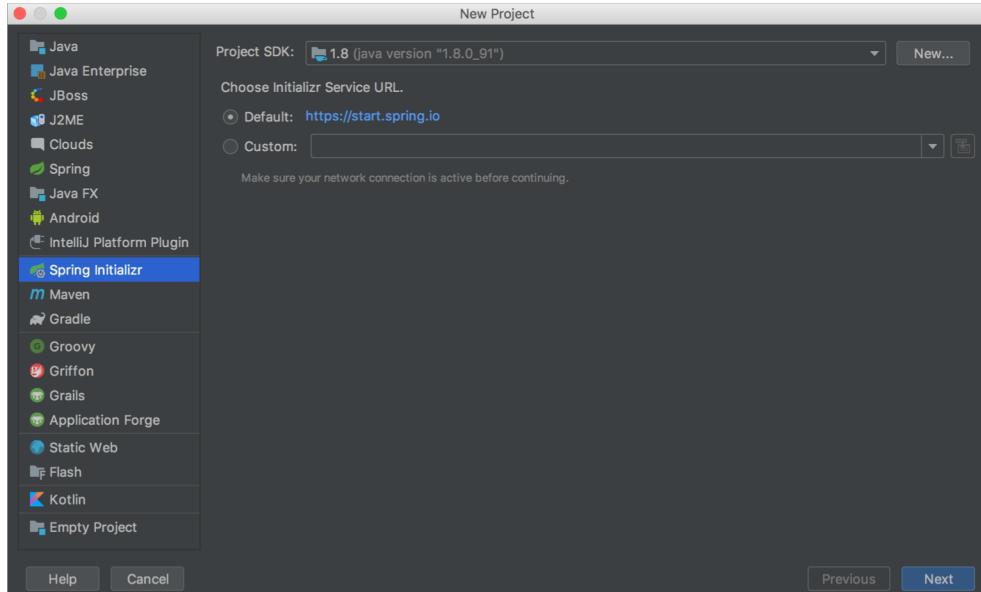
# Spring Boot To PCF Developer Workshop

- The whole source code is on GitHub. If you decide to just copy and paste code text then use GitHub itself rather than copying/pasting from PPTX
  - <https://github.com/papicella/springboot-to-pcf>



# Spring Boot To PCF Developer Workshop

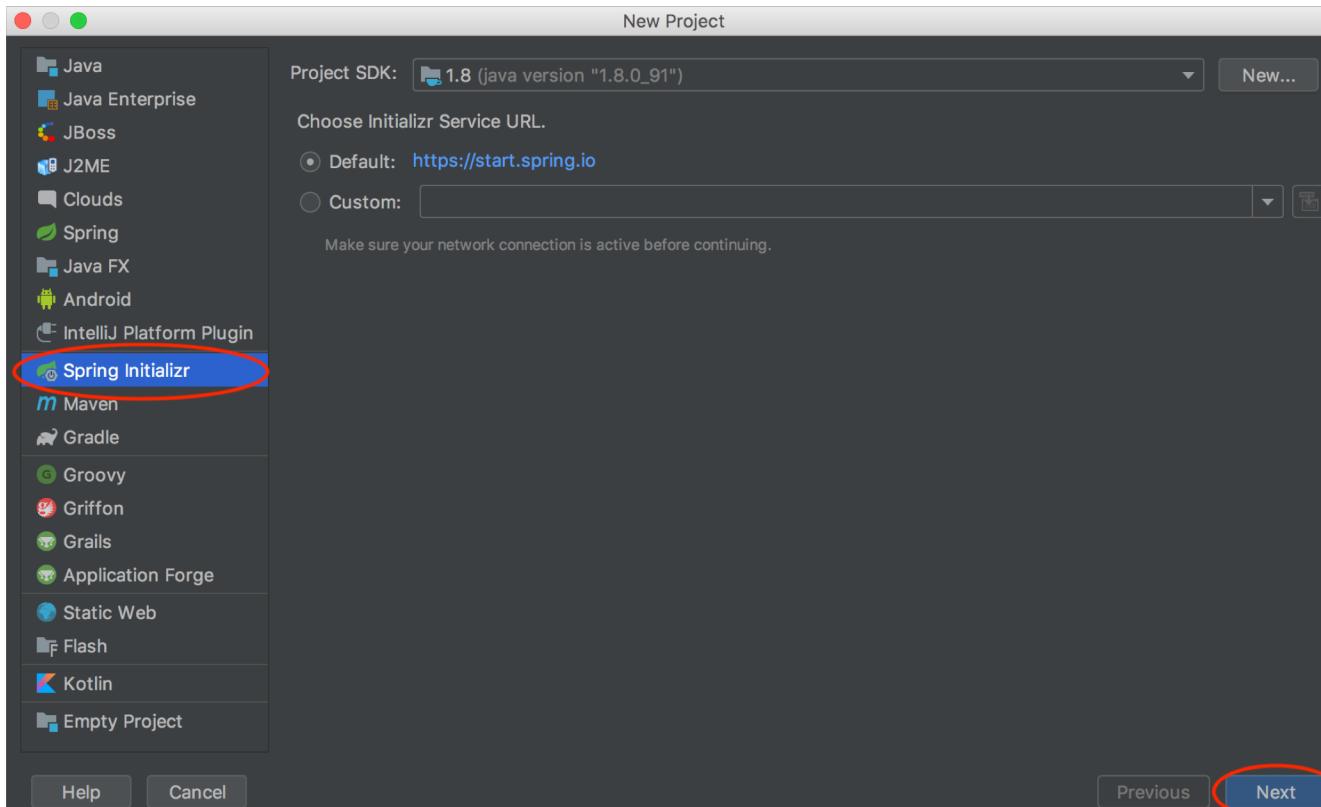
- Let's create a spring boot application one of two ways
  - http://start.spring.io (Web Browser) as shown to the right
  - Within your Java IDE (IntelliJ IDEA as shown below or Eclipse, STS)



The screenshot shows the 'SPRING INITIALIZR' web interface. At the top, it says 'bootstrap your application now' and 'Generate a Maven Project with Java and Spring Boot 2.1.0'. The 'Project Metadata' section includes 'Artifact coordinates' (Group: com.example, Artifact: demo) and 'Dependencies' (Web, Security, JPA, Actuator, Devtools...). A 'Generate Project' button is at the bottom. A large green hexagonal icon with a white power symbol is centered at the bottom of the page.

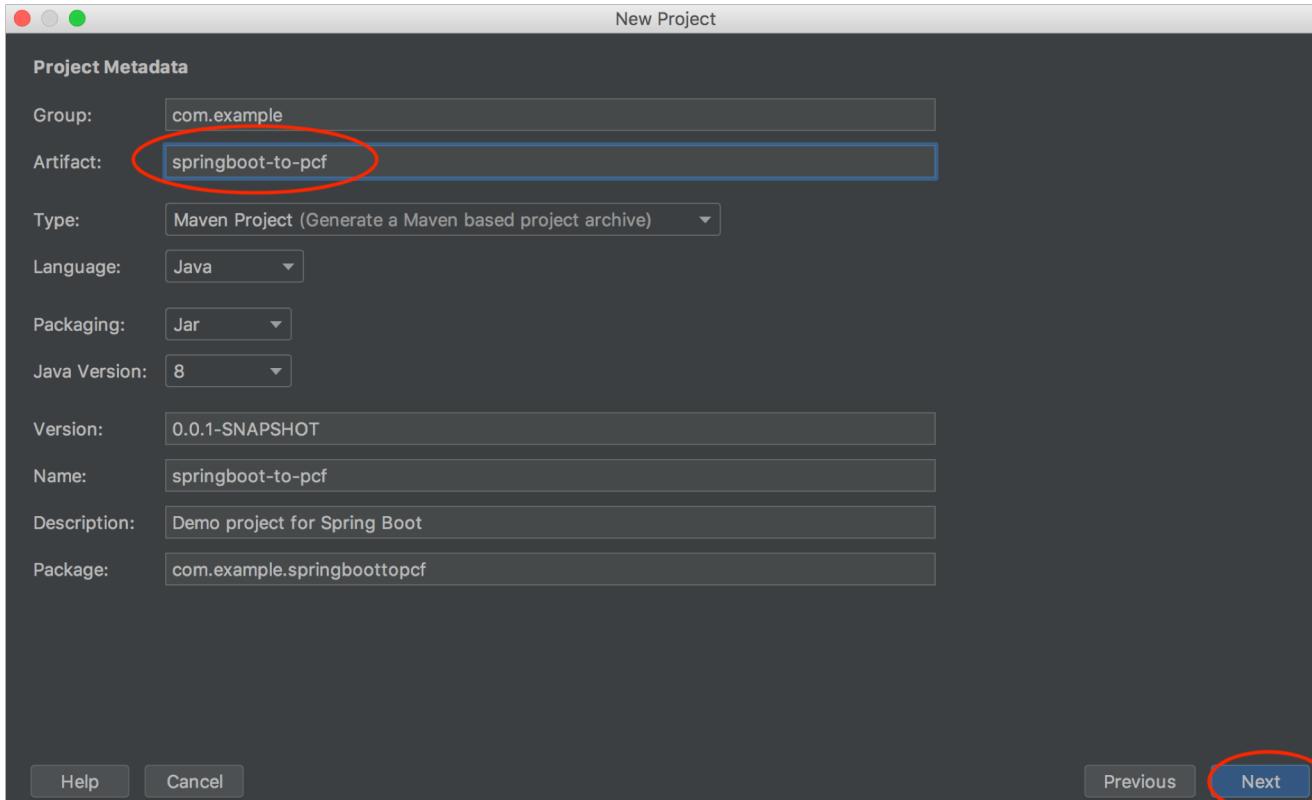
# Spring Boot To PCF Developer Workshop

- Using IntelliJ IDEA, Select “Spring Initializer” and click “Next”



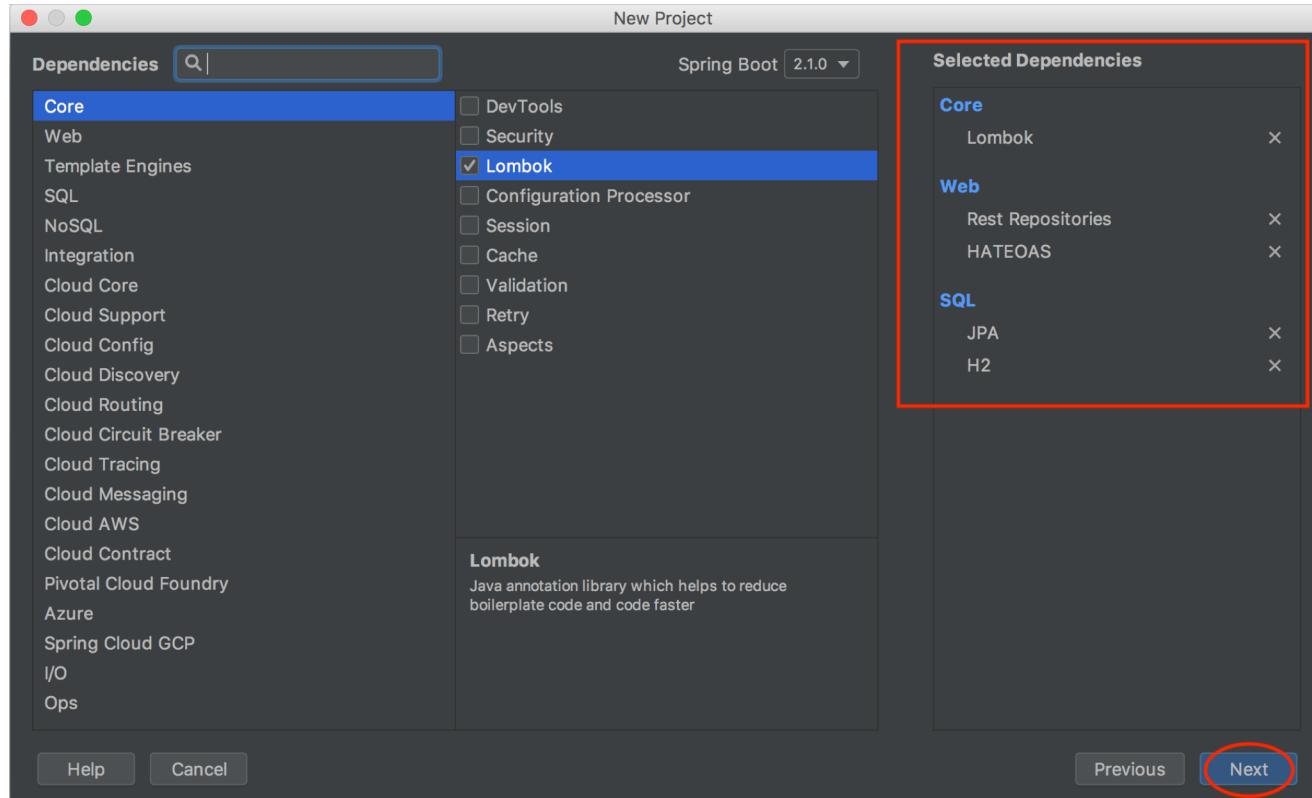
# Spring Boot To PCF Developer Workshop

- Set the “Artifact” to “springboot-to-pcf” and click “Next”



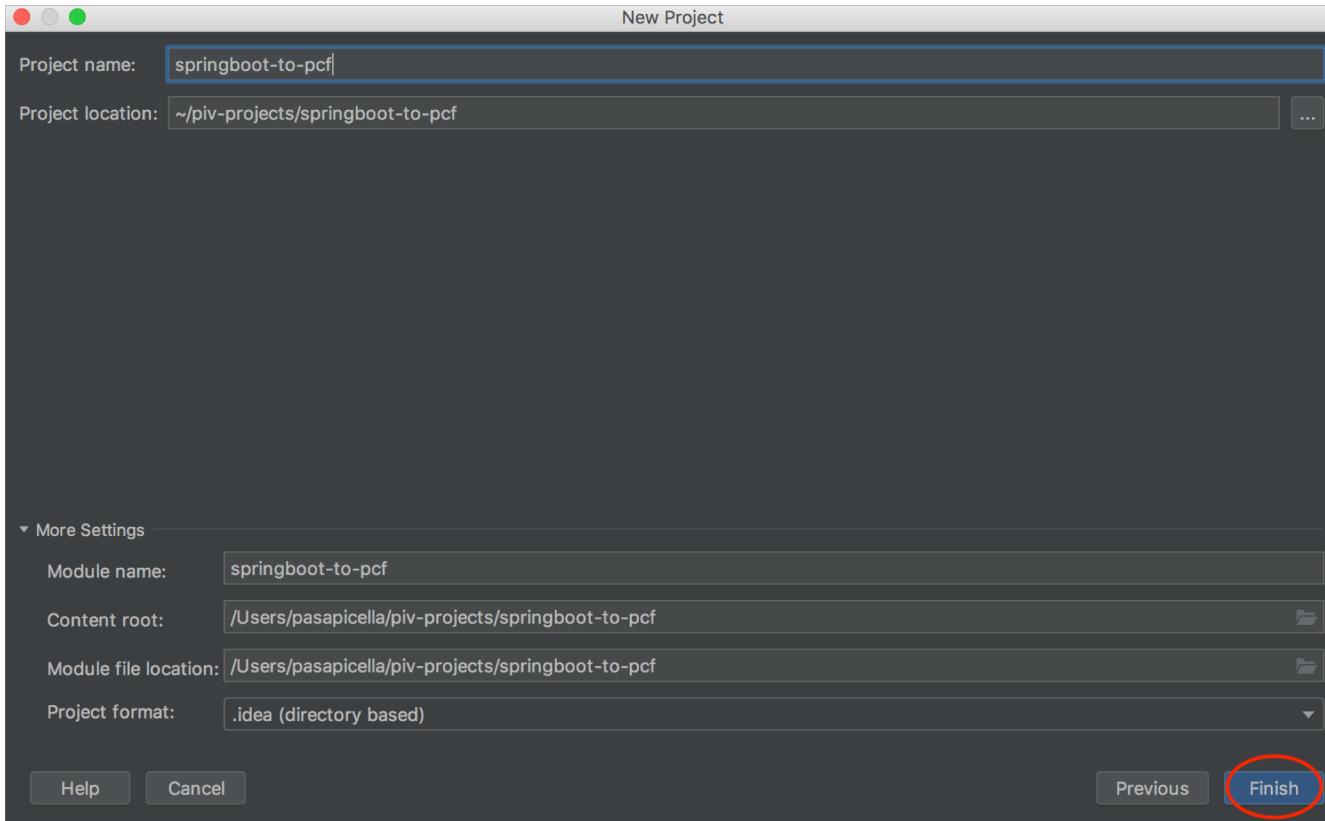
# Spring Boot To PCF Developer Workshop

- Select the following dependencies and click “Next”
  - H2
  - JPA
  - Rest Repositories
  - Lombok



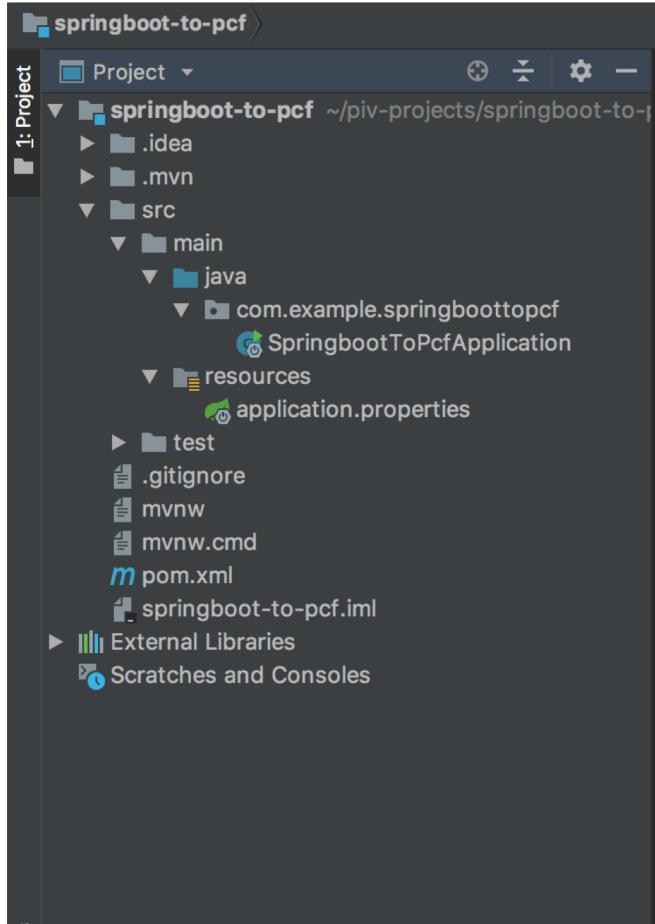
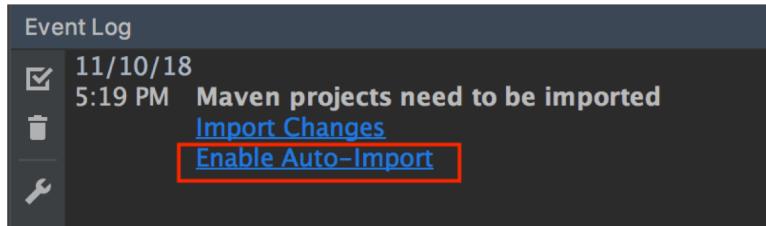
# Spring Boot To PCF Developer Workshop

- Click “Finish” and wait for Maven to download the required JAR files



# Spring Boot To PCF Developer Workshop

- Make sure you enable auto-import for maven as we will make changes at some point to our pom.xml file
- To the right is our Spring Boot project ready for us to write code



# Spring Boot To PCF Developer Workshop

- Open the class file “SpringbootToPcfApplication.java”
- Note the use of the @SpringBootApplication annotation that makes this application a Spring Boot Application and auto configuration will then be applied to sensible defaults such as:
- H2: Spring boot will scan the class path and given H2 JAR files exist will auto configure a Data Source for us out of the box. Settings can be overridden using the “application.properties” or “application.yml” file

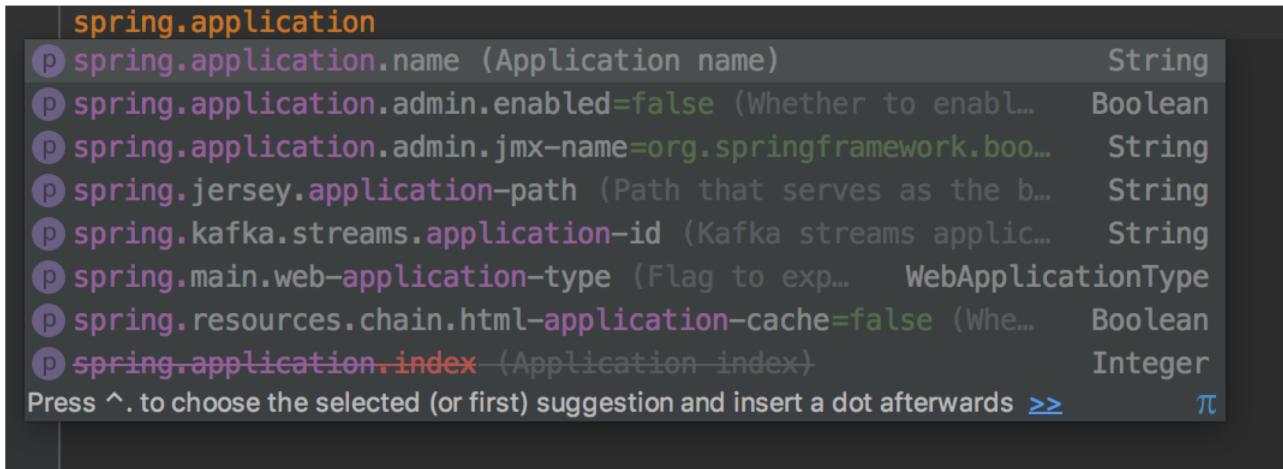
@SpringBootApplication

```
public class SpringbootToPcfApplication {  
    public static void main(String[] args) {  
        SpringApplication.run(SpringbootToPcfApplication.class, args);  
    }  
}
```

# Spring Boot To PCF Developer Workshop

- Lets actually set the Spring Boot application name by editing “application.properties” and adding a name as follows. You can use auto insight by typing just a few characters if you wish as shown below which is great way to see what other properties can be set

```
spring.application.name=My First Spring Boot Application
```



A screenshot of an IDE showing code completion for the variable 'spring.application'. A tooltip window lists several properties starting with 'spring.application.':

Property	Type
spring.application.name (Application name)	String
spring.application.admin.enabled=false (Whether to enable...)	Boolean
spring.application.admin.jmx-name=org.springframework.boot... (JMX name for the application)	String
spring.jersey.application-path (Path that serves as the base for Jersey resources)	String
spring.kafka.streams.application-id (Kafka streams application ID)	String
spring.main.web-application-type (Flag to expose the application as a web application)	WebApplicationType
spring.resources.chain.html-application-cache=false (Whether to cache static resources)	Boolean
spring.application.index (Application index)	Integer

Below the list, a message says: "Press ^ to choose the selected (or first) suggestion and insert a dot afterwards >>".

# Spring Boot To PCF Developer Workshop

- Lets add an JPA Entity to our project with a class named “Employee”.

```
import javax.persistence.*;
import lombok.*;

@Data
@NoArgsConstructor
@AllArgsConstructor
@ToString
@Entity
public class Employee {
    @Id @GeneratedValue(strategy = GenerationType.AUTO)
    private Long id;
    private String name;
}
```

# Spring Boot To PCF Developer Workshop

- Wondering what JPA provider we are using. Well it's Hibernate and Spring Boot starters are what have brought in the hibernate libraries.

```
<parent>
  <groupId>org.springframework.boot</groupId>
  <artifactId>spring-boot-starter-parent</artifactId>
  <version>2.1.0.RELEASE</version>
  <relativePath/> <!-- lookup parent from repository -->
</parent>

<dependencies>
  <dependency>
    <groupId>org.springframework.boot</groupId>
    <artifactId>spring-boot-starter-data-jpa</artifactId>
  </dependency>
</dependencies>
```

- ▶  Maven: org.hibernate.common:hibernate-commons-annotations:5.0.4.Final
- ▶  Maven: org.hibernate.validator:hibernate-validator:6.0.13.Final
- ▶  Maven: org.hibernate:hibernate-core:5.3.7.Final

# Spring Boot To PCF Developer Workshop

- Hibernate will automatically create tables in the underlying database for us. It's good practice to set your desired behavior in the "application.properties" as follows in any case and here we clearly say DON'T do this. Once again you can use Auto Insight by typing a few characters as shown below

```
spring.application.name=My First Spring Boot Application
```

```
spring.jpa.hibernate.ddl-auto=none
```

```
spring.datasource.initialization-mode=always
```

A screenshot of an IDE showing the configuration file. The first line is 'spring.application.name=My First Spring Boot Application'. Below it, the code 'jpa' is typed, and a dropdown menu shows various properties starting with 'spring.jpa.', such as 'spring.jpa.hibernate.ddl-auto', 'spring.jpa.hibernate.use-new-id-generator-mappings', and 'spring.jpa.generate-ddl'. The 'spring.jpa.hibernate.ddl-auto' entry is highlighted.

```
spring.application.name=My First Spring Boot Application
jpa
  P spring.jpa.hibernate.ddl-auto (DDL mode) String
  P spring.jpa.hibernate.use-new-id-generator-mappings (Whet... Boolean
  P spring.jpa.generate-ddl=false (Whether to initialize the... Boolean
  P spring.data.jpa.repositories.bootstrap-mode=default BootstrapMode
  P spring.data.jpa.repositories.enabled=true (Whether to en... Boolean
  P spring.jpa.database (Target database to operate on, aut... Database
  P spring.jpa.database-platform (Name of the target database... String
  P spring.jpa.hibernate.naming.implicit-strategy (Fully qual... String
  P spring.jpa.hibernate.naming.physical-strategy (Fully qual... String
  P spring.jpa.mapping-resources (Mapping resources (eq... List<String>
  P spring.jpa.open-in-view=true (Register OpenEntityManager... Boolean
  P spring.jpa.properties (Additional native prop... Map<String, String>
^↓ and ^↑ will move caret down and up in the editor >>
```

# Spring Boot To PCF Developer Workshop

- Lets create some RESTful end points for our Entity which we can do using Spring Rest Repositories
- Add an Interface as shown below
- Note the use of “@RepositoryRestResource” annotation. Spring Data REST exposes a collection resource named after the uncapitalized, pluralized version of the domain class in this case “employee”. This is all done using Spring HATEOAS which it’s goal is all about giving the client as much information as possible to be able to navigate the API

```
package com.example.springboottopcf;

import org.springframework.data.jpa.repository.JpaRepository;
import org.springframework.data.rest.webmvc.RepositoryRestController;

@RepositoryRestController
public interface EmployeeRepository extends JpaRepository <Employee, Long> {
}
```

# Spring Boot To PCF Developer Workshop

- Spring Boot Actuator is a sub-project of Spring Boot . It adds several Spring Boot's production-ready features to your application with little effort on your part.
  - <https://docs.spring.io/spring-boot/docs/current/reference/htmlsingle/#production-ready>
- Lets add it to our pom.xml as follows, yep that's all we need to do and we get RESTful end points added to our application automatically.

```
<dependency>
  <groupId>org.springframework.boot</groupId>
  <artifactId>spring-boot-starter-actuator</artifactId>
</dependency>
```

# Spring Boot To PCF Developer Workshop

- We will also need to add the following “management” properties to our “application.properties” file as shown below to ensure our actuator endpoints are enabled and accessible.

```
spring.application.name=My First Spring Boot Application  
spring.jpa.hibernate.ddl-auto=none  
spring.datasource.initialization-mode=always  
management.endpoint.health.enabled=true  
management.endpoint.health.show-details=always  
management.endpoints.enabled-by-default=true  
management.endpoints.web.exposure.include=*
```

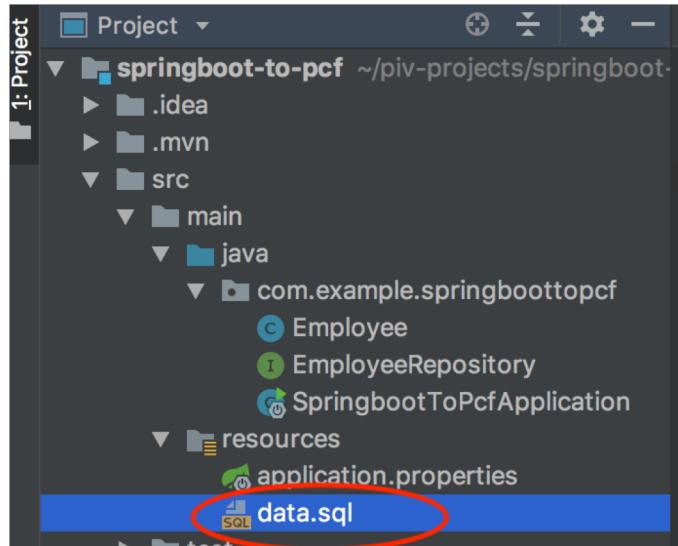
# Spring Boot To PCF Developer Workshop

- Spring Boot Actuator Endpoints can be accessed using /actuator/{name} with examples below
  - beans: Displays a complete list of all the Spring beans in your application
  - env: Exposes properties from Spring's ConfigurableEnvironment
  - health: Shows application health information
  - autoconfig: Displays an auto-configuration report showing all auto-configuration candidates and the reason why they 'were' or 'were not' applied
  - metrics: Shows 'metrics' information for the current application
  - trace: Displays trace information (by default the last 100 HTTP requests)
  - mappings: Displays a collated list of all @RequestMapping paths

Many more ...

# Spring Boot To PCF Developer Workshop

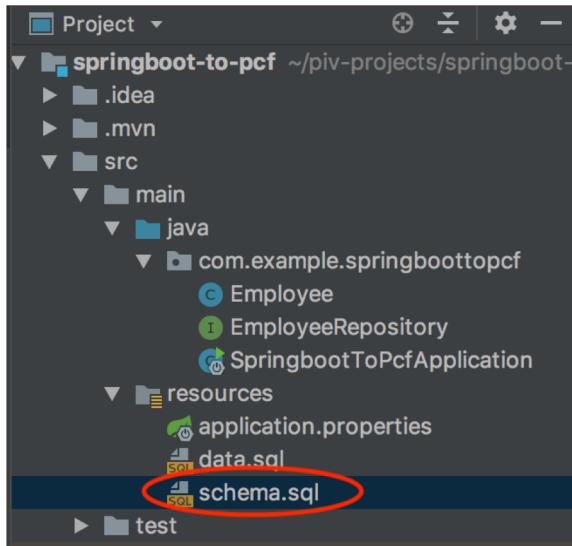
- Now lets add some Employee data through our Entity. There are various ways but using Spring Boot we can add a “data.sql” file with some SQL inserts which will be inserted at runtime for us.
- Create a file called “data.sql” under the resources folder as follows



```
insert into employee (name) values ('pas');  
insert into employee (name) values ('lucia');  
insert into employee (name) values ('lucas');  
insert into employee (name) values ('siena');
```

# Spring Boot To PCF Developer Workshop

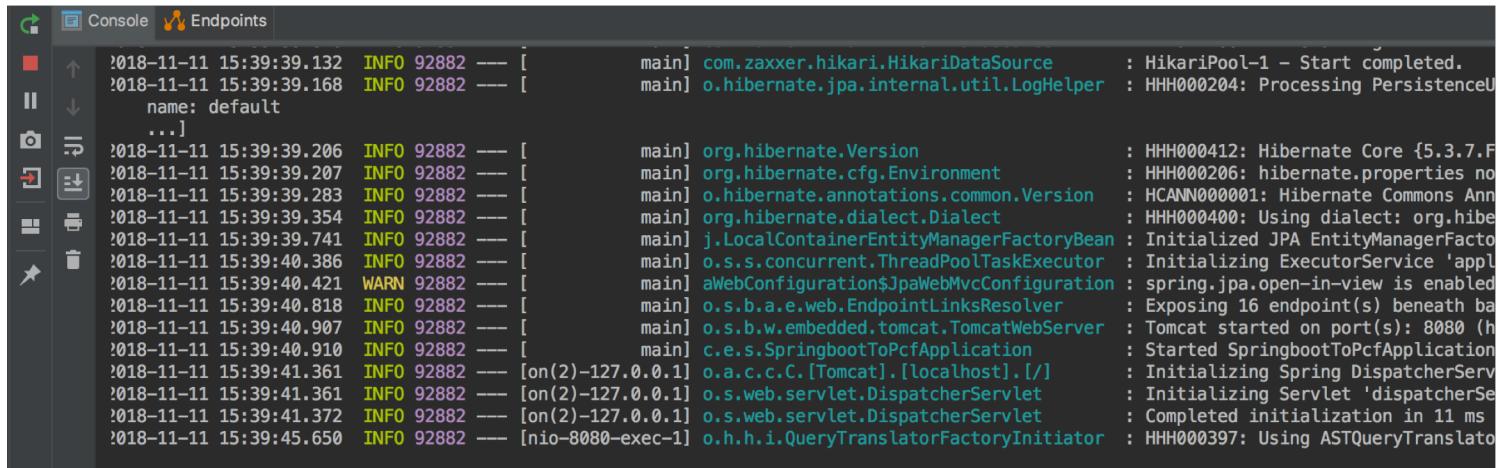
- Like we created a data.sql file a schema.sql file is required as we explicitly told hibernate not to create the table data for us and so Spring Boot is doing the schema/table initialization
- Create a file called “schema.sql” under the resources folder as follows



```
DROP TABLE IF EXISTS employee;  
  
create table employee  
(id bigint NOT NULL AUTO_INCREMENT PRIMARY KEY,  
name VARCHAR(30))AUTO_INCREMENT=1;
```

# Spring Boot To PCF Developer Workshop

- Now lets run our Spring Boot application by right clicking on “SpringbootToPcfApplication” and selecting “Run”
- Few things to Note:
  - Tomcat is bundled and used to run our application as per the spring starter pom.xml entries
  - data.sql will be run once JPA has created the underlying table's in the database
  - Various Spring Boot Auto Configurations have been run

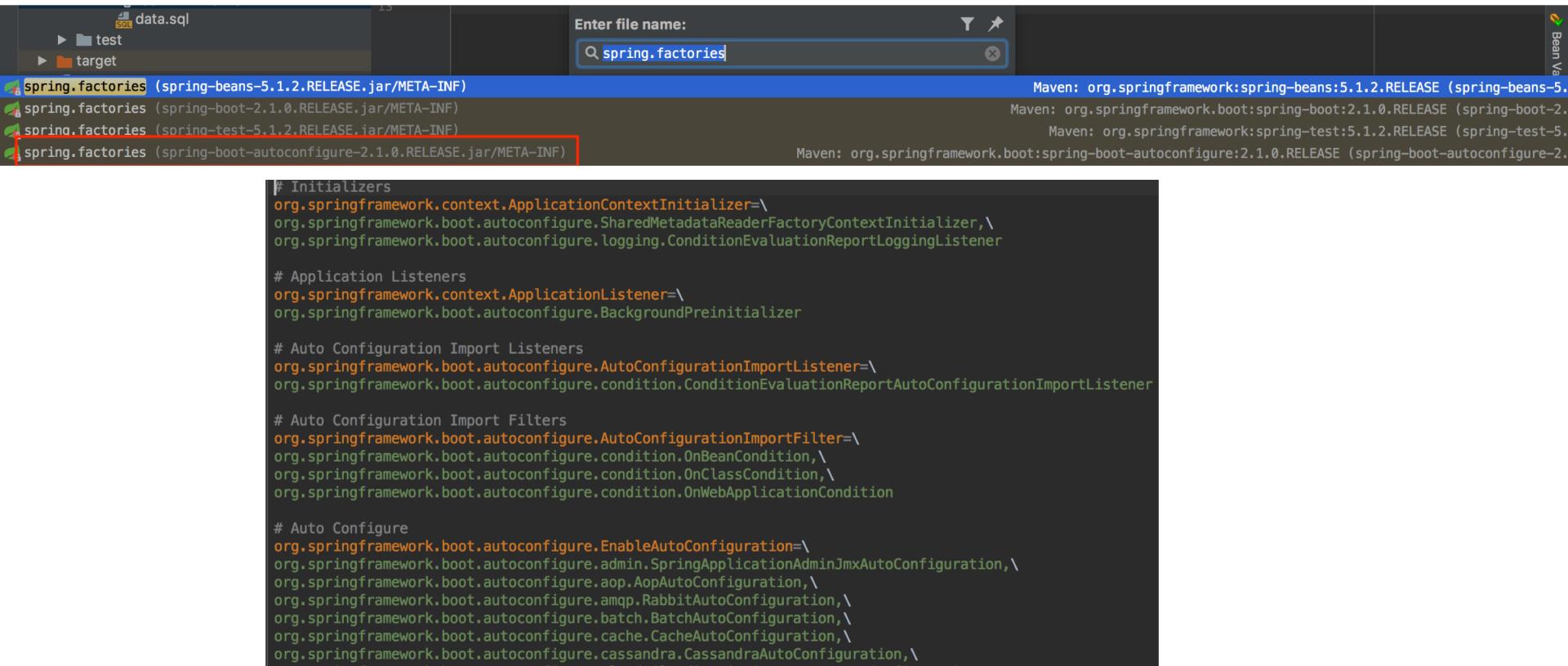


The screenshot shows a terminal window with two tabs: "Console" and "Endpoints". The "Console" tab is active, displaying a log of application startup messages. The log includes:

- 2018-11-11 15:39:39.132 INFO 92882 --- [ main] com.zaxxer.hikari.HikariDataSource : HikariPool-1 - Start completed.
- 2018-11-11 15:39:39.168 INFO 92882 --- [ main] o.hibernate.jpa.internal.util.LogHelper : HHH000204: Processing PersistenceUnitInfo...
- 2018-11-11 15:39:39.206 INFO 92882 --- [ main] org.hibernate.Version : HHH000412: Hibernate Core {5.3.7.Final}
- 2018-11-11 15:39:39.207 INFO 92882 --- [ main] org.hibernate.cfg.Environment : HHH000206: hibernate.properties not found, using defaults
- 2018-11-11 15:39:39.283 INFO 92882 --- [ main] o.hibernate.annotations.common.Version : HCANN000001: Hibernate Commons Annotations {5.0.1.Final}
- 2018-11-11 15:39:39.354 INFO 92882 --- [ main] org.hibernate.dialect.Dialect : HHH000400: Using dialect: org.hibernate.dialect.MySQL5InnoDBDialect
- 2018-11-11 15:39:39.741 INFO 92882 --- [ main] j.LocalContainerEntityManagerFactoryBean : HHH000741: EntityManagerFactory 'application' initialized
- 2018-11-11 15:39:40.386 INFO 92882 --- [ main] o.s.s.concurrent.ThreadPoolExecutor : HHH000408: Initializing ExecutorService 'applicationExecutor'
- 2018-11-11 15:39:40.421 WARN 92882 --- [ main] aWebConfiguration\$JpaWebMvcConfiguration : HHH000408: Initializing ExecutorService 'applicationExecutor'
- 2018-11-11 15:39:40.818 INFO 92882 --- [ main] o.s.b.a.e.web.EndpointLinksResolver : HHH000408: Initializing ExecutorService 'applicationExecutor'
- 2018-11-11 15:39:40.907 INFO 92882 --- [ main] o.s.b.w.embedded.tomcat.TomcatWebServer : HHH000408: Initializing ExecutorService 'applicationExecutor'
- 2018-11-11 15:39:40.910 INFO 92882 --- [ main] c.e.s.SpringbootToPcfApplication : HHH000408: Initializing ExecutorService 'applicationExecutor'
- 2018-11-11 15:39:41.361 INFO 92882 --- [on(2)-127.0.0.1] o.a.c.c.C.[Tomcat].[localhost].[/] : HHH000408: Initializing ExecutorService 'applicationExecutor'
- 2018-11-11 15:39:41.361 INFO 92882 --- [on(2)-127.0.0.1] o.s.web.servlet.DispatcherServlet : HHH000408: Initializing ExecutorService 'applicationExecutor'
- 2018-11-11 15:39:41.372 INFO 92882 --- [on(2)-127.0.0.1] o.s.web.servlet.DispatcherServlet : HHH000408: Initializing ExecutorService 'applicationExecutor'
- 2018-11-11 15:39:45.650 INFO 92882 --- [nio-8080-exec-1] o.h.h.i.QueryTranslatorFactoryInitiator : HHH000397: Using ASTQueryTranslatorFactory

# Spring Boot To PCF Developer Workshop

- Various Spring Boot Auto Configurations have been run the full list can be found using “Command - SHIFT - O” and finding the file “spring.factories” in the spring auto configuration JAR file



The screenshot shows a terminal window with the following details:

- File Explorer sidebar: Shows files like `data.sql`, `test`, and `target`.
- Search bar: Displays the search term `spring.factories`.
- Terminal content:

```
# Initializers
org.springframework.context.ApplicationContextInitializer\
org.springframework.boot.autoconfigure.SharedMetadataReaderFactoryContextInitializer,\
org.springframework.boot.autoconfigure.logging.ConditionEvaluationReportLoggingListener

# Application Listeners
org.springframework.context.ApplicationListener\
org.springframework.boot.autoconfigure.BackgroundPreinitializer

# Auto Configuration Import Listeners
org.springframework.boot.autoconfigure.AutoConfigurationImportListener\
org.springframework.boot.autoconfigure.condition.ConditionEvaluationReportAutoConfigurationImportListener

# Auto Configuration Import Filters
org.springframework.boot.autoconfigure.AutoConfigurationImportFilter\
org.springframework.boot.autoconfigure.condition.OnBeanCondition,\
org.springframework.boot.autoconfigure.condition.OnClassCondition,\
org.springframework.boot.autoconfigure.condition.OnWebApplicationCondition

# Auto Configure
org.springframework.boot.autoconfigure.EnableAutoConfiguration\
org.springframework.boot.autoconfigure.admin.SpringApplicationAdminJmxAutoConfiguration,\
org.springframework.boot.autoconfigure.aop.AopAutoConfiguration,\
org.springframework.boot.autoconfigure.amqp.RabbitAutoConfiguration,\
org.springframework.boot.autoconfigure.batch.BatchAutoConfiguration,\
org.springframework.boot.autoconfigure.cache.CacheAutoConfiguration,\
org.springframework.boot.autoconfigure.cassandra.CassandraAutoConfiguration,\
```
- Maven dependency information on the right:

  - `spring-beans-5.1.2.RELEASE.jar/META-INF/spring.factories`: Maven: `org.springframework:spring-beans:5.1.2.RELEASE` (`spring-beans-5.1.2.RELEASE.jar`)
  - `spring-boot-2.1.0.RELEASE.jar/META-INF/spring.factories`: Maven: `org.springframework.boot:spring-boot:2.1.0.RELEASE` (`spring-boot-2.1.0.RELEASE.jar`)
  - `spring-test-5.1.2.RELEASE.jar/META-INF/spring.factories`: Maven: `org.springframework:spring-test:5.1.2.RELEASE` (`spring-test-5.1.2.RELEASE.jar`)
  - `spring-boot-autoconfigure-2.1.0.RELEASE.jar/META-INF/spring.factories`: Maven: `org.springframework.boot:spring-boot-autoconfigure:2.1.0.RELEASE` (`spring-boot-autoconfigure-2.1.0.RELEASE.jar`)

# Spring Boot To PCF Developer Workshop

- Using either CURL, HTTPie or even a browser you can hit the main endpoint as follows.
  - http://localhost:8080/employees
- Various Actuator end points exist using:
  - /actuator/health, /actuator/autoconfig, /actuator/beans, /actuator/threaddump, /actuator/env , /actuator/mappings ,

```
pasapicella@pas-macbook:~$ http http://localhost:8080/actuator/health
HTTP/1.1 200
Content-Type: application/vnd.spring-boot.actuator.v2+json;charset=UTF-8
Date: Sun, 11 Nov 2018 04:52:54 GMT
Transfer-Encoding: chunked

{
  "details": {
    "db": {
      "details": {
        "database": "H2",
        "hello": 1
      },
      "status": "UP"
    },
    "diskSpace": {
      "details": {
        "free": 106433597440,
        "threshold": 10485760,
        "total": 500068036608
      },
      "status": "UP"
    }
  },
  "status": "UP"
}
```

```
pasapicella@pas-macbook:~$ http http://localhost:8080/employees
HTTP/1.1 200
Content-Type: application/hal+json; charset=UTF-8
Date: Sun, 11 Nov 2018 04:51:19 GMT
Transfer-Encoding: chunked

{
  "_embedded": {
    "employees": [
      {
        "_links": {
          "employee": {
            "href": "http://localhost:8080/employees/1"
          },
          "self": {
            "href": "http://localhost:8080/employees/1"
          }
        },
        "name": "pas"
      },
      {
        "_links": {
          "employee": {
            "href": "http://localhost:8080/employees/2"
          },
          "self": {
            "href": "http://localhost:8080/employees/2"
          }
        },
        "name": "lucia"
      }
    ]
  }
}
```

# Spring Boot To PCF Developer Workshop

- Had trouble along the way? The whole source code is on GitHub
  - <https://github.com/papicella/springboot-to-pcf>

The screenshot shows the GitHub repository page for 'papicella / springboot-to-pcf'. The repository has 1 commit, 1 branch, 0 releases, and 1 contributor. The latest commit was made 2 minutes ago. The repository contains files such as .mvn/wrapper, src, generate-load.sh, manifest.yml, mvnw, mvnw.cmd, and pom.xml, all of which were updated 2 minutes ago. A button to 'Add a README' is visible at the bottom.

No description, website, or topics provided.

Manage topics

1 commit 1 branch 0 releases 1 contributor

Branch: master New pull request Create new file Upload files Find file Clone or download

papicella Spring Boot to PCF Workshop

File	Description	Updated
.mvn/wrapper	Spring Boot to PCF Workshop	2 minutes ago
src	Spring Boot to PCF Workshop	2 minutes ago
generate-load.sh	Spring Boot to PCF Workshop	2 minutes ago
manifest.yml	Spring Boot to PCF Workshop	2 minutes ago
mvnw	Spring Boot to PCF Workshop	2 minutes ago
mvnw.cmd	Spring Boot to PCF Workshop	2 minutes ago
pom.xml	Spring Boot to PCF Workshop	2 minutes ago

Help people interested in this repository understand your project by adding a README.

Add a README

© 2018 GitHub, Inc. Terms Privacy Security Status Help Contact GitHub Pricing API Training Blog About

Spring Boot to PCF Developer Workshop

---

# What Cloud Native Platform can we deploy our employee microservice too?

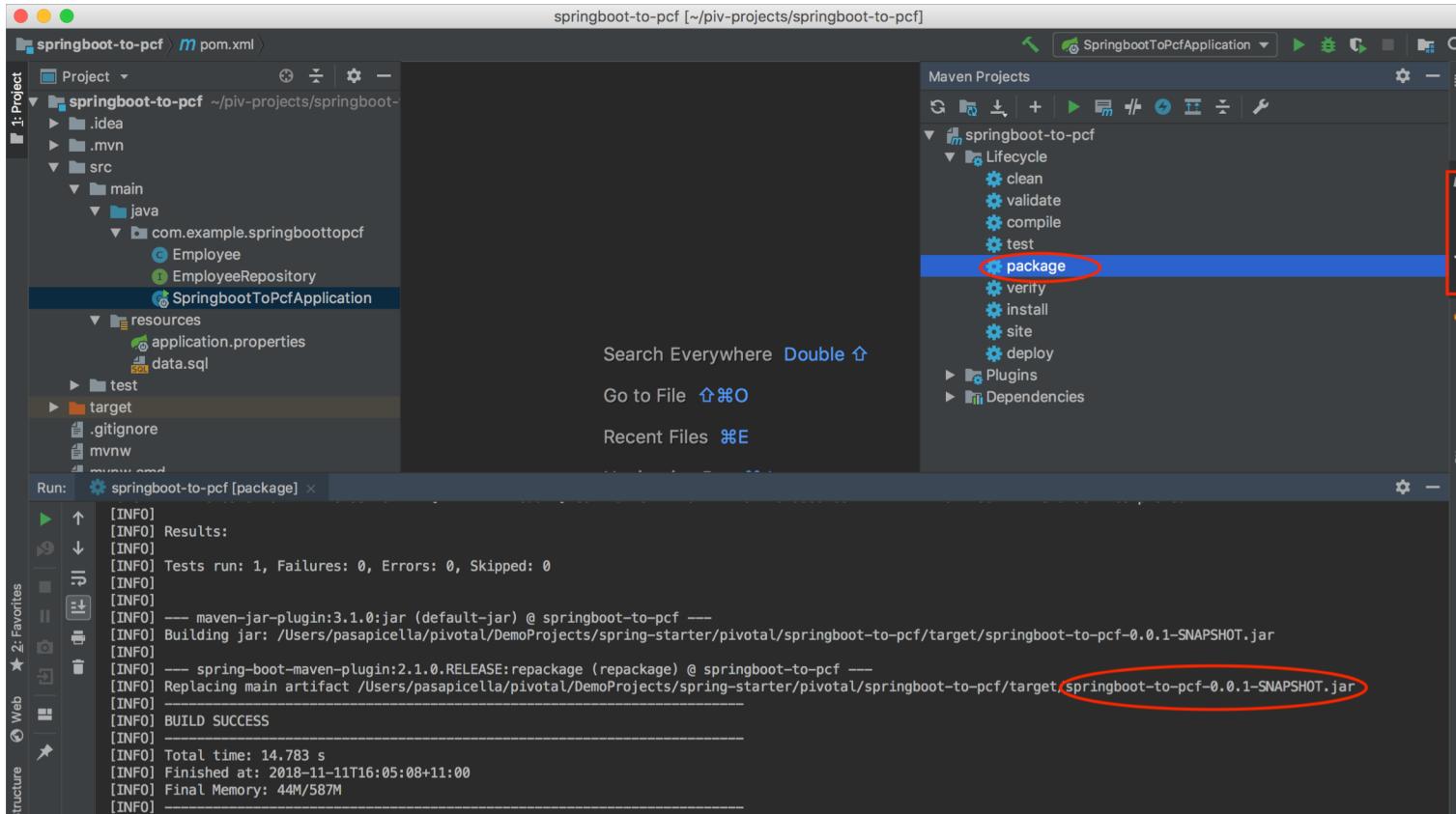
Spring Boot to PCF Developer Workshop

---

# Well Pivotal Cloud Foundry of course

# Spring Boot To PCF Developer Workshop

- Using “Maven Projects” tab package the application as shown below



# Spring Boot To PCF Developer Workshop

- Installing the Cloud Foundry CLI

[{“cf help” : “provides command line options such as push etc”,  
“cf help <command>” : “provides help for each command option”}]

- Install using Github project as follows

- \$ brew install cloudfoundry/tap(cf-cli)

- Easily install using BREW if on a MAC

```
$ brew tap cloudfoundry/tap
```

```
$ brew install cf-cli
```

```
pasapicella@pas-macbook:~$ cf --version  
cf version 6.39.0+607d4f8be.2018-09-11
```

# Spring Boot To PCF Developer Workshop

- Can also download directly from Application Manager UI as shown below, make sure you select the right O/S for your laptop

The screenshot shows the Pivotal Web Services Application Manager UI. On the left, there's a sidebar with 'Pivotal Web Services' logo, a search bar, and navigation links for 'Home' and 'Marketplace'. The main content area has a dark header 'Tools' with a sub-header: 'Cloud Foundry CLI for pushing and managing apps, creating and binding services, and more. For more info visit the [cf documentation](#)'. Below this, there's a section titled 'Download and Install the CLI' with a button 'Download for Mac OS X 64 bit'. Underneath, there are three code snippets in boxes: '\$ cf login -a api.run.pivotal.io' for logging in, '\$ cf help' for getting help, and '\$ cf push your\_app' for pushing an application. At the bottom of the main content area is a blue button 'VIEW THE GETTING STARTED TUTORIAL'. In the bottom-left corner of the sidebar, the word 'Tools' is circled in red.

# Spring Boot To PCF Developer Workshop

- Target cloud using “cf api” as shown below
- Only a single PUBLIC Instance of PWS exists in the USA, BUT PCF can also be installed on AWS, GCP, Azure or within your Firewall locally on VMware or OpenStack

[{“USA Instance” : “https://api.run.pivotal.io”}]

```
pasapicella@pas-macbook:~$ cf api https://api.run.pivotal.io
Setting api endpoint to https://api.run.pivotal.io...
OK

api endpoint:    https://api.run.pivotal.io
api version:    2.125.0
Not logged in. Use 'cf login' to log in.
```

# Spring Boot To PCF Developer Workshop

- Login using “cf login” as shown below. Your username, organization name and space will be different than the example below of course. You should already have login details and those details available to be able to perform a login

```
[{"USAGE": "cf login [-a API_URL] [-u USERNAME] [-p PASSWORD] [-o ORG] [-s SPACE]"}]
```

- \$ cf login -u papicella@pivotal.io -p \*\*\*\*\* -o papicella-org -s apple

```
papicella@pas-macbook:~$ cf login -u papicella@pivotal.io -o papicella-org -s apple
API endpoint: https://api.run.pivotal.io
```

```
Password>
Authenticating...
OK

Targeted org papicella-org

Targeted space apple
```

```
API endpoint: https://api.run.pivotal.io (API version: 2.125.0)
User: papicella@pivotal.io
Org: papicella-org
Space: apple
```

# Spring Boot To PCF Developer Workshop

- Deploy using “cf push” as shown below

```
$ cf push employee-api -p ./target/springboot-to-pcf-0.0.1-SNAPSHOT.jar -i 1 -m 1g --random-route
```

```
Waiting for app to start...

name:          employee-api
requested state:  started
routes:         employee-api-excellent-eland.cfapps.io
last uploaded:  Sun 11 Nov 16:25:37 AEDT 2018
stack:          cflinuxfs2
buildpacks:    client-certificate-mapper=1.8.0_RELEASE container-security-provider=1.16.0_RELEASE java-
               java-main java-opts java-security jvmkill-agent=1.16.0_RELEASE open-jd...
               ...
type:          web
instances:     1/1
memory usage:  1024M
start command: JAVA_OPTS="-agentpath:$PWD/.java-buildpack/open_jdk_jre/bin/jvmbi...
               -Djava.ext.dirs=$PWD/.java-buildpack/container_security_provider:$PWD/.java-buildpack/o...
               $JAVA_OPTS" && CALCULATED_MEMORY=$(($PWD/.java-buildpack/open_jdk_jre/bin/java-buildpac...
               -stackThreads=250 -vmOptions="$JAVA_OPTS") && echo JVM Memory Configuration: $CALCULATED...
               exec $PWD/.java-buildpack/open_jdk_jre/bin/java $JAVA_OPTS -cp $PWD/. org.springframework...
               ...
state      since        cpu      memory       disk
#0  running  2018-11-11T05:26:10Z  129.9%  147.8M of 1G  163.2M of 1G
```

# Spring Boot To PCF Developer Workshop

- Navigate to Pivotal Applications Manager UI and verify your deployed application is up and running
  - <http://run.pivotal.io>

The screenshot shows the Pivotal Web Services Applications Manager UI. The top navigation bar includes the Pivotal Web Services logo, a search bar, and a user account dropdown for `papicella@pivotal.io`. The main page displays the `employee-api` application under the `papicella-org` organization. The application status is shown as "Running". The left sidebar has sections for Home, Marketplace, Tools, Docs, Support, Blog, and Status. The main content area has tabs for Overview, Services, Route (1), Logs, Tasks, Trace, Threads, and Settings. The Overview tab is selected, showing the following details:

- Events:** Started app (papicella@pivotal.io 11/11/2018 at 04:25:12 PM), Mapped route to app (papicella@pivotal.io 11/11/2018 at 04:24:36 PM), and Created app (papicella@pivotal.io 11/11/2018 at 04:24:36 PM).
- App Summary:** Last Push: 04:25 PM 11/11/18.
- Instances / Allocated:** 1 / 1.
- Memory / Allocated:** 0.25 / 1.00 GB.
- Disk / Allocated:** 0.16 / 1.00 GB.
- Processes and Instances:** A table for the "web" process showing 1 instance with 1 GB allocated memory and 1 GB disk. An "Autoscaling" toggle is off. Columns include #, APP HEALTH, CPU, Memory, Disk, and Uptime. One row shows 0 instances.
- Health Check:** Status: UP for db, database: H2, hello: 1, and diskSpace.
- Logs:** A section showing log entries for the application.

# Spring Boot To PCF Developer Workshop

- Can even view applications using the CF CLI as shown below

- cf apps
- cf app employee-api

```
pasapicella@pas-macbook:~$ cf apps
Getting apps in org papicella-org / space apple as papicella@pivotal.io...
OK

name      requested state  instances   memory   disk    urls
employee-api  started        1/1       1G        1G  employee-api-excellent-eland.cfapps.io
pasapicella@pas-macbook:~$ cf app employee-api

Showing health and status for app employee-api in org papicella-org / space apple as papicella@pivotal.io...

name:            employee-api
requested state: started
routes:          employee-api-excellent-eland.cfapps.io
last uploaded:  Sun 11 Nov 16:25:37 AEDT 2018
stack:           cflinuxfs2
buildpacks:      client-certificate-mapper=1.8.0_RELEASE container-security-provider=1.16.0_RELEASE java-bu
                  java-main java-opts java-security jvmkill-agent=1.16.0_RELEASE open-jd...
type:            web
instances:       1/1
memory usage:   1024M
state          since             cpu   memory         disk
#0   running  2018-11-11T05:26:10Z  2.7%  262.7M of 1G  163.2M of 1G
```

# Spring Boot To PCF Developer Workshop

- Lets just view one RESTful endpoint
  - \$ http http://{url}/employees/2

Note: The URL can be obtained using “cf apps” as per the output in the previous slide.

```
pasapicella@pas-macbook:~$ http http://employee-api-excellent-eland.cfapps.io/employees/2
HTTP/1.1 200 OK
Connection: keep-alive
Content-Length: 239
Content-Type: application/hal+json; charset=UTF-8
Date: Sun, 11 Nov 2018 05:33:19 GMT
X-Vcap-Request-Id: f23a1bda-029a-4f6d-62a1-b6b59fbe94ae

{
  "_links": {
    "employee": {
      "href": "http://employee-api-excellent-eland.cfapps.io/employees/2"
    },
    "self": {
      "href": "http://employee-api-excellent-eland.cfapps.io/employees/2"
    }
  },
  "name": "lucia"
}
```

# Spring Boot To PCF Developer Workshop

- So how can we use a different RDBMS service like MySQL rather than H2?

Lets start by adding MySQL Dependency to the maven pom.xml as shown below and then repackage to create a new JAR file. Put the MySQL driver before the H2 driver dependency

```
<dependency>
  <groupId>mysql</groupId>
  <artifactId>mysql-connector-java</artifactId>
</dependency>
<dependency>
  <groupId>com.h2database</groupId>
  <artifactId>h2</artifactId>
  <scope>runtime</scope>
</dependency>
```

- That's it, Pivotal Cloud Foundry's buildpack along with Spring Boot will expect a MySQL database service to be bound to the application and then will create a DataSource to MySQL. If such a service isn't bound it will use H2 instead without any code changes!!!

# Spring Boot To PCF Developer Workshop

- Now using the Pivotal Applications Manager UI click on “Marketplace” and browse for “mysql” database instances as shown below

The screenshot shows the Pivotal Web Services Marketplace interface. The left sidebar has a 'Marketplace' link, which is circled in red. The main area is titled 'Marketplace' and shows a search bar with 'papicella-org' selected. A search query 'mysql' is entered in the search bar, also circled in red. Below the search bar, there is a message: 'Get started with our free marketplace services. Upgrade select plans to gain access to premium service plans.' Two service options are listed:

- ClearDB MySQL Database**: Highly available MySQL for your Apps. This option is highlighted with a red box.
- MySQL for Pivotal Cloud Foundry**: MySQL databases on demand.

# Spring Boot To PCF Developer Workshop

- Click on one of the “mysql” instances in the next few steps we select the Pivotal MySQL service. Once selected click on “Select This Plane” button

The screenshot shows the Pivotal Web Services Marketplace interface. On the left, there's a sidebar with 'Home' and 'Marketplace' buttons. The main area displays the 'MySQL for Pivotal Cloud Foundry' service page. At the top, there's a search bar and a 'press /' button. The service name is 'MySQL for Pivotal Cloud Foundry' with the subtitle 'MySQL databases on demand'. Below the service name are 'Docs' and 'Support' links. There are three plan options: '100 MB free', '1 GB free', and '20 GB free'. A red circle highlights the 'SELECT THIS PLAN' button for the '100 MB free' plan.

Pivotal Web Services  Search apps, services, spaces, & orgs press /

Home

Marketplace

SERVICE

**MySQL for Pivotal Cloud Foundry**

MySQL databases on demand

Docs Support

**100 MB**  
free

**1 GB**  
free

**20 GB**  
free

ABOUT THIS SERVICE

Creating a service instance provisions a database. Binding applications provisions unique credentials for each application to access the database.

COMPANY

Pivotal Software

**100 MB free**

- Shared MySQL server
- 100 MB storage
- 20 concurrent connections

**SELECT THIS PLAN**

# Spring Boot To PCF Developer Workshop

- In this screen lets give our service the following details then click create

Name: pas-mysql

Bind To App: [do not bind]

The screenshot shows the Pivotal Web Services Marketplace interface. On the left, there's a sidebar with 'Home' and 'Marketplace' buttons. The main area displays the 'MySQL for Pivotal Cloud Foundry' service, which is described as 'MySQL databases on demand'. It offers a '100 MB free' plan, which includes a Shared MySQL server, 100 MB storage, and 20 concurrent connections. The 'Instance Configuration' section contains fields for 'Instance Name' (set to 'pas-mysql'), 'Add To Space' (set to 'apple'), and 'Bind To App (Optional)' (set to '[do not bind]'). A red oval highlights the 'Instance Name' field, another red box highlights the 'Bind To App (Optional)' field, and a third red circle highlights the 'CREATE' button at the bottom right.

Pivotal Web Services  Search apps, services, spaces, & orgs press / papicella@pivotal.io

Home

Marketplace

SERVICE  
MySQL for Pivotal Cloud Foundry  
MySQL databases on demand

ABOUT THIS SERVICE  
Creating a service instance provisions a database. Binding applications provisions unique credentials for each application to access the database.

COMPANY  
Pivotal Software

100 MB  
free

- Shared MySQL server
- 100 MB storage
- 20 concurrent connections

Docs Support

Instance Configuration

Instance Name  
pas-mysql

Add To Space  
apple

Bind To App (Optional)  
[do not bind]

SHOW ADVANCED OPTIONS

CANCEL CREATE

# Spring Boot To PCF Developer Workshop

- Verify it has successfully created our MySQL service using Pivotal Applications Manager UI as well as the CF CLI as shown below

The screenshot shows the Pivotal Web Services Applications Manager interface. The top navigation bar includes the Pivotal logo, 'Pivotal Web Services', a search bar ('Search apps, services, spaces, & orgs'), a user icon ('press /'), and the email 'papicella@pivotal.io'. The left sidebar has links for 'Home' and 'Marketplace'. The main content area shows a space named 'apple' with one running app ('apple') and no stopped or crashed apps. The 'Service (1)' tab is selected, showing a single MySQL service named 'pas-mysql' with a 'free - 100 MB' plan. A blue button 'ADD A SERVICE' is visible on the right.

```
pasapicella@pas-macbook:~$ cf services
Getting services in org papicella-org / space apple as papicella@pivotal.io...
name      service   plan      bound apps      last operation
pas-mysql  p-mysql   100mb    create succeeded
```

# Spring Boot To PCF Developer Workshop

- Now returning to our project let's create a file called "manifest.yml" with contents as shown below

```
---
```

**applications:**

- **name:** employee-api
- memory:** 1G
- instances:** 1
- random-route:** true
- path:** ./target/springboot-to-pcf-0.0.1-SNAPSHOT.jar

**services:**

- pas-mysql

The screenshot shows a code editor with a dark theme. On the left is a project tree for a Spring Boot application named "springboot-to-pcf". The tree includes .idea, .mvn, src (with main and java subfolders), resources (with application.properties and data.sql), test, target, .gitignore, and manifest.yml. The manifest.yml file is open in the editor on the right. The code in manifest.yml is as follows:

```
---
```

```
applications:
```

```
  - name: employee-api
```

```
    memory: 1G
```

```
    instances: 1
```

```
    random-route: true
```

```
    path: ./target/springboot-to-pcf-0.0.1-SNAPSHOT.jar
```

```
services:
```

```
  - pas-mysql
```

# Spring Boot To PCF Developer Workshop

- Now let's re-push our application which should have been packaged with the MySQL driver as per a previous step so it will use the MySQL database service we created

```
$ cf push
```

```
Waiting for app to start...

name:          employee-api
requested state: started
routes:         employee-api-excellent-eland.cfapps.io
last uploaded:  Sun 11 Nov 18:03:45 AEDT 2018
stack:          cflinuxfs2
buildpacks:
  client-certificate-mapper=1.8.0_RELEASE container-securit
    java-main java-opts java-security jvmkill-agent=1.16.0_RB

type:           web
instances:      1/1
memory usage:   1024M
start command:  JAVA_OPTS="-agentpath:$PWD/.java-buildpack/open_jdk_jre/bin
                  -Djava.ext.dirs=$PWD/.java-buildpack/container_security_pro
                  $JAVA_OPTS" && CALCULATED_MEMORY=$(($PWD/.java-buildpack/o
                  -stackThreads=250 -vmOptions="$JAVA_OPTS") && echo JVM Memo
                  exec $PWD/.java-buildpack/open_jdk_jre/bin/java $JAVA_OPTS

                     state      since          cpu      memory      disk
#0   running    2018-11-11T07:04:14Z  0.0%  145.5M of 1G  165.3M of 1G
```

# Spring Boot To PCF Developer Workshop

- And that's it our Spring Boot application using Hibernate can easily switch to another RDBMS and the cloud foundry buildpack along with Spring Boot will automatically create a Data Source from the bound service for us

View all employees using `http://{URL}/employees`

```
pasapicella@pas-macbook:~$ http http://employee-api-accountable-hartebeest.cfapps.io/employees/4
HTTP/1.1 200 OK
Connection: keep-alive
Content-Length: 251
Content-Type: application/hal+json; charset=UTF-8
Date: Sun, 11 Nov 2018 10:00:00 GMT
X-Vcap-Request-Id: e7715169-782d-479c-6daf-e17fa601e735

{
  "_links": {
    "employee": {
      "href": "http://employee-api-accountable-hartebeest.cfapps.io/employees/4"
    },
    "self": {
      "href": "http://employee-api-accountable-hartebeest.cfapps.io/employees/4"
    }
  },
  "name": "pas"
}
```

# Spring Boot To PCF Developer Workshop

- I suppose we better prove we are actually using the MySQL database. To do that let's deploy Pivotal MySQL\*Web using the GitHub URL below. Instructions on how to deploy are on the GitHub link
  - <https://github.com/pivotal-cf/PivotalMySQLWeb>

The screenshot shows the GitHub repository page for 'pivotal-cf / PivotalMySQLWeb'. The repository has 149 commits, 1 branch, 0 releases, 2 contributors, and is licensed under Apache-2.0. The README.md file contains the text 'PivotalMySQLWeb is a free Pivotal open source project, intended to handle the administration of a Pivotal MySQL Service Instance over the Web'.

File	Description	Time Ago
.mvn/wrapper	Add maven wrapper for convenience	7 months ago
src	Added support for GCP Cloud MySQL	8 months ago
LICENSE	Added Pivotal License Files	2 years ago
NOTICE	Added Pivotal License Files	2 years ago
README.md	Merge branch 'master' of https://github.com/pivotal-cf/PivotalMySQLWeb	5 months ago
manifest.yml	Added Tests for IndexDAOImpl class	a year ago
mvnw	Add maven wrapper for convenience	7 months ago
mvnw.cmd	Merge branch 'master' of https://github.com/pivotal-cf/PivotalMySQLWeb	5 months ago
package-no-tests.sh	Moved to Spring Boot 1.4.2 release	2 years ago
pom.xml	Added HTTP Basic Authentication using Spring Security	a year ago

# Spring Boot To PCF Developer Workshop

- Once Pivotal MYSQL\*Web is deployed accessing it using it's URL and you get to a home page as follows providing we have bound the application to the same service name our "employee-api" using which is "pas-mysql"

The screenshot shows the Pivotal MySQL\*Web application interface. At the top, there is a navigation bar with links for Home, Logout, Menu, Schema Objects, Themes, and a user session indicator. Below the navigation bar, a green header bar displays the message "Success! Connected to MySQL using P-MYSQL bound instance". The main content area has a dark background. On the left, there is a sidebar with a summary of database objects: Tables[1], Views[0], Indexes[1], Constraints[1], and a SQL Worksheet link. Below this, a paragraph of text is displayed, followed by a note from Pivotal. On the right, there is a "Schema Objects" section with a list of items: Tables[1], Views[0], Indexes[1], and Constraints[1]. Below this is a "Select Theme" section with options: Default, Sandstone, Cyborg, Slate, and Spacelab. At the bottom, there is a "Current Databases" section showing a table with two rows:

database	charset	collation
cf_a7220396_4545_42e7_bafe_22664d6c5d7e	utf8	utf8_unicode_ci
information_schema	utf8	utf8_general_ci

At the very bottom of the page, there is a footer with the text "Powered By Pivotal" and "© 2017. Pas Apicella".

# Spring Boot To PCF Developer Workshop

- Click on tables then click on “employee” table and you will see the data has been loaded into the MySQL database and we are not using H2 anymore

The screenshot shows the Pivotal MySQL Web interface. At the top, there's a navigation bar with links for Home, Logout, Menu, Schema Objects, Themes, and a user session indicator. Below the navigation is a main title "Pivotal MySQL\*Web - Tables". The top navigation bar also includes links for Home, Tables[1], Views[0], Indexes[1], Constraints[1], SQL Worksheet, Connections, Endpoints, Information, and Variables.

In the center, there's a search bar with a "Search" button. Below it, a green success message says "Success! Found 1 Table(s)".

The main content area is a table titled "Tables". It has columns for Catalog, Schema, Name, Type, and Action. One row is visible, showing "def" as the Catalog, "cf\_a7220396\_4545\_42e7\_bafe\_226d6c5d7e" as the Schema, "employee" as the Name, "BASE TABLE" as the Type, and a set of icons for Action. The table has a "Show 10 entries" dropdown and a "Search:" input field.

At the bottom, there's a footer with links for "Check All / Uncheck All" and "With selected:", followed by a page navigation section with "Previous", "1", and "Next" buttons. The footer also includes the text "Powered By Pivotal" and "© 2017. Pas Apicella".

# Spring Boot To PCF Developer Workshop

- His our “employee” data

The screenshot shows the Pivotal MySQL Web interface. At the top, there's a navigation bar with links for Home, Logout, Menu, Schema Objects, Themes, and a user session indicator. Below the header, the title "Pivotal MySQL\*Web - EMPLOYEE Table Viewer" is displayed. Underneath, a secondary navigation bar includes links for Home, Tables[1], Views[0], Indexes[1], Constraints[1], SQL Worksheet, Connections, Endpoints, Information, and Variables. The main content area is titled "Sample Data" and contains a table with five rows of employee data. The columns are labeled "id" and "name". The data is as follows:

id	name
4	pas
10	lucia
16	lucas
22	siena

Below the table, a message says "Showing 1 to 4 of 4 entries". At the bottom right, there are buttons for "Previous", "1", and "Next". At the very bottom, there are four buttons labeled "DDL [EMPLOYEE]", "DESCRIBE [EMPLOYEE]", "DETAILS [EMPLOYEE]", and "INDEXES [EMPLOYEE]". In the bottom right corner, there's a "Powered By Pivotal" logo with the text "© 2017. Pas Apicella".

Spring Boot to PCF Developer Workshop

---

# My application on PCF

# Spring Boot To PCF Developer Workshop

- Let's start by navigating to Applications Manager UI. We see we have two applications here. Our employee-api micro service and Pivotal MySQL\*Web. Click on "employee-api" as shown below

The screenshot shows the Pivotal Web Services Applications Manager UI. At the top, there is a navigation bar with a logo, a search bar containing "Search apps, services, spaces, & orgs", a "press" button, and a user account dropdown for "papicella@pivotal.io". Below the navigation bar, the main interface displays information about a space named "apple". It shows 2 RUNNING instances, 0 STOPPED instances, and 0 CRASHED instances. There are tabs for "Apps (2)", "Service (1)", "Routes (5)", "Member (1)", and "Settings". The "Apps" tab is selected, showing a table with two rows. The first row is for the "employee-api" application, which is circled in red. The second row is for the "pivotal-mysqlweb" application. The table columns are: Status, Name, Instances, Memory, Last Push, and Route. The "employee-api" row shows 1 instance, 1 GB memory, and a route at <https://employee-api-accountable-hartebeest.cfapps.io>. The "pivotal-mysqlweb" row shows 1 instance, 1 GB memory, and a route at <https://pivotal-mysqlweb-exhausted-sable.cfapps.io>.

Status	Name	Instances	Memory	Last Push	Route
● Running	employee-api	1	1 GB	27 minutes ago	<a href="https://employee-api-accountable-hartebeest.cfapps.io">https://employee-api-accountable-hartebeest.cfapps.io</a>
● Running	pivotal-mysqlweb	1	1 GB	27 minutes ago	<a href="https://pivotal-mysqlweb-exhausted-sable.cfapps.io">https://pivotal-mysqlweb-exhausted-sable.cfapps.io</a>

# Spring Boot To PCF Developer Workshop

- From here you can view logs, events, routes, env variables etc
- Notice how a Spring ICON appears on this page that's because the application is using the Spring Actuator Library which the Applications Manager then provides further details for us

The screenshot shows the Pivotal Web Services Applications Manager interface. At the top, there is a navigation bar with a logo, the text "Pivotal Web Services", a search bar containing "Search apps, services, spaces, & orgs", and a user dropdown "press [I] papicella@pivotal.io". Below the navigation bar, the left sidebar has "Home" and "Marketplace" options. The main content area shows the "employee-api" application under the "papicella-org / apple" space. The app icon is circled in red. The status is "Running". Below the app name are tabs: Overview (selected), Service (1), Route (1), Logs, Tasks, Trace, Threads, and Settings. The "Overview" tab shows the last push was at 08:54 PM on 11/11/18. The "Events" section lists three entries: "Started app" (by papicella@pivotal.io at 08:54:34 PM), "Mapped route to app" (by papicella@pivotal.io at 08:54:14 PM), and "Created app" (by papicella@pivotal.io at 08:54:13 PM). The "App Summary" section displays resource usage: Instances / Allocated 1 / 1, Memory / Allocated 0.24 / 1.00 GB, and Disk / Allocated 0.16 / 1.00 GB. The "Processes and Instances" section shows a "web" process with 1 instance, 1 GB memory allocated, and 1 GB disk allocated. A "SCALE" button is available for this process. An "Autoscaling" toggle switch is also present. The bottom part of the screen shows detailed metrics for the "web" instance, including CPU, Memory, Disk, and Uptime.

#	APP HEALTH	CPU	Memory	Disk	Uptime
>	Up	1%	247.62 MB	165.26 MB	30 min

# Spring Boot To PCF Developer Workshop

- Here we can view a Health Check for example as shown below

The screenshot shows the Pivotal Web Services dashboard for the `employee-api` application. The application is listed under the `APP` section, showing it is `Running`. The `Overview` tab is selected. In the `Events` section, there are three entries: `Started app`, `Mapped route to app`, and `Created app`. The `App Summary` section provides resource allocation details: `Instances / Allocated: 1 / 1`, `Memory / Allocated: 0.24 / 1.00 GB`, and `Disk / Allocated: 0.16 / 1.00 GB`. The `Processes and Instances` section shows a single `web` process with `Instances: 1`, `Memory Allocated: 1 GB`, and `Disk Allocated: 1 GB`. A `SCALE` button is available for this process. Below this, the `Autoscaling` toggle is off. The `Health Check` section is highlighted with a red box. It displays the following status information:

```
status: UP
db
  status: UP
  database: MySQL
  hello: 1
diskSpace
  status: UP
  free: 900456448
  threshold: 104857600
  total: 1073741824
```

# Spring Boot To PCF Developer Workshop

- Here we can view the application threads clicking on “Threads” as shown below

The screenshot shows the Pivotal Web Services dashboard for the `employee-api` application. The `Threads` tab is selected. The main pane displays thread details for the `Abandoned connection cleanup thread`, which is in a `TIMED_WAITING` state. The stack trace for this thread is as follows:

```
"Abandoned connection cleanup thread" #11
    Thread State: TIMED_WAITING (sleeping) (on object monitor)
    at java.lang.Object.wait(Native Method)
    - waiting on <0x000000007a7368fd> (a java.lang.ref.ReferenceQueue$Lock)
    at java.lang.ref.ReferenceQueue.remove(ReferenceQueue.java:144)
    at com.mysql.cj.jdbc.AbandonedConnectionCleanupThread.run(AbandonedConnectionCleanupThread.java:70)
    at java.util.concurrent.ThreadPoolExecutor.runWorker(ThreadPoolExecutor.java:1149)
    at java.util.concurrent.ThreadPoolExecutor$Worker.run(ThreadPoolExecutor.java:624)
    at java.lang.Thread.run(Thread.java:748)

    Locked ownable synchronizers:
    - <0x00000000793be5ca> (a java.util.concurrent.ThreadPoolExecutor$Worker)
```

Below this, there are two more threads listed:

- `container-0` (TIMED\_WAITING)
- `ContainerBackgroundProcessor[StandardEngine[Tomcat]]` (TIMED\_WAITING)
- `DestroyJavaVM` (RUNNABLE)

# Spring Boot To PCF Developer Workshop

- Here we can view the trace information by clicking on “Traces” as shown below

The screenshot shows the Pivotal Web Services dashboard for the `employee-api` application. The application is listed under the `employee-api` service, status is `Running`. The `Trace` tab is selected. The trace log displays several requests:

```
> #0 21:31:51.101 200 OPTIONS /cloudfoundryapplication/httptrace
< #0 21:31:46.717 200 GET /employees/4
  60ms

Request:
  x-cf-applicationid: ee60fec7-65b7-49d9-a12e-e2abfcc9ebf5
  x-forwarded-proto: http
  x-b3-traceid: 64c8619e0c856657
  x-b3-spanid: 64c8619e0c856657
  x-cf-instanceid: 1f5b6486-c724-4967-5b87-c70b
  x-cf-cap-request-id: fe29f41-b635-4e79-499d-71ff324453af
  x-request-start: 1541932306715
  host: employee-api-accountable-hartebeest.cfapps.io
  x-forwarded-port: 80
  x-cf-instanceindex: 0
  accept-encoding: gzip, deflate
  accept: /*
  user-agent: HTTPie/1.0.0

Response:
  Transfer-Encoding: chunked
  Date: Sun, 11 Nov 2018 10:31:46 GMT
  Content-Type: application/hal+json; charset=UTF-8
  status: 200

> #0 21:31:29.259 200 GET /cloudfoundryapplication/mappings
< #0 21:31:29.241 200 GET /cloudfoundryapplication/health
< #0 21:31:29.216 200 GET /cloudfoundryapplication/info
< #0 21:31:28.129 200 GET /cloudfoundryapplication/mappings
```

The sidebar on the left includes links for Home, Marketplace, Tools, Docs, Support, Blog, and Status. A bottom navigation bar has a `GIVE FEEDBACK` button.

# Spring Boot To PCF Developer Workshop

- Click on the logs tab as shown below to view logs and we can also configure Log Levels without restarting the application

The screenshot shows the Pivotal Web Services dashboard for the application 'employee-api'. The 'Logs' tab is selected. A modal window titled 'Configure Logging Levels' is open, displaying a grid of loggers and their current log levels (OFF, FATAL, ERROR, WARN, INFO, DEBUG, TRACE). The 'ROOT' logger is set to TRACE. Other loggers listed include org.apache.catalina.startup.DigesterFactory, org.apache.catalina.util.LifecycleBase, org.apache.coyote.http11.Http1NioProtocol, org.apache.sshd.common.util.SecurityUtils, org.apache.tomcat.util.net.NioSelectorPool, org.eclipse.jetty.util.component.AbstractLifeCycle, and org.hibernate.validator.internal.util.Version. The modal also contains a 'CONFIGURE LOGGING LEVELS' button and a 'CLOSE' button.

Configure Logging Levels

LOGGER	OFF	FATAL	ERROR	WARN	INFO	DEBUG	TRACE
ROOT	●	●	●	●	●	●	●
org.apache.catalina.startup.DigesterFactory	●	●	●	●	●	●	●
org.apache.catalina.util.LifecycleBase	●	●	●	●	●	●	●
org.apache.coyote.http11.Http1NioProtocol	●	●	●	●	●	●	●
org.apache.sshd.common.util.SecurityUtils	●	●	●	●	●	●	●
org.apache.tomcat.util.net.NioSelectorPool	●	●	●	●	●	●	●
org.eclipse.jetty.util.component.AbstractLifeCycle	●	●	●	●	●	●	●
org.hibernate.validator.internal.util.Version	●	●	●	●	●	●	●

press [ ] papicella@pivotal.io ▾

Home / papicella-org / apple / employee-api

APP employee-api [ ] [ ] [ ] ● Running

VIEW APP [ ]

Buildpack: N/A

Overview Service (1) Route (1) Logs Tasks Trace Threads Settings

Configure Logging Levels [ ]

CLOSE

GIVE FEEDBACK

Pivotal. © 2018 Pivotal Software Inc. All rights reserved. Terms | Privacy

# Spring Boot To PCF Developer Workshop

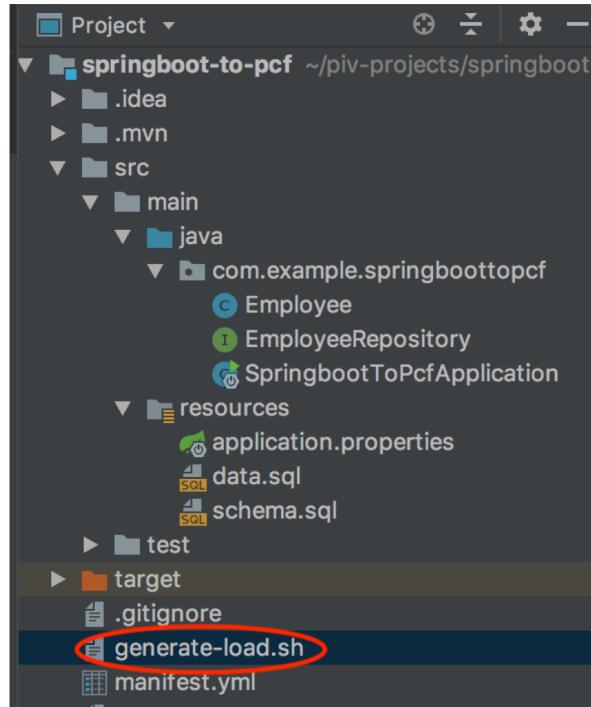
- Let's scale our application by adding a second instance either using Applications Manager UI or from the command line using
  - \$ cf scale employee-api -i 2

The screenshot shows the Pivotal Web Services Applications Manager interface. On the left, the main dashboard for the 'employee-api' application is visible, showing its status as 'Running' with 2 instances. The 'Overview' tab is selected. On the right, a modal dialog titled 'Scale app' is open, specifically for the 'web' service. In this dialog, the 'Instances' field is set to 2, and both 'Memory Limit' and 'Disk Limit' are set to 1 GB. Below the scaling form, there is a summary section titled 'Usage Total' which also shows 2 instances, 2.00 GB of memory, and 2.00 GB of disk. At the bottom right of the dialog is a large blue 'APPLY CHANGES' button.

# Spring Boot To PCF Developer Workshop

- Let's generate some load on our application so we can show how we monitor applications on PCF. In your project create a file called "generate-load.sh" with contents as shown below. If you don't have HTTPie installed use "curl" instead

```
#!/bin/bash
while true
do
echo "Press [CTRL+C] to stop.."
http http://$1/employees
sleep 2
done
```



# Spring Boot To PCF Developer Workshop

- Start your script by ensuring that it's executed and you pass in the URL of your employee-api micro service

```
$ cf apps | grep employee-api
```

```
employee-api      started      2/2      1G      1G      employee-api-accountable-hartebeest.cfapps.io
```

```
$ chmod +x generate-load.sh
```

```
$ ./generate-load.sh employee-api-accountable-hartebeest.cfapps.io
```

This will just call our “employee” RESTful endpoint over and over again until we use CNTRL-C. Leave it running as we need some application activity for the next part of this workshop

# Spring Boot To PCF Developer Workshop

Built in Metrics on PCF can be accessed from the application home page as shown to the right. Click on “View in PCF Metrics” link

- Metrics include:

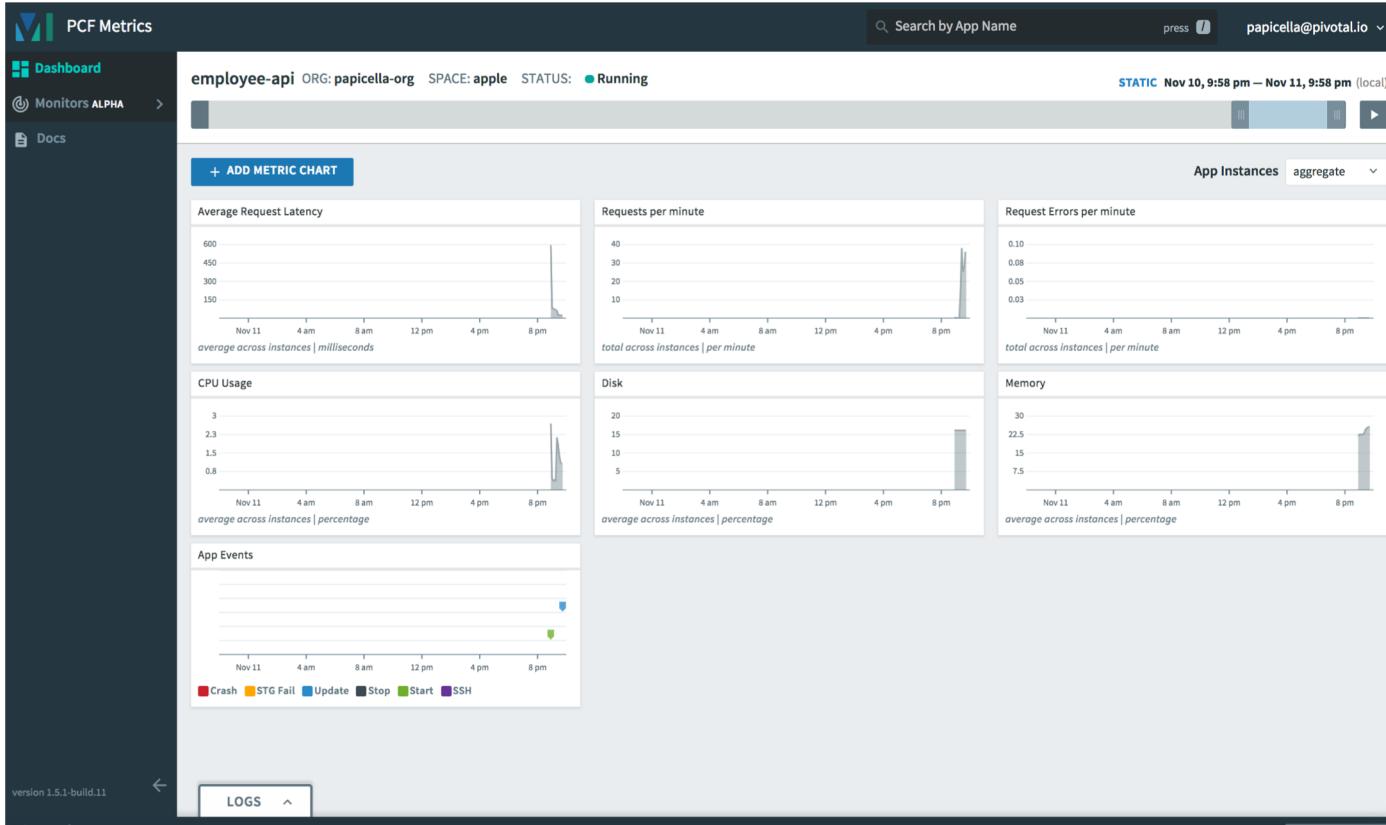
- HTTP requests and errors
- Avg. response latency
- CPU, memory and disk
- App events like start, stop, scale, update, and crash
- Application logs
- Request tracing

The screenshot shows the Pivotal Web Services application dashboard. On the left, there's a sidebar with 'Home' and 'Marketplace' options. The main area displays the 'employee-api' application under the path 'Home / papicella-org / apple / employee-api'. The application status is 'Running'. Below the status, there are tabs for 'Overview', 'Service (1)', 'Route (1)', 'Logs', 'Tasks', 'Trace', 'Threads', and 'Settings'. The 'Overview' tab is selected. It shows an 'Events' section with three entries: 'Started app' (papicella@pivotal.io 11/11/2018 at 08:54:34 PM), 'Mapped route to app' (papicella@pivotal.io 11/11/2018 at 08:54:14 PM), and 'Created app' (papicella@pivotal.io 11/11/2018 at 08:54:13 PM). To the right of the events, there's an 'App Summary' section with metrics: Instances / Allocated 1 / 1, Memory / Allocated 0.26 / 1.00 GB, and Disk / Allocated 0.16 / 1.00 GB. A 'View in PCF Metrics' button is highlighted with a red circle. Further down, there's a 'Processes and Instances' section for the 'web' process, showing 1 instance, 1 MB memory allocated, and 1 GB disk allocated. An 'Autoscaling' toggle is off. At the bottom, there's a table with columns: #, APP HEALTH, CPU, Memory, Disk, and Uptime, showing values for 0 instances.

All without any agent, and no application configuration!

# Spring Boot To PCF Developer Workshop

- We should see some activity on the graphs as shown below



# Spring Boot To PCF Developer Workshop

- Click on the “Logs” tab to verify the logs at any point in time on the graph

The screenshot shows the PCF Metrics dashboard for the `employee-api` application. The top navigation bar includes "PCF Metrics", "Dashboard", "Monitors ALPHA", and "Docs". A search bar allows searching by App Name, and the user is logged in as `papicella@pivotal.io`. The status of the application is "Running". The main area displays several metrics: "Average Request Latency" (a line chart peaking around 8:30 pm), "Requests per minute" (a line chart peaking around 9:30 pm), "Request Errors per minute" (a line chart near zero), and "CPU Usage", "Disk", and "Memory" (all showing low usage). Below these charts is a "LOGS" tab, which is currently selected. The log viewer shows eight log entries from November 11, 2018, at various times between 21:50 and 21:51. Each entry includes a timestamp, log level, application name, and a detailed log message. The log messages describe requests to the "/cloudfoundryapplication/health" and "/cloudfoundryapplication/info" endpoints, listing browser details like Mozilla/5.0 (Macintosh; Intel Mac OS X 10\_13\_6) AppleWebKit/537.36 (KHTML, like Gecko) Chrome/69.0.3497.100 Safari/537.36 and response times.

Timestamp	Log Level	Application	Message
Sun Nov 11 2018 [RTR/10]	INFO	employee-api-accountable-hartebeest.cfapps.io	[21:52:00.504] "OPTIONS /cloudfoundryapplication/health HTTP/1.1" 200 0 0 "-" "Mozilla/5.0 (Macintosh; Intel Mac OS X 10_13_6) AppleWebKit/537.36 (KHTML, like Gecko) Chrome/69.0.3497.100 Safari/537.36" "10.10.2.166:1916" "10.10.149.9:6104" x_forwarded_for:"110.175.56.52, 10.10.2.166" x_forwarded_proto:"https" vcap_request_id:"a59309f3-e283-4eb6-5ad8-b7122532c2c" response_time:0.001735227 app_id:"ee60fec7-65b7-49d9-a12e-e2abfcc9ebf5" app_index:"0" x_b3_traceid:"bad25080e2ee07ee" x_b3_spanid:"bad25080e2ee07ee" x_b3_parentspanid:""-"
Sun Nov 11 2018 [RTR/8]	INFO	employee-api-accountable-hartebeest.cfapps.io	[21:52:00.492] "OPTIONS /cloudfoundryapplication/info HTTP/1.1" 200 0 0 "-" "Mozilla/5.0 (Macintosh; Intel Mac OS X 10_13_6) AppleWebKit/537.36 (KHTML, like Gecko) Chrome/69.0.3497.100 Safari/537.36" "10.10.66.181:55476" "10.10.149.9:6104" x_forwarded_for:"110.175.56.52, 10.10.2.181" x_forwarded_proto:"https" vcap_request_id:"6c8e3172-79db-42d3-42c7-73b5a0e635c6" response_time:0.0024242981 app_id:"ee60fec7-65b7-49d9-a12e-e2abfcc9ebf5" app_index:"0" x_b3_traceid:"af604277b9edb7f7" x_b3_spanid:"af604277b9edb7f7" x_b3_parentspanid:""-"
Sun Nov 11 2018 [RTR/6]	INFO	employee-api-accountable-hartebeest.cfapps.io	[21:51:50.555] "OPTIONS /cloudfoundryapplication/mappings HTTP/1.1" 200 0 0 "-" "Mozilla/5.0 (Macintosh; Intel Mac OS X 10_13_6) AppleWebKit/537.36 (KHTML, like Gecko) Chrome/69.0.3497.100 Safari/537.36" "10.10.2.166:50849" "10.10.149.9:6104" x_forwarded_for:"110.175.56.52, 10.10.2.166" x_forwarded_proto:"https" vcap_request_id:"85856fe12-a220-4a60-4801-c57430d5026" response_time:0.001893196 app_id:"ee60fec7-65b7-49d9-a12e-e2abfcc9ebf5" app_index:"0" x_b3_traceid:"5d80bfbff80d94e0d" x_b3_spanid:"5d80bfbff80d94e0d" x_b3_parentspanid:""-"
Sun Nov 11 2018 [RTR/2]	INFO	employee-api-accountable-hartebeest.cfapps.io	[21:51:41.070] "GET /cloudfoundryapplication/health HTTP/1.1" 200 0 186 "-" "Mozilla/5.0 (Macintosh; Intel Mac OS X 10_13_6) AppleWebKit/537.36 (KHTML, like Gecko) Chrome/69.0.3497.100 Safari/537.36" "10.10.66.181:30430"

# Spring Boot To PCF Developer Workshop

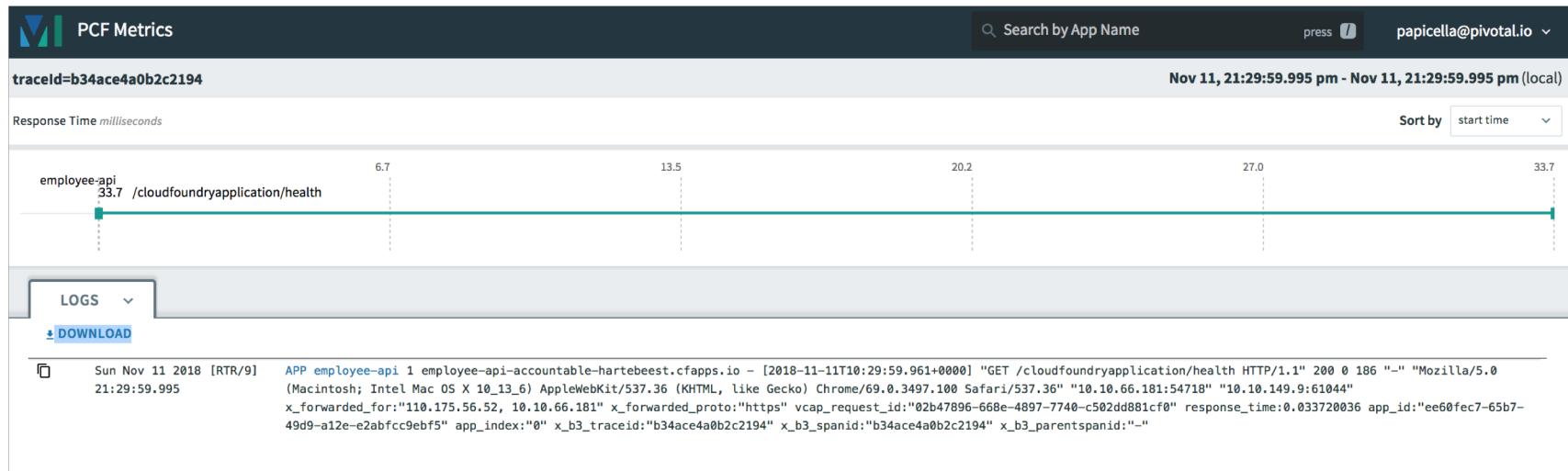
- We can even trace micro service calls using the “Trace Explorer” icon on a specific log entry

The screenshot shows the PCF Metrics interface for the 'employee-api' application. The top navigation bar includes 'PCF Metrics', a search bar ('Search by App Name'), and a user account ('papicella@pivotal.io'). The main dashboard displays several metrics charts: Average Request Latency, Requests per minute, Request Errors per minute, CPU Usage, Disk, and Memory. Below the charts, a log viewer is shown with a list of log entries. One specific log entry from 'Sun Nov 11 2018 [RTR/9] 21:54:20.340' is highlighted with a red circle around the 'View in Trace Explorer' button. The log entry details a request to /cloudfoundryapplication/health with various headers and parameters.

Time	Request ID	Method	URL	Headers	Parameters
Sun Nov 11 2018 [RTR/9] 21:54:20.340	b4da07ba-f728-4f15-5ba4-66fd02f3a970	OPTIONS	/cloudfoundryapplication/health	HTTP/1.1 Mozilla/5.0 (Macintosh; Intel Mac OS X 10_13_6) AppleWebKit/537.36 (KHTML, like Gecko) Chrome/69.0.3497.100 Safari/537.36	X-Forwarded-For: 10.10.149.9:61044 X-Forwarded-Proto: https Vcap-Request-ID: b4da07ba-f728-4f15-5ba4-66fd02f3a970 Response-Time: 0.002064084 App-ID: ee60fec7-65b7-49d9-a12e-e2abfcc9ebf5 App-Index: 0 X-B3-TraceID: c92b9d09fa1d810 X-B3-SpanID: 1c92b9d09fa1d810 X-B3-ParentSpanID: _
Sun Nov 11 2018 [RTR/3] 21:54:01.103	11d28669-1870-4688-4a04-cf875bc157de	GET	/cloudfoundryapplication/health	HTTP/1.1 Mozilla/5.0 (Macintosh; Intel Mac OS X 10_13_6) AppleWebKit/537.36 (KHTML, like Gecko) Chrome/69.0.3497.100 Safari/537.36	X-Forwarded-For: 10.10.148.170:61014 X-Forwarded-Proto: https Vcap-Request-ID: 11d28669-1870-4688-4a04-cf875bc157de Response-Time: 0.003345792 Ann-ID: ee60fec7-65b7-49d9-a12e-e2abfcc9ebf5 Ann-Index: 1 X-H3-TraceID: 100d7346a1471841 X-H3-SpanID: 100d7346a1471841

# Spring Boot To PCF Developer Workshop

- To use the “Trace Explorer” from Spring Boot you need to ensure your using the dependency “org.springframework.cloud:spring-cloud-starter-sleuth”. The Trace Explorer displays the apps and endpoints involved in completing a request, along with the corresponding logs
- For more information refer to this link
  - <https://docs.pivotal.io/pcf-metrics/1-5/using.html#trace>



# Spring Boot To PCF Developer Workshop

- You can view an application instance itself as shown below

The screenshot shows the PCF Metrics interface for the `employee-api` application. The application is running in the `papicella-org` organization and the `apple` space, with a status of "Running". The time range is set from `Nov 11, 9:23 pm` to `Nov 11, 9:30 pm (local)`. The interface includes several charts and a log viewer.

- App Instances:** `instance 0` (highlighted with a red oval).
- Average Request Latency:** A line chart showing latency over time.
- Requests per minute:** A line chart showing request volume.
- Request Errors per minute:** A line chart showing error rates.
- CPU Usage:** A chart showing CPU usage.
- Disk:** A chart showing disk usage.
- Memory:** A chart showing memory usage.
- Logs:** A table displaying log entries for four instances (RTR/9, RTR/4, RTR/8, RTR/21) on Nov 11, 2018, at 21:29:59.9xx. The logs show standard Spring Boot and Cloud Foundry application logs.

At the bottom, there are buttons for `ALL`, `SELECTED`, `DOWNLOAD`, `CLEAR`, and `APPLY`. The footer includes the version `1.5.1-build.11` and links for `GIVE FEEDBACK`.

Instance	Time	Log Message
RTR/9	Sun Nov 11 2018 [21:29:59.995]	employee-api-accountable-hartebeest.cfapps.io - [2018-11-11T10:29:59.936+0000] "GET /cloudfoundryapplication/health HTTP/1.1" 200 0 186 "-" "Mozilla/5.0 (Macintosh; Intel Mac OS X 10_13_6) AppleWebKit/537.36 (KHTML, like Gecko) Chrome/69.0.3497.100 Safari/537.36" "10.10.66.181:54718" "10.10.149.9:61044" x_forwarded_for:"10.175.56.52, 10.10.66.181" x_forwarded_proto:"https" vcap_request_id:"02b47896-668e-4897-7740-c502dd881cf0" response_time:0.033720036 app_id:"ee60fec7-65b7-49d9-a12e-e2abfc9c9ebf5" app_index:"0" x_b3_traceid:"b34ace4a0b2c2194" x_b3_spanid:"b34ace4a0b2c2194" x_b3_parentspanid:""-"
RTR/4	Sun Nov 11 2018 [21:29:59.968]	employee-api-accountable-hartebeest.cfapps.io - [2018-11-11T10:29:59.936+0000] "GET /cloudfoundryapplication/info HTTP/1.1" 200 0 2 "-" "Mozilla/5.0 (Macintosh; Intel Mac OS X 10_13_6) AppleWebKit/537.36 (KHTML, like Gecko) Chrome/69.0.3497.100 Safari/537.36" "10.10.66.181:6150" "10.10.149.9:61044" x_forwarded_for:"10.175.56.52, 10.10.66.181" x_forwarded_proto:"https" vcap_request_id:"40104a57-c0fe-4d8d-6a64-013943ab1583" response_time:0.031413667 app_id:"ee60fec7-65b7-49d9-a12e-e2abfc9c9ebf5" app_index:"0" x_b3_traceid:"f72a80c3a4d33950" x_b3_spanid:"f72a80c3a4d33950" x_b3_parentspanid:""-"
RTR/8	Sun Nov 11 2018 [21:29:58.137]	employee-api-accountable-hartebeest.cfapps.io - [2018-11-11T10:29:58.102+0000] "GET /cloudfoundryapplication/mappings HTTP/1.1" 200 0 82052 "-" "Mozilla/5.0 (Macintosh; Intel Mac OS X 10_13_6) AppleWebKit/537.36 (KHTML, like Gecko) Chrome/69.0.3497.100 Safari/537.36" "10.10.66.181:52176" "10.10.149.9:61044" x_forwarded_for:"10.175.56.52, 10.10.66.181" x_forwarded_proto:"https" vcap_request_id:"0f679caa-f0be-430d-6112-0fff87c9a715" response_time:0.03482345 app_id:"ee60fec7-65b7-49d9-a12e-e2abfc9c9ebf5" app_index:"0" x_b3_traceid:"069c0fd9e1082550" x_b3_spanid:"069c0fd9e1082550" x_b3_parentspanid:""-"
RTR/21	Sun Nov 11 2018 [21:29:58.119]	employee-api-accountable-hartebeest.cfapps.io - [2018-11-11T10:29:58.070+0000] "GET /cloudfoundryapplication/health HTTP/1.1" 200 0 186 "-" "Mozilla/5.0 (Macintosh; Intel Mac OS X 10_13_6) AppleWebKit/537.36 (KHTML, like Gecko) Chrome/69.0.3497.100 Safari/537.36" "10.10.66.181:27190" response_time:0.03482345 app_id:"ee60fec7-65b7-49d9-a12e-e2abfc9c9ebf5" app_index:"0" x_b3_traceid:"069c0fd9e1082550" x_b3_spanid:"069c0fd9e1082550" x_b3_parentspanid:""-"

# Spring Boot To PCF Developer Workshop

- Currently in ALPHA we will soon have the ability to add an “Event Monitor” or “Metric Monitor” as shown below. At this point we can use CNTRL-C to stop our script which is constantly hitting our micro service

The screenshot shows two views of the PCF Metrics interface. On the left is a smaller view of the main dashboard, and on the right is a detailed configuration screen for creating a new monitor.

**Main Dashboard View:**

- PCF Metrics logo
- Dashboard link
- Monitors ALPHA link (highlighted)
- Triggered link
- Manage link
- Docs link
- employee-api ORG: papicella-org SPACE: apple STATUS: Running
- TRIGGERED and MANAGE tabs (MANAGE is selected)
- No Monitors to manage
- + ADD EVENT MONITOR and + ADD METRIC MONITOR buttons

**Detailed Configuration View:**

- PCF Metrics logo
- Dashboard link
- Monitors ALPHA link (highlighted)
- Triggered link
- Manage link
- Docs link
- employee-api ORG: papicella-org SPACE: apple STATUS: Running
- TRIGGERED and MANAGE tabs (MANAGE is selected)
- New Event Monitor
- CRITERIA: For Q, App Crash (per minute), trigger when the number of events over the last minute is at or above the threshold.
- THRESHOLD:
  - Critical Threshold:  per minute
  - Warning Threshold: (optional)  per minute
- Don't trigger this monitor more than once every:  15m
- TITLE:
- CRITICAL NOTIFICATION:
  - Webhook URL: <https://hooks.slack.com/services/T00000000/B00000000/X00000000000000000000000>
  - POST body (JSON):

```
{ "text": "This is an example alert. <https://metrics.run.pivotal.io/apps/ee60fec7-65b7-49d9-a12e-e2abfcc9ebf5|Click here> to view on PCF Metrics." }
```
- WARNING NOTIFICATION:
  - Webhook URL: <https://hooks.slack.com/services/T00000000/B00000000/X00000000000000000000000>
  - POST body (JSON):

```
{ "text": "This is an example alert. <https://metrics.run.pivotal.io/apps/ee60fec7-65b7-49d9-a12e-e2abfcc9ebf5|Click here> to view on PCF Metrics." }
```

At the bottom of the configuration view, there are buttons for CANCEL and SAVE.

# Spring Boot To PCF Developer Workshop

- Let's add a scaling policy for our application so we can scale up / down as required. Start by clicking on the "Autoscaling" radio checkbox to add the "Auto Scaler" service and bind it to our application

The screenshot shows the Pivotal Web Services dashboard for the application `employee-api`. The application is listed under the `papicella-org` organization. The main interface includes a sidebar with links for Home, Marketplace, and a search bar. The central area displays the application's status as "Running" with three monitoring icons. Below this, there are tabs for Overview, Service (1), Route (1), Logs, Tasks, Trace, Threads, and Settings. The Overview tab is selected. On the left, a sidebar shows recent events: "Updated app", "Started app", "Mapped route to app", and "Created app". The main content area has sections for "Events" (with the last push at 09:43 PM on 11/11/18) and "App Summary". The App Summary section provides resource usage details: Instances / Allocated (2 / 2), Memory / Allocated (0.56 / 2.00 GB), and Disk / Allocated (0.32 / 2.00 GB). Below this is a "Processes and Instances" section for the "web" process, showing 2 instances, 1 GB memory allocated, and 1 GB disk allocated. A "SCALE" button is available for this process. A red circle highlights the "Autoscaling" toggle switch, which is currently off. Below the process table is a detailed table of instance metrics:

#	APP HEALTH	CPU	Memory	Disk	Uptime
> 0	Up	0%	296.64 MB	165.26 MB	1 hr 29 min
> 1	Up	0%	271.75 MB	165.26 MB	40 min

# Spring Boot To PCF Developer Workshop

- Now we will need to re-stage the application to pick up the service details and inject those into the application

The screenshot shows the Pivotal Web Services dashboard. On the left, there's a sidebar with 'Home' and 'Marketplace' buttons. The main area displays an application named 'employee-api'. At the top, a green banner indicates: 'Service instance "autoscale-apple" created and successfully bound to "employee-api"'. A red circle highlights the text 'TIP: Restage your app to ensure your service binding is available to your app.' Below the banner, the application name 'employee-api' is shown with a green power icon and three status icons: a square (grey), a circle (grey), and a refresh symbol (green). The status is 'Running'. To the right, there's a 'VIEW APP' button and a note 'Buildpack: N/A'. Below this, a navigation bar has tabs for 'Overview' (which is selected), 'Services (2)', 'Route (1)', 'Logs', 'Tasks', 'Trace', 'Threads', and 'Settings'. The 'Overview' tab shows an 'Events' section with four entries: 'Updated app', 'Started app', 'Mapped route to app', and 'Created app', all timestamped on 11/11/2018. To the right of the events, there's an 'App Summary' section with metrics: 'Instances / Allocated 2 / 2', 'Memory / Allocated 0.56 / 2.00 GB', and 'Disk / Allocated 0.32 / 2.00 GB'. Below the summary is a 'Processes and Instances' section for 'web' processes, showing 'Instances 2' and 'Memory Allocated 1 GB'. There's a 'SCALE' button and an 'Autoscaling' toggle switch. The table below lists two instances: one with 0 processes and another with 1 process, both marked as 'Up'.

#	APP HEALTH	CPU	Memory	Disk	Uptime
> 0	Up	1%	297.75 MB	165.26 MB	1 hr 31 min
> 1	Up	0%	272.57 MB	165.26 MB	43 min

# Spring Boot To PCF Developer Workshop

- Now click on “Manage Autoscaling” link as shown below

Pivotal Web Services press ⌘ papicella@pivotal.io ▾

Search apps, services, spaces, & orgs

Home Marketplace

App employee-api successfully started.

Home / papicella-org / apple / employee-api

APP employee-api ■ ⟳ ↻ • Running VIEW APP

Overview Services (2) Route (1) Logs Tasks Trace Threads Settings Buildpack: N/A

Events Last Push: 10:29 PM 11/11/18 App Summary

- Started app papicella@pivotal.io 11/11/2018 at 10:29:20 PM
- Stopped app papicella@pivotal.io 11/11/2018 at 10:29:18 PM
- Updated app papicella@pivotal.io 11/11/2018 at 09:43:46 PM
- Started app papicella@pivotal.io 11/11/2018 at 08:54:34 PM
- Mapped route to app papicella@pivotal.io 11/11/2018 at 08:54:14 PM
- Created app papicella@pivotal.io 11/11/2018 at 08:54:13 PM

Instances / Allocated 2 / 2 Memory / Allocated 0.47 / 2.00 GB Disk / Allocated 0.32 / 2.00 GB

Processes and Instances View in PCF Metrics

web	SCALE
Instances 2 Memory Allocated 1 GB Disk Allocated 1 GB	<span>Autoscaling</span>
# APP HEALTH CPU Memory Disk Uptime	<span>Manage Autoscaling</span>
> 0 Up 1% 239.74 MB 165.26 MB 2 min	⋮
> 1 Up 3% 245.89 MB 165.26 MB 2 min	⋮

# Spring Boot To PCF Developer Workshop

- Set the maximum instance limit to “4” and then “Apply Changes”. Once that is done click on the “Edit” link to add a scaling rule

Manage Autoscaling

Download Autoscaler CLI ↗

INSTANCE LIMITS  
Currently in use: 2

Minimum  Maximum

Manage Autoscaling

Instance Limits updated successfully

Download Autoscaler CLI ↗

INSTANCE LIMITS  
Currently in use: 2

Minimum  Maximum

SCALING RULES

No rules set

**EDIT**

- Full example exists here

- <http://theblasfrompas.blogspot.com/2018/04/usingverifying-autoscale-service-from.html>

# Spring Boot To PCF Developer Workshop

- Add a Scaling rule as shown below

< Edit Scaling Rules

Apps scale by 1 instance per event. Apps will scale up when any metric maximum is met and scale down only when all metric minimums are met.

HTTP Throughput

Scale down if less than:  /s

Scale up if more than:  /s

**ADD RULE**

**CANCEL** **SAVE**

Manage Autoscaling

Download Autoscaler CLI 

**INSTANCE LIMITS**  
Currently in use: 2

Minimum  Maximum  **APPLY CHANGES**

**SCALING RULES** **EDIT**

Metric	Low	High
HTTP Throughput	2/s	5/s

**SCHEDULED LIMITS** **EDIT**

No Scheduled Limit Changes

# Spring Boot To PCF Developer Workshop

- The Auto Scaler has a CF CLI Plugin which you can find out more details about below as well as install it as shown in the link below.
  - <https://docs.run.pivotal.io/appsman-services/autoscaler/using-autoscaler-cli.html#install>

```
pasapicella@pas-macbook:~$ cf plugins
Listing installed plugins...

  plugin      version   command name          command help
App Autoscaler 1.0.66   autoscaling-apps    Displays apps bound to the autoscaler
App Autoscaler 1.0.66   autoscaling-events  Displays previous autoscaling events for the app
App Autoscaler 1.0.66   autoscaling-rules   Displays rules for an autoscaled app
App Autoscaler 1.0.66   configure-autoscaling Configures autoscaling using a manifest file
App Autoscaler 1.0.66   create-autoscaling-rule Create rule for an autoscaled app
App Autoscaler 1.0.66   delete-autoscaling-rule Delete rule for an autoscaled app
App Autoscaler 1.0.66   delete-autoscaling-rules Delete all rules for an autoscaled app
App Autoscaler 1.0.66   disable-autoscaling   Disables autoscaling for the app
App Autoscaler 1.0.66   enable-autoscaling   Enables autoscaling for the app
App Autoscaler 1.0.66   update-autoscaling-limits Updates autoscaling instance limits for the app
```

# Spring Boot To PCF Developer Workshop

- To be able to test our Auto Scaling service we will need to generate some load to generate enough HTTP Throughput to force the Auto Scaler service to create new instances. Few ways to do that here is two of them
  - Use Apache JMeter (<https://jmeter.apache.org/>)
  - Use “ab” from your MacBook pro or Linux file system
    - \$ ab -n 10000 -c 25 http://{URL}/employees

```
pasapicella@pas-macbook:~$ ab -n 10000 -c 25 http://employee-api-accountable-hartebeest.cfapps.io/employees
This is ApacheBench, Version 2.3 <$Revision: 1826891 $>
Copyright 1996 Adam Twiss, Zeus Technology Ltd, http://www.zeustech.net/
Licensed to The Apache Software Foundation, http://www.apache.org/

Benchmarking employee-api-accountable-hartebeest.cfapps.io (be patient)
Completed 1000 requests
Completed 2000 requests
Completed 3000 requests
Completed 4000 requests
Completed 5000 requests
```

# Spring Boot To PCF Developer Workshop

- It won't take long before we see the Auto Scaler service kick in as shown below

The screenshot shows the Pivotal Web Services dashboard for the `employee-api` application. The app status is "Starting". The "Events" sidebar lists several recent events, including updates and starts. The "App Summary" section shows 3/4 instances allocated, with memory and disk usage details. The "Processes and Instances" section displays the current state of the application processes. A red box highlights the row for instance 2, which is labeled "STARTING...".

#	APP HEALTH	CPU	Memory	Disk	Uptime
> 0	Up	13%	275.16 MB	165.26 MB	27 min
> 1	Up	13%	278.56 MB	165.26 MB	27 min
> 2	Up	115%	245.33 MB	165.26 MB	0 min
> 3	STARTING...				

# Spring Boot To PCF Developer Workshop

- In fact the Auto Scaler service even emails you!!!

 Pivotal CF

A scaling event has been reported for application **employee-api**.

**Event Message:** Scaled up from 3 to 4 instances. Current HTTP Throughput of 10.46/s/instance is above upper threshold of 5.00/s /instance.

**Event Time:** 2018-11-11 11:57:08 +0000 UTC

The autoscaler will continue to scale this app within the current instance limits:

**Minimum Instances:** 2

**Maximum Instances:** 4

Manage your [notification preferences](#) or [unsubscribe](#)

Pivotal CF, and the Pivotal CF logo are registered trademarks or trademarks of Pivotal Software, Inc. in the United States and other countries. All other trademarks used herein are the property of their respective owners.

© 2014 Pivotal Software, Inc. All rights reserved. Published in the USA.

# Spring Boot To PCF Developer Workshop

- In fact the Auto Scaler service even emails you when it starts to scale back down as well

Pivotal CF

A scaling event has been reported for application **employee-api**.

**Event Message:** Scaled down from 4 to 3 instances. All metrics are currently below minimum thresholds.

**Event Time:** 2018-11-11 12:01:46 +0000 UTC

The autoscaler will continue to scale this app within the current instance limits:

**Minimum Instances:** 2  
**Maximum Instances:** 4

Manage your [notification preferences](#) or [unsubscribe](#)

Pivotal CF, and the Pivotal CF logo are registered trademarks or trademarks of Pivotal Software, Inc. in the United States and other countries. All other trademarks used herein are the property of their respective owners.

© 2014 Pivotal Software, Inc. All rights reserved. Published in the USA.

# Spring Boot To PCF Developer Workshop

- Finally back to normal after all our load test has ended!!!

The screenshot shows the Pivotal Web Services (PCF) dashboard for the `employee-api` application. The application is listed under the `papicella-org` organization. The main interface includes a sidebar with links for Home and Marketplace, a search bar at the top, and a navigation bar with tabs for Overview, Services (2), Route (1), Logs, Tasks, Trace, Threads, and Settings.

The **Events** section shows several log entries:

- Updated app (11/11/2018 at 11:02:21 PM)
- Updated app (11/11/2018 at 11:01:46 PM)
- Updated app (11/11/2018 at 10:57:08 PM)
- Updated app (11/11/2018 at 10:56:32 PM)
- Started app (papicella@pivotal.io 11/11/2018 at 10:29:20 PM)
- Stopped app (papicella@pivotal.io 11/11/2018 at 10:29:18 PM)

The **App Summary** section provides resource usage details:

Instances / Allocated	Memory / Allocated	Disk / Allocated
2 / 2	0.55 / 2.00 GB	0.32 / 2.00 GB

The **Processes and Instances** section shows the `web` process with two instances:

Instances	Memory Allocated	Disk Allocated	SCALE
2	1 GB	1 GB	SCALE

Autoscaling settings are shown, with a button to "Manage Autoscaling".

A table displays the current state of the application instances:

#	APP HEALTH	CPU	Memory	Disk	Uptime
> 0	Up	0%	282.61 MB	165.26 MB	34 min
> 1	Up	0%	279 MB	165.26 MB	34 min

The last two rows of the table are highlighted with a red box.

# Spring Boot To PCF Developer Workshop

- From the applications Manager UI you will see we have two services bound to the “employee-api” application as the Auto Scaler service is a service itself

The screenshot shows the Pivotal Web Services Applications Manager UI. The top navigation bar includes the Pivotal logo, a search bar, and user information for 'press' and 'papicella@pivotal.io'. The main area displays the 'employee-api' application under the 'Home / papicella-org / apple / employee-api' path. The application status is 'Running'. Below the app name are several control icons: a power button, a refresh, and a deployment. A 'VIEW APP' link is also present. The navigation tabs include 'Overview', 'Services (2)', 'Route (1)', 'Logs', 'Tasks', 'Trace', 'Threads', and 'Settings'. The 'Services (2)' tab is selected. The 'Bound Services' section lists two services: 'App Autoscaler free - Standard' (instance name: 'autoscale-apple') and 'MySQL for Pivotal Cloud Foundry free - 100 MB' (instance name: 'pas-mysql'). Each service row has a 'BIND SERVICE' and 'NEW SERVICE' button, and a three-dot menu icon.

Service	Instance Name	Binding Name
App Autoscaler free - Standard	autoscale-apple	
MySQL for Pivotal Cloud Foundry free - 100 MB	pas-mysql	

# Spring Boot To PCF Developer Workshop

- Had trouble along the way? The whole source code is on GitHub
  - <https://github.com/papicella/springboot-to-pcf>

The screenshot shows the GitHub repository page for 'papicella / springboot-to-pcf'. The repository has 1 commit, 1 branch, 0 releases, and 1 contributor. The latest commit was made 2 minutes ago. The repository contains files such as .mvn/wrapper, src, generate-load.sh, manifest.yml, mvnw, mvnw.cmd, and pom.xml, all of which were updated 2 minutes ago. A button to 'Add a README' is visible at the bottom.

No description, website, or topics provided.

Manage topics

1 commit 1 branch 0 releases 1 contributor

Branch: master New pull request Create new file Upload files Find file Clone or download

papicella Spring Boot to PCF Workshop

File	Description	Updated
.mvn/wrapper	Spring Boot to PCF Workshop	2 minutes ago
src	Spring Boot to PCF Workshop	2 minutes ago
generate-load.sh	Spring Boot to PCF Workshop	2 minutes ago
manifest.yml	Spring Boot to PCF Workshop	2 minutes ago
mvnw	Spring Boot to PCF Workshop	2 minutes ago
mvnw.cmd	Spring Boot to PCF Workshop	2 minutes ago
pom.xml	Spring Boot to PCF Workshop	2 minutes ago

Help people interested in this repository understand your project by adding a README.

Add a README

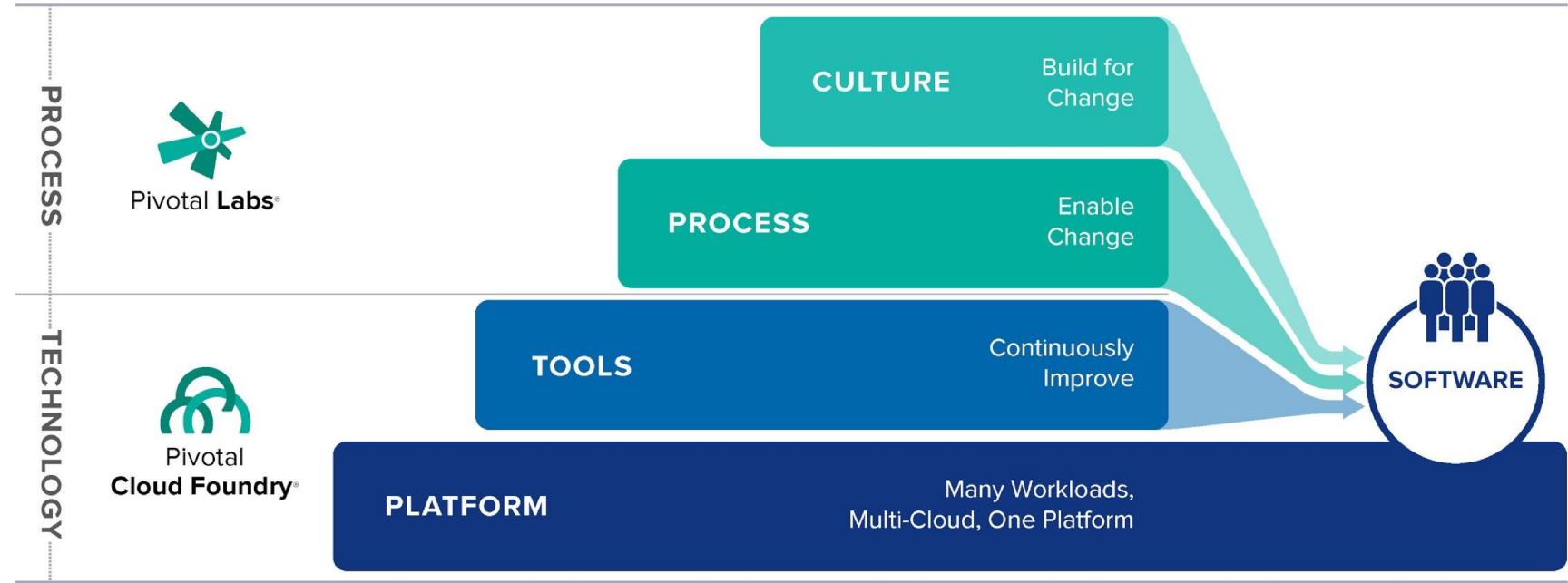
© 2018 GitHub, Inc. Terms Privacy Security Status Help Contact GitHub Pricing API Training Blog About

**Spring Boot to PCF Developer Workshop**

---

# Summary

# Pivotal Enables Cloud-Native Transformation



Spring Boot to PCF Developer Workshop

---

# Questions?





# Pivotal®

---

## Transforming How The World Builds Software