**DEPLOYING SPRING APPS IN PCF**

Use sample app in GITgit clone <https://github.com/cloudfoundry-samples/pong_matcher_spring> to clone the pong\_matcher\_spring app.

**PREREQUISITES:**

* A Spring app that runs locally on your workstation
* Intermediate to advanced Spring knowledge
* The Cloud Foundry Line Interface (cf CLI)
* JDK 1.6, 1.7 or 1.8 for JAVA 6, 7, or 8 configured on workstation

**STEP 1. Declare App Dependencies**

Be sure to declare all the dependency tasks for your app in the build script of your chosen build tool.

The pom.xml file contains dependencies for the pong\_matcher\_spring sample app, as the example below shows.

<dependencies>

<dependency>

<groupId>mysql</groupId>

<artifactId>mysql-connector-java</artifactId>

</dependency>

<dependency>

<groupId>org.flywaydb</groupId>

<artifactId>flyway-core</artifactId>

</dependency>

<dependency>

<groupId>org.springframework.boot</groupId>

<artifactId>spring-boot-starter-data-jpa</artifactId>

</dependency>

<dependency>

<groupId>org.springframework.boot</groupId>

<artifactId>spring-boot-starter-test</artifactId>

</dependency>

<dependency>

<groupId>org.springframework.boot</groupId>

<artifactId>spring-boot-starter-web</artifactId>

</dependency>

<dependency>

<groupId>com.h2database</groupId>

<artifactId>h2</artifactId>

<scope>test</scope>

</dependency>

<dependency>

<groupId>com.jayway.jsonpath</groupId>

<artifactId>json-path</artifactId>

<scope>test</scope>

</dependency>

</dependencies>

**STEP 2. Allocate Sufficient Memory**

Use the cf push –m command to specify the amount of memory that should be allocated to the application. Memory allocated this way is done in preset amounts of 64M, 128M, 256M, 512M, 1G, or 2G.

$ cf push <APP-NAME> –m 128M

***Sample App Step*** *You can skip this step. The Cloud Foundry Java buildpack uses settings declared in the sample app to allocate 1 GB of memory to the app.*

**STEP 3. Provide JDBC Driver**

The Java buildpack does not bundle a JDBC driver with your application. If your application accesses a SQL RDBMS, you must do the following:

* Include the appropriate driver in your application.
* Create dependency task for the driver in the build script for your build tool or IDE.

***Sample App Step*** *You can skip this step. In the pong\_matcher\_spring sample app, the src/main/resources/application.yml file declares the JDBC driver, and the pom.xml file includes the JDBC driver as a dependency.*

**STEP 4. Configure Service Connections for a Spring App**

PWS provide extensive support for creating and binding a Spring application to services such as MySQL, PostgreSQL, MongoDB, Redis, and RabbitMQ.

***Sample App Step: Create a Service Instance*** *Run cf create-service cleardb spark mysql. This creates a service instance named mysql that uses the cleardb service and the spark plan*

$ cf create-service cleardb spark mysql

***Sample App Step: Bind a Service Instance*** *You can skip this step because the service instance is already bound. Open the manifest.yml file in a text editor to view the bound service instance information. Locate the file in the app root directory and search for the services sub-block in the applications block, as the example below shows.*

**STEP 5. Configure the Deployment Manifest**

You can specify deployment option in a manifest file *manifest.yml* that the *cf push* command uses when deploying your app.

***Sample App Step*** *You can skip this step. The manifest.yml file for the pong\_matcher\_spring sample app does not require any additional configuration to deploy the app.*

**STEP 6. Log in and Target API Endpoint**

Run *cf login –a API-ENDPOINT* , enter your login credentials, and select a space and org. The API endpoint is *api.run.pivotal.io*.

***Sample App Step*** *You must do this step to run the sample app.*

$ cf login –a <https://api.run.pivotal.io>

**STEP 7. Deploy Your Application**

From the root directory of your application, run *cf push APP-NAME –p PATH-TO-FILE.war* to deploy your application.

***Note****: Most Spring apps include an artifact, such as a .jar, .war, or .zip file. You must include the path to this file in the cf push command using the -p option if you do not declare the path in the applications block of the manifest file. The example shows how to specify a path to the .war file for a Spring app. Refer to the*[*Tips for Java Developers*](https://docs.run.pivotal.io/buildpacks/java/java-tips.html)*topic for CLI examples for specific build tools, frameworks, and languages that create an app with an artifact.*

*cf push APP-NAME* creates a URL route to your application in the form HOST.DOMAIN, where HOST Is your APP-NAME and DOMAIN is specified by your administrator. Your DOMAIN is *cfapps.io.* For example: *cf push my-app* creates the URL *my-app.cfapps.io.*

The URL for your app must be unique from other apps that PWS hosts of the push will fail. Use the following options to help create unique URL:

* *-n* to assign a different HOST name for the app.
* *--random-route* to create a URL that includes the app name and random words.
* *cf help push* to view other options for this command

If you want to view log activity while the app deploys, launch a new terminal window and run *cf logs APP-NAME.*

Once your app deploys, browse to your app URL. Search for the urls field in the App started block in the output of the *cf push* command. Use the URL to access your app online.

***Sample App Step*** *1. Run brew install maven.   
2. Change to the app directory, and run mvn package to build the app.   
3. Run cf push pong\_matcher\_spring -n HOSTNAME to push the app.   
  
Example: cf push pong\_matcher\_spring -n my-spring-app*