Lab 6 - Binding to Cloud Foundry Services

The Spring Music application was designed to illustrate the ease with which various types of data services can be bound to and utilized by Spring applications running on Cloud Foundry. In this lab, we'll be binding the application to a MySql database.

Cloud Foundry services are managed through two primary types of operations:

Create/Delete

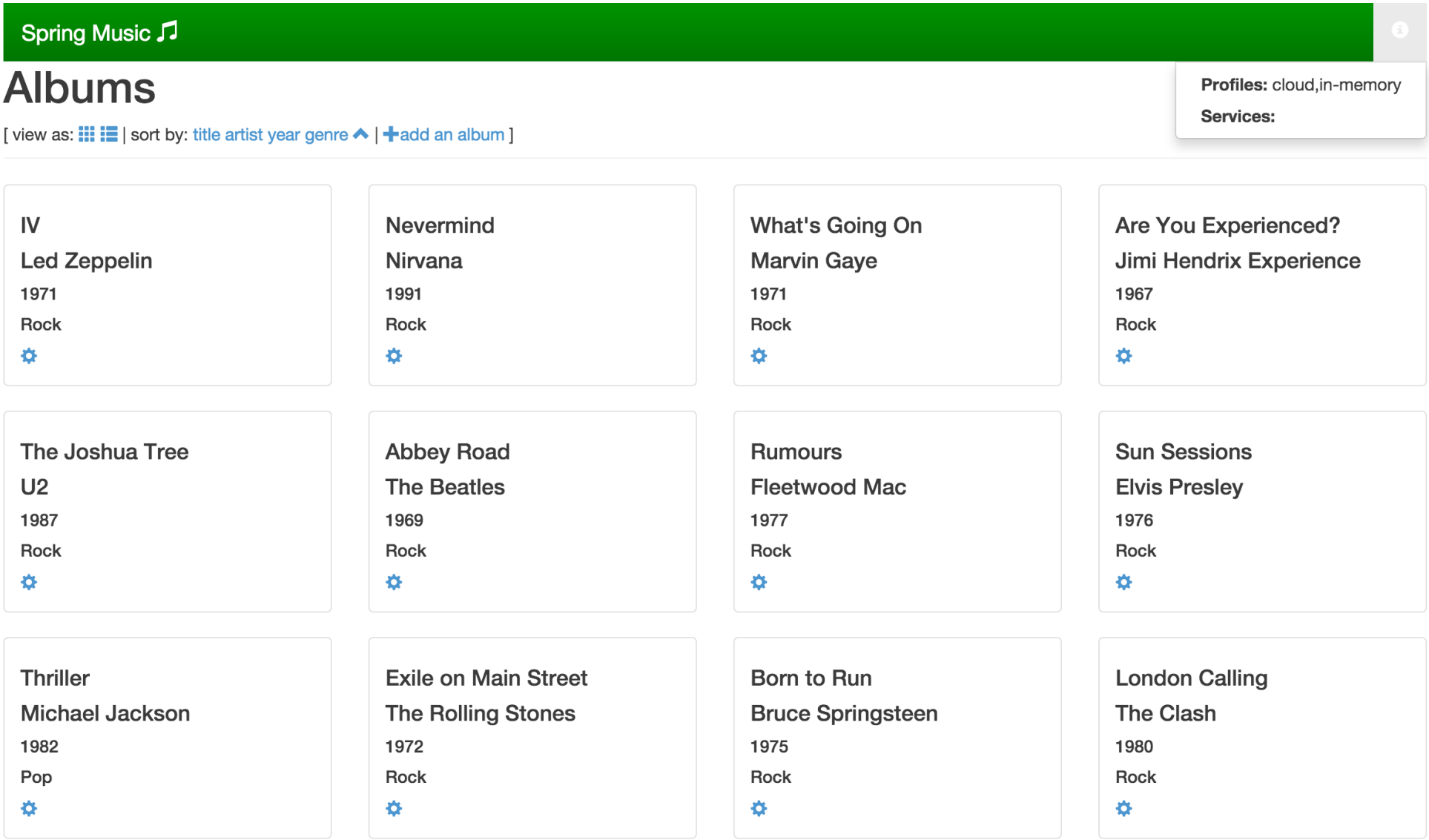
> *These operations create or delete instances of a service. For a database this could mean creating/deleting a schema in an existing multitenant cluster or creating/deleting a dedicated database cluster.*

Bind/Unbind

> *These operations create or delete unique credential sets for an existing service instance that can then be injected into the environment of an application instance.*

A Bit of Review

Your instance of *Spring Music* should still be running from the end of Lab 5. Visit the application in your browser by hitting the route that was generated by the CLI:



The information dialog in the top right-hand corner indicates that we're currently running with an in-memory database, and that we're not bound to any services. Let's change that.

The Services Marketplace

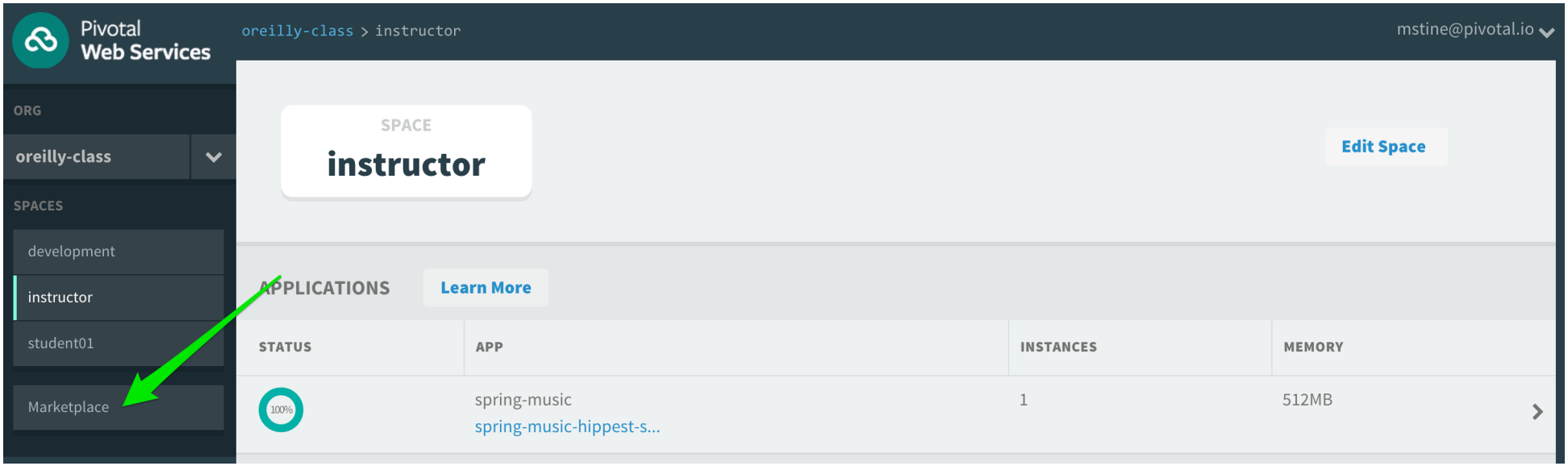
There are two ways to discover what services are available on Pivotal Web Services. The first is available on any instance of Cloud Foundry: the CLI. Just type:

```
$ cf marketplace
```

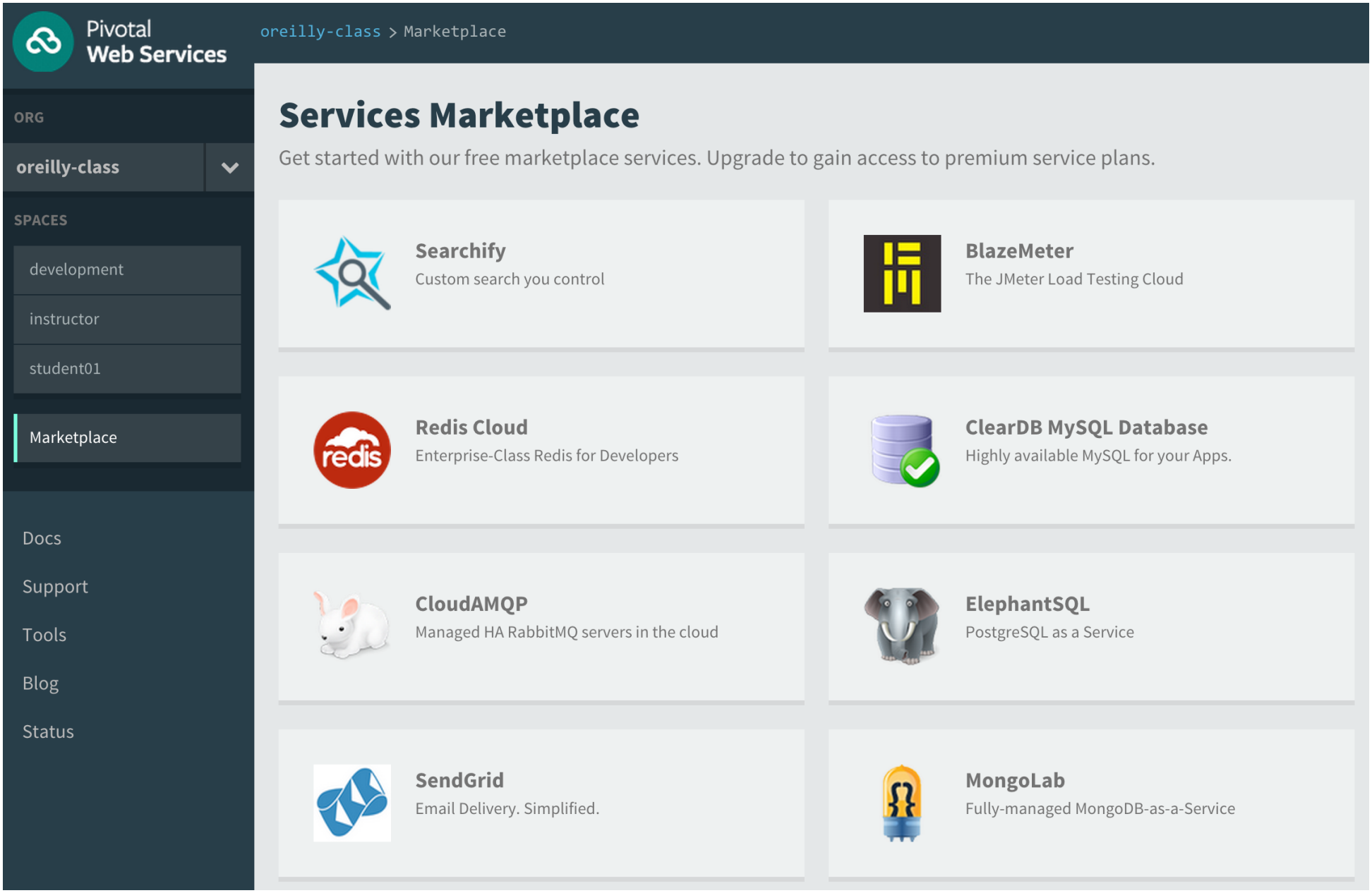
and you'll get a list of services, their available plans, and descriptions.

The second way is specific to Pivotal Cloud Foundry's Application Manager UI. If you haven't already, login to it by visiting {CF-CONSOLE-URL}.

Click on the "Marketplace" link: *The Screenshots below are from Pivotal Web Services (public managed Pivotal CF). If you are using Pivotal Cloud Foundry, you will see something similar.*



and you'll see the same service/plan/description listing in the browser:



Creating and Binding to a Service Instance

1. Let's begin by creating a MySql instance provided by Pivotal.

From the CLI, let's *create* a p-mysql service instance:

+

```
$ cf create-service p-mysql 100mb-dev spring-music-db
Creating service spring-music-db in org ACME / space instructor as mstine@pivotal.io...
OK
```

1. Next we'll *bind* the newly created instance to our `spring-music` application:

   ```
   $ cf bs spring-music spring-music-db
   Binding service spring-music-db to app spring-music in org ACME / space instructor as mstine@pivotal.io...
   OK
   TIP: Use 'cf restage' to ensure your env variable changes take effect
   ```

2. Notice the admonition to `Use 'cf restage' to ensure your env variable changes take effect`. Let's take a look at the environment variables for our application to see what's been done. We can do this by typing:

   ```
   $ cf env spring-music
   ```

   The subset of the output we're interested in is located near the very top, titled `System-Provided`:

```
System-Provided:
{
 "VCAP_SERVICES": { (1)
  "p-mysql": [
   {
    "credentials": {
     "hostname": "10.68.50.61",
     "jdbcUrl": "jdbc:mysql://10.68.50.61:3306/cf_ddbe0221_d1fc_478f_b693_69279b2de702?
user=mWsYAON34zbaQPAA\u0026password=CZImCZ9iSNU7HfP2",
     "name": "cf_ddbe0221_d1fc_478f_b693_69279b2de702",
     "password": "CZImCZ9iSNU7HfP2",
     "port": 3306,
     "uri":
"mysql://mWsYAON34zbaQPAA:CZImCZ9iSNU7HfP2@10.68.50.61:3306/cf_ddbe0221_d1fc_478f_b693_69279b2de702?
reconnect=true", (2)
     "username": "mWsYAON34zbaQPAA"
    },
    "label": "p-mysql",
    "name": "spring-music-db",
    "plan": "100mb-dev",
    "tags": [
     "mysql",
     "relational"
    ]
   }
  ]
 }
}

{
 "VCAP_APPLICATION": {
  "application_name": "spring-music",
  "application_uris": [
   "spring-music-unbragging-pourparler.pcf2.fe.gopivotal.com"
  ],
```

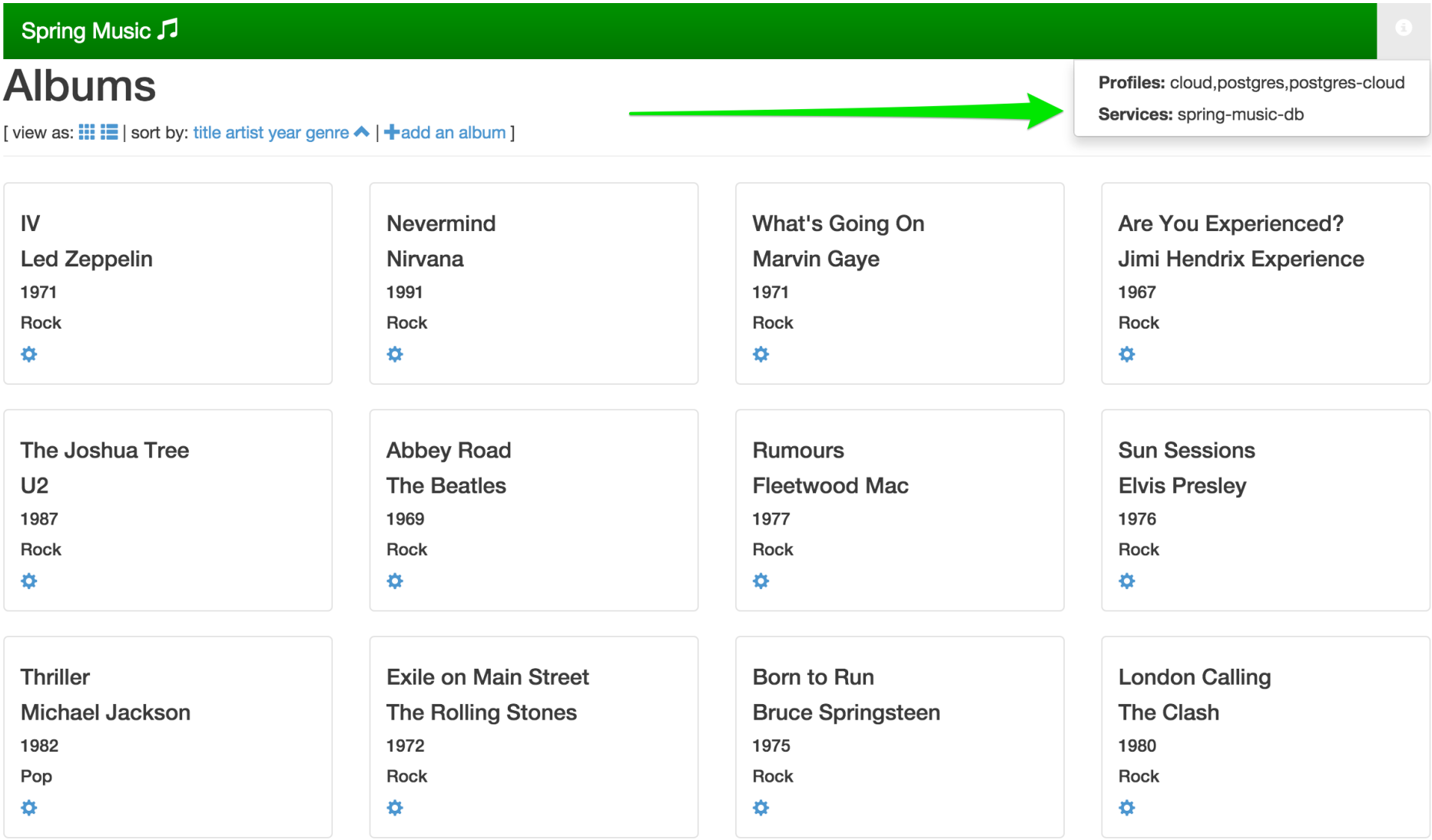
```
   "application_version": "e3fa423d-69ad-41cb-bd17-0e88427dfe4c",
   "limits": {
    "disk": 1024,
    "fds": 16384,
    "mem": 512
   },
   "name": "spring-music",
   "space_id": "080ba4e6-bafc-482b-9a98-42f612eef736",
   "space_name": "jfullam",
   "uris": [
    "spring-music-unbragging-pourparler.pcf2.fe.gopivotal.com"
   ],
   "users": null,
   "version": "e3fa423d-69ad-41cb-bd17-0e88427dfe4c"
  }
 }
```

1. `VCAP_SERVICES` is a special Cloud Foundry environment variable that contains a JSON document containing all of the information for any services bound to an application.
2. Notice here the unique URI for this instance of MySql that `spring-music` has been bound to.

3. Now let's *restage* the application, which cycles our application back through the staging/buildpack process before redeploying the application.[1]

```
$ cf restage spring-music
```

Once the application is running again, revisit or refresh the browser tab where you have the *Spring Music* application loaded:



As you can see from the information dialog, the application is now utilizing a MySql database via the `spring-music-db` service.

4. **(OPTIONAL STEPS)** If you have a MySql GUI tool handy and you are using a lab environment that has the necessary network access (ask your instructor), you can inspect the music database created. Otherwise, your instructor will demo via a Pivotal VPN connection.

5. In your DB tool, create a new server connection and populate the properties with values from the URI in your `VCAP_SERVICES` environment variable (remember `cf env spring-muisc`!):

---

1. In this case, we could accomplish the same goal by only *restarting* the application via `cf restart spring-music`. A *restage* is generally recommended because Cloud Foundry buildpacks also have access to injected environment variables and can install or configure things differently based on their values.