(../..)

Pre-Requisite

Clone the repository:

```
git clone https://github.com/platform-acceleration-lab/apps-platform-a
cceleration-removing-instance-specific-state-code.git
```

Install Nginx:

```
brew install nginx
```

We will be using Nginx as our local load-balancer.

Install Redis, which we will use as our session store. You can use brew to install Redis on MacOS.

```
brew install redis
```

On Windows we recommend either these instructions (https://github.com/ServiceStack/redis-windows#running-the-latest-version-of-redis-with-vagrant) or using Chocolatey:

```
choco install redis-64
```

Getting Started

Run the application with the following command:

```
mvn -Dmaven.tomcat.port=8080 -DinstanceNumber=1 tomcat7:run-war
```

Browse to http://localhost:8080 (http://localhost:8080) and sign in with the following credentials:

- Username: `palstudent@pivotal.io`
- Password: `super-secret`

This should redirect you to the `profile` page.

Now start a second instance in a second terminal:

```
mvn -Dmaven.tomcat.port=9090 -DinstanceNumber=2 tomcat7:run-war
```

Start a load-balancer using our provided `nginx.conf` in a third terminal:

```
./start-load-balancer.sh
```

This starts a new server on port `9000`. This server will randomly hit the other server on `8080` and on `9090`.

Sign in to the new server http://localhost:9000 (http://localhost:9000).

Does the profile page work? If it seems like it does, refresh the page several times. At some point, it should fail to maintain the session. This is because the server is stateful and each instance has its own state. We need to fix this so that we can scale the application horizontally.

Redis session management

We are going to take advantage of Spring's automated session management with Redis.

- Add dependencies to `pom.xml`

```
<dependency>
    <groupId>org.springframework</groupId>
    <artifactId>spring-web</artifactId>
    <version>4.2.5.RELEASE</version>
</dependency>
<dependency>
    <groupId>org.springframework.session</groupId>
    <artifactId>spring-session-data-redis</artifactId>
    <version>1.2.2.RELEASE</version>
</dependency>
```

- Create a `SpringInitializer` class:

```
public class SpringInitializer implements WebApplicationInitializer {
    @Override
    public void onStartup(ServletContext servletContext) throws ServletException {
        servletContext.setInitParameter("contextClass", "org.springframework.web.context.support.AnnotationConfigWebApplicationContext");
        servletContext.setInitParameter("contextConfigLocation", "io.pivotal.pal");
        servletContext.addListener(ContextLoaderListener.class);

        servletContext.addFilter("springSessionRepositoryFilter", DelegatingFilterProxy.class)
                .addMappingForUrlPatterns(null, false, "/*");
    }
}
```

- Create a `RedisConfiguration` class:

```
@Configuration
@EnableRedisHttpSession
public class RedisConfiguration {
    @Bean RedisConnectionFactory redisConnectionFactory() {
        return new JedisConnectionFactory();
    }

    @Bean
    public RedisTemplate<String, Object> redisTemplate(RedisConnec
tionFactory redisConnectionFactory) {
        RedisTemplate<String, Object> template = new RedisTemplate
<>();
        template.setConnectionFactory(redisConnectionFactory);
        template.setDefaultSerializer(new StringRedisSerializer())
;
        return template;
    }
}
```

- Add the following line in the `<body>` of `profile.jsp`

```
Instance number: <%= System.getProperty("instanceNumber") %>
```

- Run `Redis` and the load-balancer in one terminal:

```
brew services start redis
```

- Run an instance in a second terminal:

```
mvn -Dmaven.tomcat.port=8080 -DinstanceNumber=1 tomcat7:run-war
```

- Run another instance in a third terminal:

```
mvn -Dmaven.tomcat.port=9090 -DinstanceNumber=2 tomcat7:run-war
```

- Sign in to the website at http://localhost:9000 (http://localhost:9000). Refresh the profile page several times. Make sure you can see your profile on `Instance number: 1` and `Instance number: 2`.

Next we will deploy to Pivotal Cloud Foundry.

Set up for PCF

- Add the Spring Cloud service connector dependency:

  ```
  <dependency>
      <groupId>org.springframework.cloud</groupId>
      <artifactId>spring-cloud-spring-service-connector</artifactId>
      <version>1.2.3.RELEASE</version>
  </dependency>
  ```

- Disable the `Session Destroyed Event` on PCF. Add the following bean to your `RedisConfiguration` class:

  ```
  @Bean
  public static ConfigureRedisAction configureRedisAction() {
      return ConfigureRedisAction.NO_OP;
  }
  ```

  You will find more explanations on why this is needed in this Spring Session Github issue (https://github.com/spring-projects/spring-session/issues/124)

- Deploy the application, do not start it yet

  ```
  mvn clean package
  cf push remove-session-state -p target/remove-session-state-lab.wa
  r --random-route --no-start
  ```

- Create a redis service instance, and bind it to your application.

  ```
  cf create-service p-redis shared-vm my-redis
  cf bind-service remove-session-state my-redis
  ```

- Start the application

  ```
  cf start remove-session-state
  ```

Once the deployment is done, visit the app at the generated url, and ensure it is working as expected.

Assignment

Once you are done with the lab and the application is deployed and working on PCF, you can submit the assignment using the `submitReplatformingRemoveInstanceSpecificState` gradle task. It requires you to provide the `sessionStateUrl` project property. For example:

```
cd ~/workspace/assignment-submission
./gradlew submitReplatformingRemoveInstanceSpecificState -PsessionStat
eUrl=http://session-state-app.cfapps.io
```