EXERCISE 12 – MIGRATE SAP CLOUD SDK BASED APPLICATIONS FROM SAP CLOUD PLATFORM NEO ENVIRONMENT TO CLOUD FOUNDRY

SAP Partner Workshop	**60 min**

Description

In this exercise, you'll learn how

- To Migrate SAP Cloud SDK Based Applications from SAP Cloud Platform Neo Environment to Cloud Foundry

For further reading on SAP Cloud SDK, click link below.
https://www.sap.com/germany/developer/topics/s4hana-cloud-sdk.html

Target group

- Developers
- People interested in learning about S/4HANA extension and SAP Cloud SDK

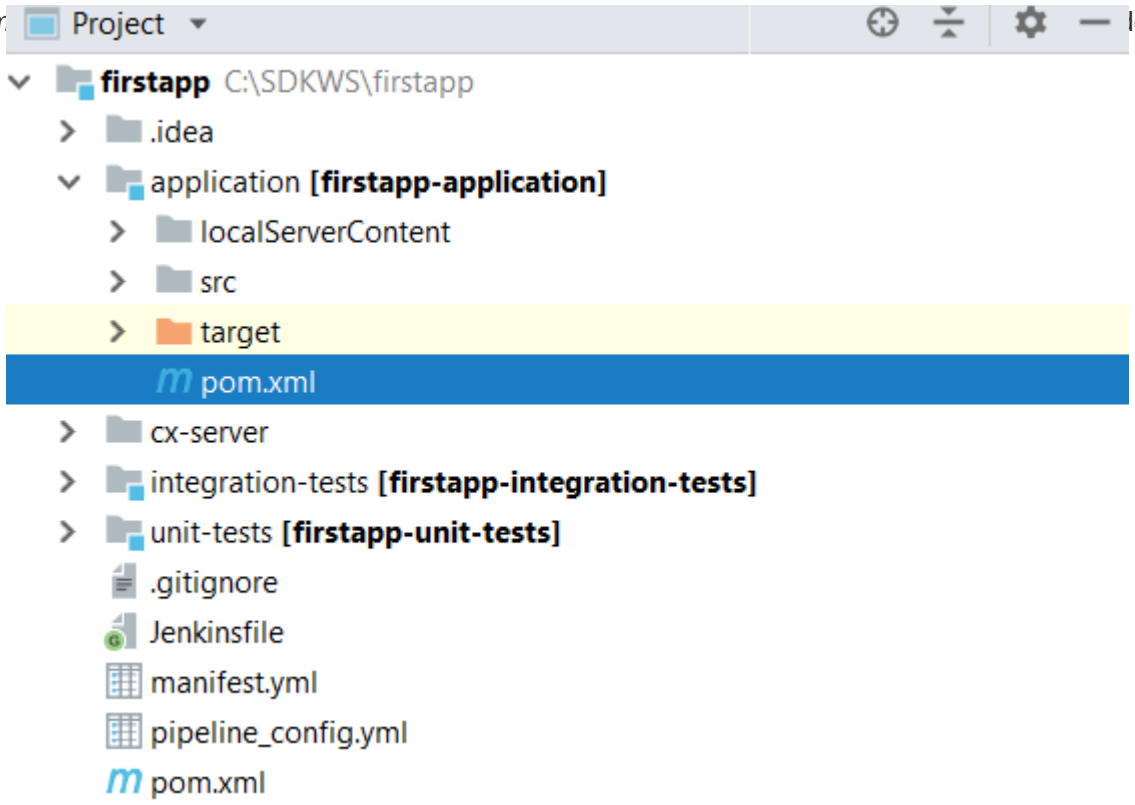
Goal

In the blog Step 2 with SAP Cloud SDK: HelloWorld on SCP Neo you can find the information on how to create a HelloWorld example on SAP Cloud Platform Neo using the SAP Cloud SDK. In this exercise, we will make the necessary modification to this HelloWorld application and deploy it to Cloud Foundry.

Prerequisites

Create a Hello World SDK project by following the blog Step 2 with SAP Cloud SDK: HelloWorld on SCP Neo

Step 1: Change pom.xml file

The *pom.xml* 

dependencies and plugins.

This file contains the dependency related to Neo environment. We need to comment those dependencies and add dependencies for Cloud Foundry environment. I will advise you to take a backup of pom.xml file before making any change.

Open this file and comment the entire <properties>, *<dependencies>, <build> and <profiles>* section. To comment, enclose the code in <!– .. –>. Instead of commenting it, you may also remove these sections.

Add below dependency and build sections to pom.xml file.

```
    <dependencyManagement>
        <dependencies>
            <dependency>
                <groupId>com.google.guava</groupId>
                <artifactId>guava</artifactId>
                <version>25.1-jre</version>
                <exclusions>
                    <exclusion>
                        <groupId>org.checkerframework</groupId>
                        <artifactId>checker-qual</artifactId>
                    </exclusion>
                </exclusions>
            </dependency>
        </dependencies>
    </dependencyManagement>

    <dependencies>
        <dependency>
            <groupId>com.sap.cloud.s4hana.cloudplatform</groupId>
            <artifactId>scp-cf</artifactId>
        </dependency>
        <dependency>
            <groupId>com.sap.cloud.s4hana</groupId>
            <artifactId>s4hana-all</artifactId>
        </dependency>
```

```
        <dependency>
            <groupId>org.slf4j</groupId>
            <artifactId>slf4j-api</artifactId>
        </dependency>
        <dependency>
            <groupId>org.slf4j</groupId>
            <artifactId>jcl-over-slf4j</artifactId>
            <scope>runtime</scope>
        </dependency>
        <dependency>
            <groupId>ch.qos.logback</groupId>
            <artifactId>logback-classic</artifactId>
        </dependency>

        <dependency>
            <groupId>javax.inject</groupId>
            <artifactId>javax.inject</artifactId>
            <scope>provided</scope>
        </dependency>

        <dependency>
            <groupId>javax.servlet</groupId>
            <artifactId>javax.servlet-api</artifactId>
            <scope>provided</scope>
        </dependency>
    </dependencies>

    <build>
        <finalName>${project.artifactId}</finalName>

        <plugins>
            <plugin>
                <groupId>org.apache.maven.plugins</groupId>
                <artifactId>maven-war-plugin</artifactId>
                <version>3.2.1</version>
                <configuration>
                    <attachClasses>true</attachClasses>
                </configuration>
            </plugin>

            <plugin>
                <groupId>org.apache.openejb.maven</groupId>
                <artifactId>tomee-maven-plugin</artifactId>
                <version>1.7.5</version>
                <configuration>
                    <tomeeClassifier>jaxrs</tomeeClassifier>
                    <context>ROOT</context>
                    <libs>
                        <lib>remove:slf4j-jdk14</lib>
                    </libs>
                </configuration>
            </plugin>
        </plugins>
    </build>
```

The final pom file should look like below.

```
<?xml version='1.0' encoding='utf-8'?>
<project xmlns="http://maven.apache.org/POM/4.0.0" xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
xsi:schemaLocation="http://maven.apache.org/POM/4.0.0 http://maven.apache.org/xsd/maven-4.0.0.xsd">
```

```
<modelVersion>4.0.0</modelVersion>
<name>firstapp - Application</name>
<description>firstapp - Application</description>

<groupId>com.sap.cloud.sdk.tutorial</groupId>
<artifactId>firstapp-application</artifactId>
<version>1.0-SNAPSHOT</version>
<packaging>war</packaging>

<parent>
    <groupId>com.sap.cloud.sdk.tutorial</groupId>
    <artifactId>firstapp</artifactId>
    <version>1.0-SNAPSHOT</version>
</parent>

<dependencyManagement>
    <dependencies>
        <dependency>
            <groupId>com.google.guava</groupId>
            <artifactId>guava</artifactId>
            <version>25.1-jre</version>
            <exclusions>
                <exclusion>
                    <groupId>org.checkerframework</groupId>
                    <artifactId>checker-qual</artifactId>
                </exclusion>
            </exclusions>
        </dependency>
    </dependencies>
</dependencyManagement>

<dependencies>
    <dependency>
        <groupId>com.sap.cloud.s4hana.cloudplatform</groupId>
        <artifactId>scp-cf</artifactId>
    </dependency>
    <dependency>
        <groupId>com.sap.cloud.s4hana</groupId>
        <artifactId>s4hana-all</artifactId>
    </dependency>

    <dependency>
        <groupId>org.slf4j</groupId>
        <artifactId>slf4j-api</artifactId>
    </dependency>
    <dependency>
        <groupId>org.slf4j</groupId>
        <artifactId>jcl-over-slf4j</artifactId>
        <scope>runtime</scope>
    </dependency>
    <dependency>
        <groupId>ch.qos.logback</groupId>
        <artifactId>logback-classic</artifactId>
    </dependency>

    <dependency>
        <groupId>javax.inject</groupId>
        <artifactId>javax.inject</artifactId>
        <scope>provided</scope>
    </dependency>

    <dependency>
        <groupId>javax.servlet</groupId>
        <artifactId>javax.servlet-api</artifactId>
        <scope>provided</scope>
    </dependency>
```

```
    </dependencies>

    <build>
        <finalName>${project.artifactId}</finalName>

        <plugins>
            <plugin>
                <groupId>org.apache.maven.plugins</groupId>
                <artifactId>maven-war-plugin</artifactId>
                <version>3.2.1</version>
                <configuration>
                    <attachClasses>true</attachClasses>
                </configuration>
            </plugin>

            <plugin>
                <groupId>org.apache.openejb.maven</groupId>
                <artifactId>tomee-maven-plugin</artifactId>
                <version>1.7.5</version>
                <configuration>
                    <tomeeClassifier>jaxrs</tomeeClassifier>
                    <context>ROOT</context>
                    <libs>
                        <lib>remove:slf4j-jdk14</lib>
                    </libs>
                </configuration>
            </plugin>
        </plugins>
    </build>
</project>
```

Step 2: Modify web.xml file

The web.xml file in src/main/webapp folder contains the deployment descriptor for your web application. This file contains the login and security related configuration which are not valid for cloud foundry. Open this file and comment <login-config>, <session-config>, <security-role> and <security-constraint>. After commenting, the file should look similar to below.

```
<?xml version="1.0" encoding="UTF-8"?>
<web-app xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
         xmlns="http://java.sun.com/xml/ns/javaee"
         xsi:schemaLocation="http://java.sun.com/xml/ns/javaee http://java.sun.com/xml/ns/javaee/web-
app_3_0.xsd"
         version="3.0" metadata-complete="false">

    <!--<login-config>-->
        <!--<auth-method>FORM</auth-method>-->
    <!--</login-config>-->

    <!--<session-config>-->
        <!--<session-timeout>20</session-timeout>-->
    <!--</session-config>-->

    <!--<security-role>-->
        <!--<role-name>Everyone</role-name>-->
    <!--</security-role>-->

    <!--<security-constraint>-->
        <!--<web-resource-collection>-->
            <!--<web-resource-name>All SAP Cloud Platform users</web-resource-name>-->
            <!--<url-pattern>/*</url-pattern>-->
        <!--</web-resource-collection>-->
        <!--<auth-constraint>-->
```

```
            <!--<role-name>Everyone</role-name>-->
        <!--</auth-constraint>-->

        <!--<user-data-constraint>-->
            <!--<transport-guarantee>NONE</transport-guarantee>-->
            <!--&lt;!&ndash; Use CONFIDENTIAL as transport guarantee to ensure SSL connection (HTTPS) on
public deployments-->
            <!--<transport-guarantee>CONFIDENTIAL</transport-guarantee> &ndash;&gt;-->
        <!--</user-data-constraint>-->
    <!--</security-constraint>-->

    <filter>
        <filter-name>RestCsrfPreventionFilter</filter-name>
        <filter-class>org.apache.catalina.filters.RestCsrfPreventionFilter</filter-class>
    </filter>
    <filter-mapping>
        <filter-name>RestCsrfPreventionFilter</filter-name>
        <url-pattern>/*</url-pattern>
    </filter-mapping>

    <filter>
        <filter-name>HttpSecurityHeadersFilter</filter-name>
        <filter-
class>com.sap.cloud.sdk.cloudplatform.security.servlet.HttpSecurityHeadersFilter</filter-class>
    </filter>
    <filter-mapping>
        <filter-name>HttpSecurityHeadersFilter</filter-name>
        <url-pattern>/*</url-pattern>
    </filter-mapping>

    <filter>
        <filter-name>HttpCachingHeaderFilter</filter-name>
        <filter-class>com.sap.cloud.sdk.cloudplatform.security.servlet.HttpCachingHeaderFilter</filter-
class>
    </filter>
    <filter-mapping>
        <filter-name>HttpCachingHeaderFilter</filter-name>
        <url-pattern>/*</url-pattern>
    </filter-mapping>
</web-app>
```

Step 3: Add manifest.yml file

manifest.yml is the deployment descriptor for CloudFoundry application. Create a new file in the root folder with name *manifest.yml* and add below code.

```
---
applications:

- name: firstapp
  memory: 768M
  random-route: true
  path: application/target/firstapp-application.war
  buildpack: sap_java_buildpack
  env:
    TARGET_RUNTIME: tomee
    JBP_CONFIG_SAPJVM_MEMORY_SIZES: 'metaspace:96m..'
```

Note: You may have to change the *name* and *path* if you have chosen different artifact id. You may also have to increase the memory if your application does more than the plain HelloWorld example.

Step 4: Deploy to Cloud Foundry

We have made all the necessary modification and the project is ready to be deployed to cloud foundry. Open your command line or terminal in IntelliJ IDEA. Change into the firstapp directory, the root directory of your project and run the following command:

```
cd /path/to/firstapp

mvn clean package
```

This tells maven to remove any files from previous assemblies (clean) and to assemble our project (package). After running the command there should be a directory target inside of the application directory, containing a file called firstapp-application.war. This is the file that we will deploy to Cloud Foundry.

Now you can deploy the application by entering the following command:

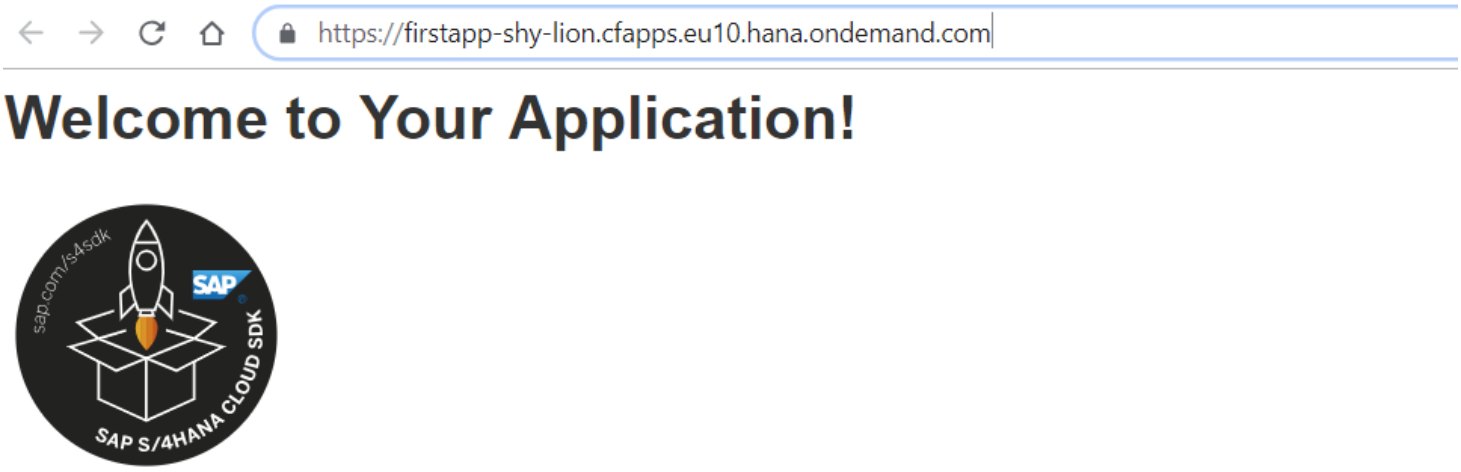
```
cf push
```

After the deployment is finished, cf CLI's output should look like this:

```
routes:           firstapp-shy-lion.cfapps.eu10.hana.ondemand.com
last uploaded:    Sun 24 Feb 11:55:50 IST 2019
stack:            cflinuxfs3
buildpacks:       sap_java_buildpack

type:           web
instances:      1/1
memory usage:   768M
start command:  JRE_HOME="META-INF/.sap_java_buildpack/sapjvm" JBP_CLASSPATH="" JBP_CONFIG_S
                JBP_CONFIG_SAPJVM_MEMORY_SETTINGS="" CATALINA_HOME="META-INF/.sap_java_build
                -Daccess.logging.enabled=false -Dlogback.configurationFile=file:META-INF/.sa
                -DSAPJVM_EXTENSION_COMMAND_HANDLER=com.sap.xs2rt.dropletaddon.JvmExtensionCo
                -agentpath:/app/META-INF/.sap_java_buildpack/jvm_kill/jvmkill-1.12.0.RELEASE
                ./META-INF/.sap_java_buildpack/tomee/bin/catalina.sh run
     state     since                  cpu     memory          disk           details
#0   running   2019-02-24T06:26:22Z   226.5%   753M of 768M   182.3M of 1G
```

Visit the application under its corresponding URL as it is shown in the output above.



That’s it. We have successfully migrated our application from SAP Cloud Platform Neo environment to Cloud Foundry environment without adapting any source code. Thanks to SAP Cloud SDK.