

Oracle Cloud Developer Workshop

Cloud-Native DevOps Lab 01

V1.0

ORACLE LAB BOOK | OCT 2018



ORACLE®

Disclaimer

The following is intended to outline our general product direction. It is intended for information purposes only, and may not be incorporated into any contract. It is not a commitment to deliver any material, code, or functionality, and should not be relied upon in making purchasing decisions. The development, release, and timing of any features or functionality described for Oracle's products remains at the sole discretion of Oracle.

Table of Contents

Disclaimer	1
Overview	1
Pre-Requisites	2
Practice 1: Create Oracle Developer Cloud Service project	3
Practice 2: Configure build job for Spring Boot sample application	7
Practice 3: Configure Application Container Cloud Service Deployment	11
Practice 4: Optional step: Make changes in the application	14
Practice 5: Optional step: Configure IDE to connect with DevCS	15

Overview

Oracle Developer Cloud Service is a cloud-based software development Platform as a Service (PaaS) and a hosted environment for your application development infrastructure. It provides an open-source standards-based solution to manage the application development life cycle effectively through integration with Hudson, Git, Maven, issues, and wikis. Using Oracle Developer Cloud Service, you can commit your application source code to the Git repository on the Oracle Cloud, track assigned issues and defects online, share information using wiki pages, peer review the source code, and monitor project builds. After successful testing, you can deploy the project to Oracle Java Cloud Service - SaaS Extension, publicly available Oracle Java Cloud Service instances, Oracle Application Container Cloud Service instances, or to an on-premise production environment.



The key features of Oracle Developer Cloud Service include:

Project creation, configuration, and user management

- Version control and source code management with Git
- Storage of application dependencies and libraries with Maven
- Continuous build integration with Hudson
- Wiki for document collaboration
- Issue tracking system to track tasks, defects, and features
- Repository branch merge after code review

Oracle Developer Cloud Service is available as a web interface accessible from a web browser and from Integrated Development Environments (IDEs) such as Oracle Enterprise Pack for Eclipse (OEPE), Oracle JDeveloper, and NetBeans IDE.

This workshop shows how to deploy Spring Boot sample application to Application Container Cloud Services using Oracle Developer Cloud Services.

The Spring Boot sample application is a web application serving simple JSP pages.

This tutorial demonstrates how to:

- create Oracle Developer Cloud Service project cloning an existing external Git repository
- configure build job for sample application
- configure Application Container Cloud Service deployment in Developer Cloud Service
- build and deploy sample application using Developer Cloud Service

Pre-Requisites

- Oracle Public Cloud Service account including Developer Cloud Service



Practice 1: Create Oracle Developer Cloud Service project

Overview

In this practice, you sign in to the Oracle Public Cloud - Oracle Developer Cloud Services console using your credentials provided by the instructor and create a development project.

Assumptions

Note: Some of the UIs might look a little different than the screenshots included in the instructions, but students can still use the instructions to complete the hands-on labs.

Before You Begin

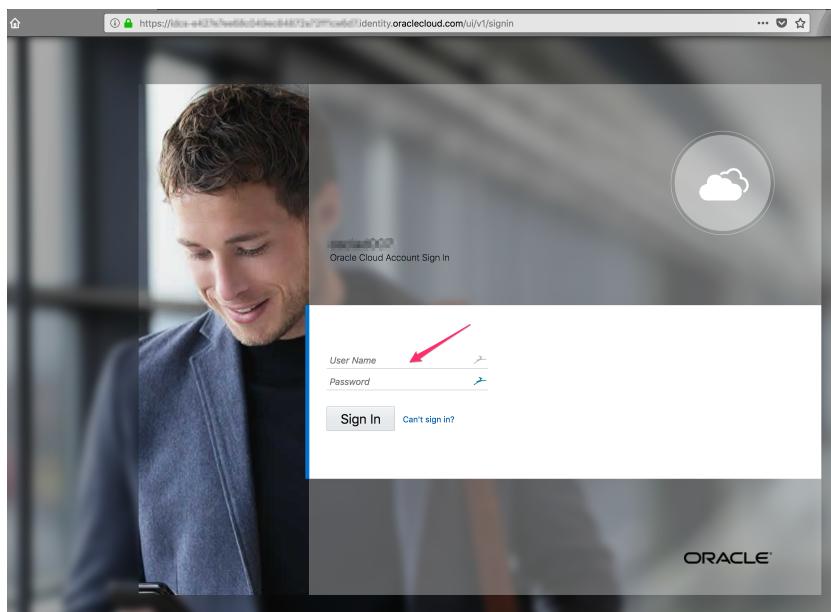
To sign in to the Console, you need the following:

- Username and Password (provided by the instructor)
- URL for the Console (provided by the instructor)
- Please use Firefox browser (Recommended)

Duration: 5 minutes

Tasks

1. Sign In
 - a. Open a supported browser and go to the Console URL.
 - b. Enter your username and password and click Sign In.



ORACLE®

When you sign in to the Console, the home page is displayed.

The screenshot shows the Oracle Developer Cloud Service home page. At the top, there's a search bar with the placeholder text "Select an existing project or create a new one.". Below the search bar is a navigation bar with four tabs: "Member" (which is selected and highlighted in blue), "Favorites", "Owner", and "All". To the right of the navigation bar is a green button with a plus sign and the text "+ New Project". A red arrow points to this button. Below the navigation bar, there's a section titled "Filter Projects" with two entries: "accs-employees-app" (ACCS Employees Sample Application) and "wrecker-oke-demo" (Oracle Pipeline & Oracle Engine for Kubernetes Demo). To the right of the projects is a sidebar with a section titled "The New Continuous Integration Engine Deep ..." which includes a video thumbnail showing a computer monitor displaying the developer cloud interface.

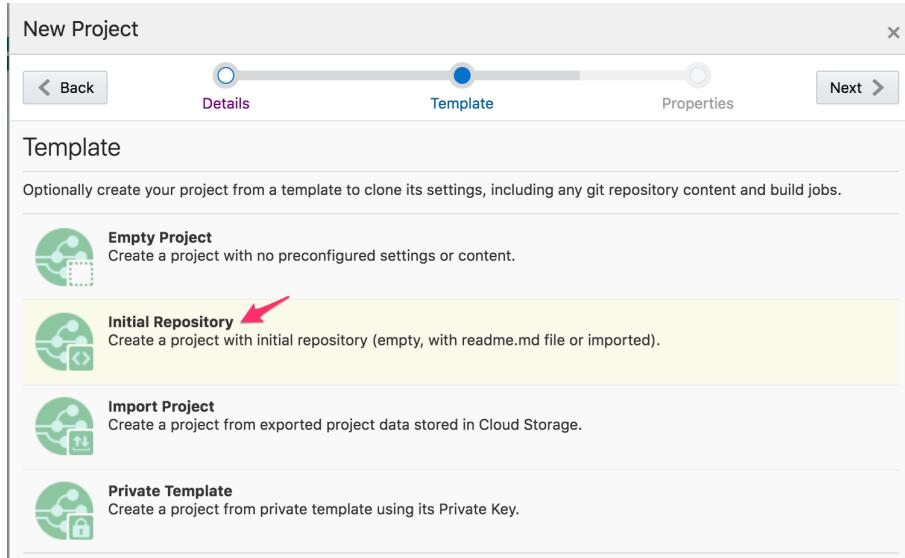
2. Create a New Project

- Enter the name of the project and set the desired properties. Click **Next**.
Use the name provided by the instructor : “employee-app-[your number]”

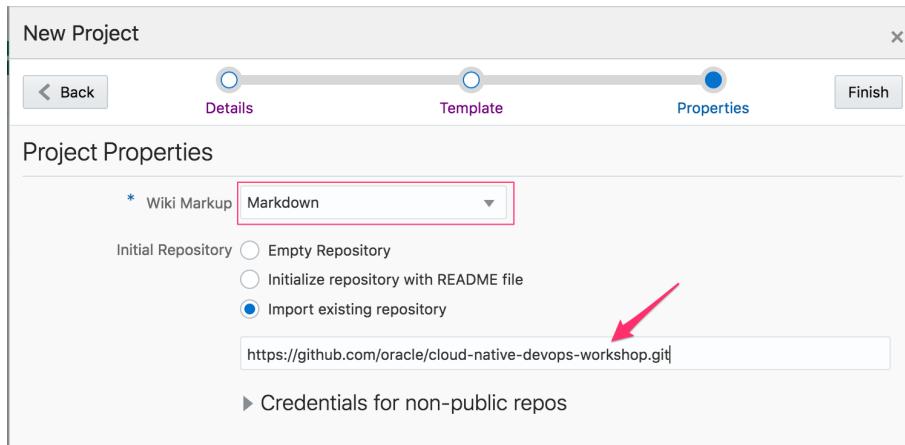
The screenshot shows the "New Project" wizard, step 1: Details. The wizard has three steps: "Details" (selected), "Template", and "Properties". The "Details" step contains the following fields:

- * Name: employee-app-01
- Description: Oracle Employee Application - Dev 01
- * Security: Private (Shared)
- * Preferred Language: English - English

b. Select “Initial Repository” as a Template and Click **Next**.



c. On the Properties page select “Markdown” as Wiki Markup Language and “*Import existing repository*” as SCM Properties. Enter or copy the <https://github.com/oracle/cloud-native-devops-workshop.git> repository address.



Now click **Finish** to create the project and to clone the specified repository.

During project creation, all tools needed to help on the Development Project are provisioned as : Repository of Sharing Code with a clone the specified repository (Git), a Maven Repository, a Build Server, and so on.

The screenshot shows the Oracle Developer Cloud Service interface. At the top, it says "employee-app-01 | Summary". In the center, there's a large blue wrench and screwdriver icon. Below it, the text "Project employee-app-01 is being provisioned." is displayed. A note says "Provisioning may take up to several minutes. Please wait until all modules are provisioned." To the right, there's a link "Log in to enable personalized checks and other features". On the left, a sidebar lists various project management tools: Agile, Binary Repository, Build, Code, Component Catalog, Deploy, Docker, Environments, Issues, Maven, Merge Requests, Mobile Build, Project, and Wiki. Each tool has a green checkmark next to it. At the bottom right, there's a red "ORACLE" logo.

After the creation process, the Project Home Page is presented.

The screenshot shows the Oracle Developer Cloud Service Project Home Page for "employee-app-01". The left sidebar has a dark theme with icons for Project, Code, Maven, Environments, Releases, Snippets, Merge Requests, Issues, Agile, Build, Deploy, Docker Registry, Wiki, and Administration. The main area has a blue header "employee-app-01 | Summary". Below it, there's a "ENVIRONMENTS" section with a "Create Your First Environment" button and a "RECENT ACTIVITIES - TODAY" section showing two entries: "System created hosted repository employee-app-01.git" and "Project created by oracle dev". To the right, there's a "REPOSITORIES" section with a "New Repository" button, a "Filter Git Repositories" dropdown set to "All", and two entries: "employee-app-01.git" (HTTP) and "Maven" (HTTP). There's also a "Docker" section which is currently empty. The bottom right corner has the red "ORACLE" logo.

Practice 2: Configure build job for Spring Boot sample application

Overview

Once the project provisioning is ready let's create the build job to compile and package the sample Spring Boot application to the desired format for Application Container Cloud Services.

Before You Begin

You should have completed Practice 1.

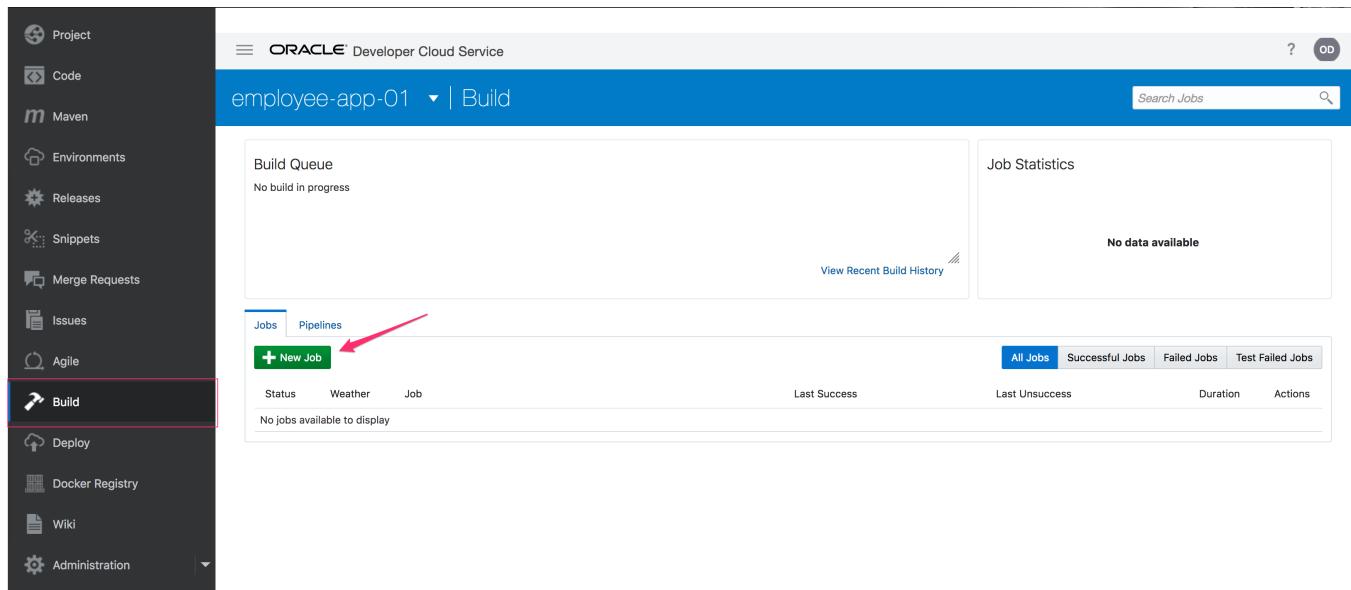
Before you can configure a new build job:

- You must have access to OCI VM Template. This machine will be applied to run the job.

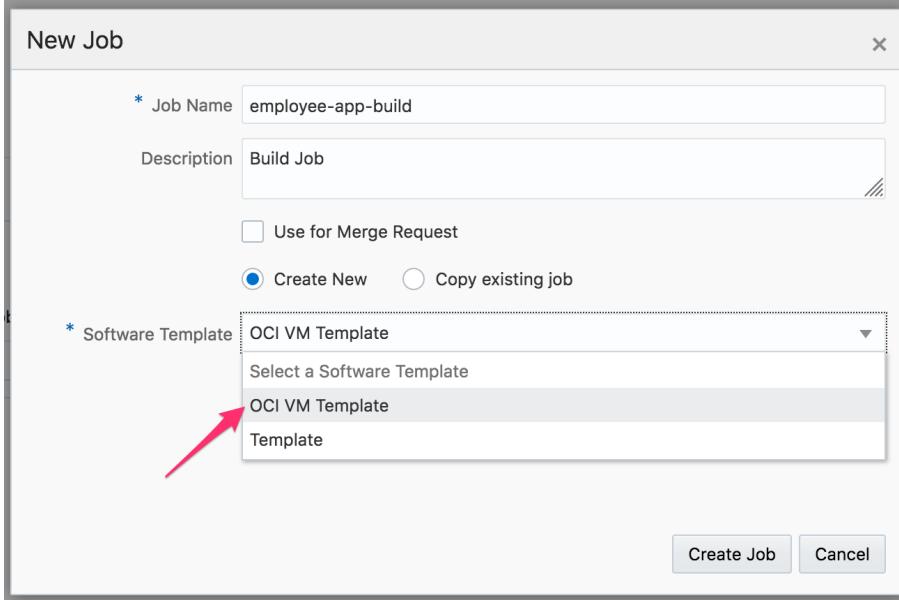
Duration: 5 minutes

Tasks – Create a new Build Job

Select **Build** item on the left side menu and click the **New Job** button.



Enter a name for the new job. Select the *Create New*. On the *Software Template* select **OCI VM Template**. Thus, Click on **Create job**



Add **Git** as the Source Control and select the project repository “employee-app-[your number]” as the source of code. Select “Automatically perform build on SCM commit” option. Leave the advanced settings default.

On **Builders** tab, add a **Maven Builder** and select *clean install* as Goals and *springboot-sample/pom.xml* to **POM File**.

employee-app-01 ▾ | Build

Jobs Overview > employee-app-build > **Configure**

Job Configuration

Configure Builders

Maven Builder

Goals: clean install

POM File: pom.xml (Red arrow points here)

Add Builder ▾

- Unix Shell Builder
- Ant Builder
- Maven Builder** (Selected)
- Gradle Builder
- Node.js Builder
- SQLcl Builder
- PSMcli Builder
- OCIClif Builder
- Docker Builder

On **Post Build** tab, add an **Artifact Archiver** activity and `springboot-sample/target/*.zip` to **File to archive** field.

employee-app-01 ▾ | Build

Jobs Overview > employee-app-build > **Configure**

Job Configuration

Configure Post Build Actions

Add Post Build Action ▾

Artifact Archiver

Artifacts from files:

Files to archive: `springboot-sample/target/*.zip` (Red box highlights this entry)

Files to exclude:

Artifacts from Maven build steps:

Archive Maven artifacts

Click on **Save** to update the new job configurations. To check the build job click on **Build Now** on the job's detail page. Once the job is done check the archived artifacts. It should be the following:
`springbootdemo-0.0.1.zip`



employee-app-01 ▾ | Build

Jobs Overview > employee-app-build

Job Details

Build Job

Notifications: On Off CC Me

Build History

By	Status	Build
3 minutes ago	Started	Duration 1 m 14 s Actions
#1	Success	

Build Trend

Log in to enable personalized checks and other features

Duration (Min)

Build Number

Legend: Canceled (grey), Failed (red), Test Failed (orange), Success (green)

employee-app-01 ▾ | Build

Jobs Overview > employee-app-build > Build #1 > Artifacts

Artifacts Archived

- springboot-sample
 - target
 - springbootdemo-0.0.1.zip (21.4 MB)
 - (All files in zip)

Please note the build job contains an extra build step which packs the default artifact `springbootdemo-0.0.1.war` and `manifest.json` (ACCS descriptor from the `springboot-sample/src/resources` folder) into a zip archive. This archive is the desired format to deploy a Java application to ACCS.

employee-app-01 ▾ | Code

employee-app-01.git master

/ springboot-sample / src / resources / manifest.json

JSON { December 20, 2016 6:05 PM +0000 Update manifest.json

```
1 { "runtime": { "majorVersion": "8" }, "command": "java -Dserver.port=$PORT -jar springbootdemo-0.0.1.war", "startupTime": "300", "notes": "SpringBoot demo application" }
```

ORACLE

Practice 3: Configure Application Container Cloud Service Deployment

Overview

Once the build is ready and the package to deployment is available let's configure the deployment of the sample Spring Boot application to the desired Application Container Cloud Services instance.

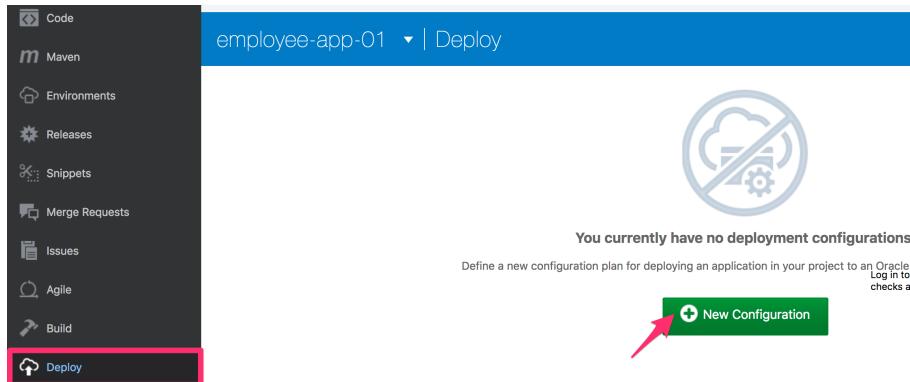
Before You Begin

- You should have completed Practice 2.
- You should have access to an ACCS instance. (provided by the instructor)

Duration: 10 minutes

Tasks

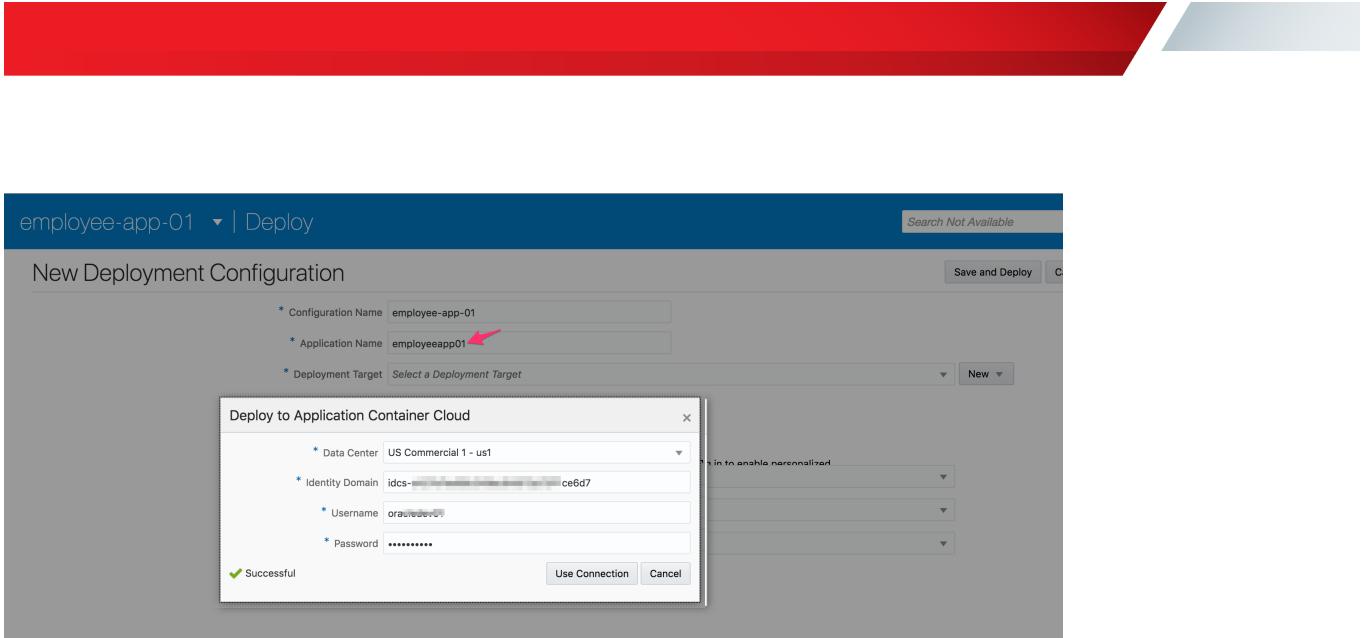
Now create deployment configuration which enable direct deployment to Application Container Cloud services after a successful build job. Change to **Deploy** page in DevCS and create **New Configuration**



Set the following properties:

- Configuration Name : employee-app-[your number]
- Application Name : employeeapp[your number] (* only can use characters letters and numbers)
- Data Center (provided by the instructor)
- Identity Domain (provided by the instructor)
- Username and Password (provided by the instructor)
- ACCS Properties: Java and Automatic type
- Job previously configurated
- Artifact previously generated

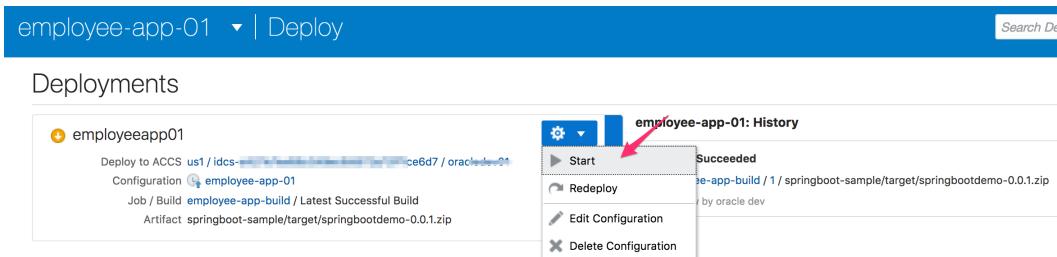
ORACLE®



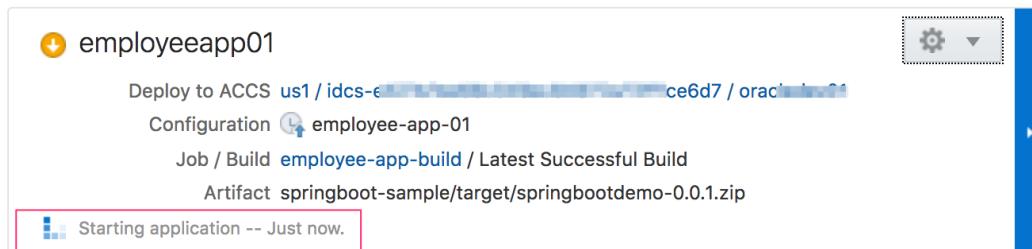
After connection had being tested, you Click Use Connection. Thus Click **Save**.

Build and deploy the sample application

To initiate a deployment to Application Container Cloud Service now there are two options. You can Start deployment process using the newly created Deployment configuration. Click gear icon and select Start.



The DevCS after deployment successful will try to Start the Application.



ORACLE



If Spring Boot sample application is deployed and executed correctly, the DevCS presents the history and the link to deployment application “employeeapp01”

Deployments

The screenshot shows the Oracle DevCloud Service interface. On the left, a sidebar lists "Deploy to ACCS us1 / idcs-... / employeeapp01". Below it, a message says "Last deployment succeeded -- Just now." A red arrow points to the application name "employeeapp01". To the right, a main panel displays the "employee-app-01: History" section. It shows three log entries: "Start Succeeded", "Deployment Succeeded", and "Create Succeeded", each with a timestamp and a link to the log entry.

And the application is available to be accessed.

The screenshot shows the Oracle Application Container Cloud Service application page. The URL is https://employeeapp01-...-central-1.oraclecloud.com. The page has a blue header with the Oracle Cloud logo and the text "Oracle Application Container Cloud Service". Below the header, there's a sub-header with icons for Java SE, Node.js, and PHP, followed by the text "Easy, rapid and agile deployment of Java SE, Node.js or PHP application. Experience full control and flexibility of your application in public cloud." The main content area displays the server time as "Tue Oct 16 08:45:35 UTC 2018". Below this, there's a section titled "About Oracle Application Container Cloud" with text explaining the service's features and how it runs applications in Docker containers. To the right, there's a diagram illustrating the architecture, showing a developer interacting with a "Developer Cloud Service" which then connects to a "Load Balancer". The Load Balancer distributes traffic to "Java SE" and "Node.js" services running in Docker containers. Arrows point from these services to various "Cloud Service" components: Database, Storage, Java, and Messaging.

The key features of Oracle Application Container Cloud are:
Pre-configured environment for Java and Node.js applications.
Java SE advanced features such as Java Flight Recorder, Java Mission Control, advanced memory management, and ongoing and timely security updates.
Open platform that supports all Java frameworks and containers such as Spring, Play, Tomcat, and Jersey.

ORACLE

Practice 4: Optional step: Make changes in the application

Prerequisites: Git client, Text editor

Clone your newly created Git repository hosted on Developer Cloud Service to your local machine using basic or your favorite Git tool. Make small changes for example in the JSP file. Push changes to DevCS remote repository, execute Build again and check the changes on the redeployed application.

Practice 5: Optional step: Configure IDE to connect with DevCS

Overview

The Eclipse IDE and the Oracle Enterprise Pack for Eclipse (OEPE) includes integration for Oracle Developer Cloud Service, which conveniently exposes the most common Cloud development tasks from within the IDE.

You can download the Eclipse IDE from <http://www.eclipse.org/> and OEPE from <http://www.oracle.com/technetwork/developer-tools/eclipse/downloads/index.html>.

If you are using the Eclipse IDE, download and install the Oracle Cloud Tools plugin from the Eclipse IDE marketplace. In OEPE, the plugin is installed by default.

Oracle Developer Cloud Service integration with the Eclipse IDE includes the following:

- A dedicated Oracle Cloud view that displays Oracle Developer Cloud Service projects of which you are a member
- Integration with Mylyn and the Oracle Developer Cloud Service Issues system
- Source control system integration with the Oracle Developer Cloud Service Git repository

Before You Begin

- You must have OEPE with Oracle Cloud Tools Plugin installed

Duration: 5 minutes

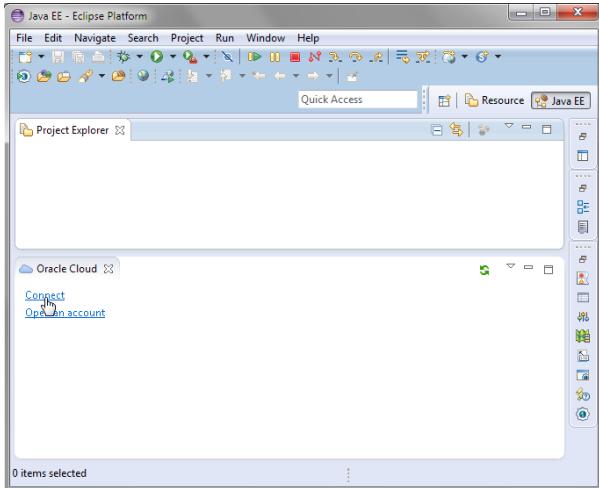
Tasks

You can log in to Oracle Developer Cloud Service from the Oracle Cloud view.

To open the Oracle Cloud view:

1. In the Eclipse IDE, from the Window menu, click Show View and then Oracle Cloud.
If the Oracle Cloud option does not appear in the Show View menu, click Other. In the Show View dialog, expand Oracle Cloud and select Oracle Cloud.
2. If you are connecting to Oracle Developer Cloud Service for the first time, click the Connect link. If the Connect link is not available, click the New Cloud Connection icon in the Oracle Cloud view.





3. In the Oracle Cloud Connection dialog box, select your account type.
 - a. Select Traditional Cloud Account if your account is a traditional account. Select Developer Cloud Service if you are using an IDCS account.
4. If you selected Traditional Cloud Account, enter your identity domain.
If you selected Developer Cloud Service, enter the Oracle Developer Cloud Service URL in the [https://<hostname>.oraclecloud.com/<org-name>/](https://<hostname>.oraclecloud.com/<org-name>) format.
5. In Username and Password, enter your Oracle Cloud user name and password.
6. Click Finish.

After the connection is successful, expand the identity domain node in the Oracle Cloud view, and then expand the Developer node to view projects that are assigned to you.

