\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

Oracle EMEA Pre-sales – Digital Business Platform v3

**Integrating Application Express and Developer Cloud Service**

**----------------------------------------------------------------------------------**

Jules Lane – EMEA Specialist Pre-sales

Edi Vasermann – ETC Cloud Pre-sales

May 2016

\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

Contents

[Demo Attributes 2](#_Toc451938844)

[Executive Summary 2](#_Toc451938845)

[What’s New in this Release 4](#_Toc451938846)

[Storyline 4](#_Toc451938847)

[Supporting Material 4](#_Toc451938848)

[Prerequisites 5](#_Toc451938849)

[Known Issues 5](#_Toc451938850)

[Demo Setup 5](#_Toc451938851)

[Before each demo 5](#_Toc451938852)

[Login / environment details 6](#_Toc451938853)

[Demo Flow 7](#_Toc451938854)

## Demo Attributes

|  |  |
| --- | --- |
| **Product(s)** | Oracle Application Express 5.0, Oracle Database 12.1, Oracle Database Cloud Service, Oracle Developer Cloud Service 16.1 |
| **Date last updated** | May 24, 2016 |
| **Author(s)** | Jules Lane |
| **GSE Demo Release Version(s)** | Digital Business Platform v3 |
| **Demo Title(s)** | Oracle APEX – DevCS Integration |

## Executive Summary

Oracle Application Express (APEX 5) is the fastest and easiest way to develop database-centric web and mobile apps, either in the cloud or on premise. It is included as a standard feature of all Oracle databases and is pre-configured and ready to run in all Oracle Database Cloud Services. It is ideal for very rapid and iterative development methods and for larger projects has traditionally been used in conjunction with 3rd party version control systems.

Oracle Developer Cloud Service is a cloud-based software environment for your application development infrastructure. It provides an open source, standards-based solution to manage and automate the control, collaboration, and deployment applications within Oracle Cloud. It helps you to manage the application development life cycle effectively through integration with the widely used Hudson, Git and Maven tools, as well as issue tracking, and team wikis. DevCS integrates with all appropriate Oracle cloud services as well as on-premise environments.

This demonstration provides an illustration of how the Oracle Developer Cloud Service can be used to manage APEX development artifacts and environments, thus enabling APEX projects to be governed in the same way as other development technologies, such as Java, Python, Ruby, Javascript etc.

Other demonstrations are available to showcase the wider functionality of both APEX and DevCS.



## What’s New in this Release

* First Release

## Storyline

An APEX developer uses Oracle Developer Cloud Service to manage his APEX source code and automate the deployment of new or updated APEX apps to a cloud based test environment.

The scenario is;

1. [Optionally build and] Run a simple app in any APEX environment
2. Login to target APEX cloud environment to show that the APEX workspace is not there
3. Export new version of the app sql files to local GIT repository
4. Using the Sourcetree Git client (or Git commands), commit file into the DevCS GIT repository
5. [Note that other SQL and XML files to create the Workspace, database objects and data have already been created using APEX utilities and loaded into the DevCS Git repository]
6. Show and explain the main DevCS capabilities, focusing on Git repository and Hudson continuous deployment engine.
7. Run a pre-defined set of Hudson Builds to;
   1. Execute a script to provision a new DBaaS environment (optional – takes 30 mins)
   2. Execute scripts to import the sql and XML files into the APEX test environment in the new DBaaS cloud service in order to create the new APEX Workspace, schema objects, data and APEX app.
8. Run the APEX app in the new DBaaS APEX environment.
9. Make minor changes to the App and repeat the process to create and run an updated version of the app in the target cloud environment.

## Supporting Material

* [APEX](http://www.oracle.com/technetwork/developer-tools/apex/overview/index.html)
* [Database Cloud Service](https://cloud.oracle.com/en_US/database?resolvetemplatefordevice=true)
* [Developer Cloud Service](https://cloud.oracle.com/en_US/developer_service?resolvetemplatefordevice=true).

## Prerequisites

* Any modern browser
* Git installed on your laptop, if possible with a Git client such as Sourcetree.
* Database Cloud and DevCS accounts.
  + Not necessarily both in the same identity domain. If you use ID’s other than PaaSdem0002 for the DBCS target you will need to edit the configuration of many of the build steps and shell scripts as per the setup doc noted below.

## Known Issues

* None

## Demo Setup

* Initial set up of DevCS environment - see [this](https://stbeehive.oracle.com/content/dav/st/APEX%20JL/Public%20Documents/DevCS%20Demo/APEX%20DevCS%20Demo%20Setup%20v1%20(EV).doc) document.

## Before each demo

* Sourcetree - Update with the current GSE password for PaaSdem0002 - Tools 🡪 Options / Authentication
* DevCS - Run the Build ***Dropall*** to remove workspace JULES (and the schema and users needed by our app) from the target environment
* Login to the target APEX environment as internal / admin / <apex\_admin\_pwd>, ready to show that the new workspace is not initially there

## Login / environment details

Replace the following with details of the environments *you* plan to use.

**1/ APEX Development Environment**

* APEX on my local laptop
  + jules / jules / Pitta111
* Local git repository (origin)
  + C:JL\APEXDevCS\git (this was originally cloned from the master on DevCS, put there originally by Edi)

**2/ DevCS environment**

* + url = https://developer.us2.oraclecloud.com/developer73023-paasdem0002/#projects/developer73023-paasdem0002\_jules-1/
  + Cloud Identity Domain = Paasdem0002
  + Cloud User = cloud.admin
  + Cloud pwd = <get this weeks from GSE launchpad>
  + Git Repository (master) = APEX DevCS.git
    - The build *Dropall* should be run *in advance* to remove workspace JULES from the target APEX environment

**3/ APEX target (test) Environment**

* Database Cloud environment (created in advance, not during the demo)
  + Cloud Identity Domain = PaasDemo0002
  + Cloud User = cloud.admin
  + Cloud pwd = <get this weeks from GSE launchpad>
  + System / Db pwd = Pitta111\_
  + APEX Admin pwd = Pitta222”
* APEX
  + url = https://129.191.5.61/ords/pdb1/f?p=4550:1
    - internal / admin / Pitta111!
    - jules / jules / Pitta111 (to be created during the demo)

## Demo Flow

* Login to target environment (internal) and show no workspace called JULES (and therefore no schema, no app)
* Optionally also use SQL Dev to show no Schema (User) called JULES in that Db.

**Move entire new app from Dev to Test via DevCS**

1. Run any APEX app – note the workspace and schema names
   1. Optionally make a change to the app.
2. Export – save to file and put in local project repository. Note you can export the workspace, the app or individual pages.
   1. Rename file as *buildApexApp.sql* (case sensitive)
   2. Show our local Git repo (C:\JL\APEXDevCS)
   3. Move file to C:\JL\APEXDevCS\dbcs\sql – REPLACE (this is in the local git repository – overwrites the old version of the file)
   4. Note that all other files needed by the app (DDL scripts and Data files) have already been created (with standard APEX utilities or SQL Dev)
   5. Make any small update to C:\JL\APEXDevCS\Docs\*GSE Pwords*.txt and save.
3. Sourcetree (our Git client)
   1. Select Log/History tab at the bottom
   2. Select 1st line (Uncommitted changes) – Updated files shows up as yellow (git command line equivalent is “git status”)
   3. Select earlier versions to see changes made then
   4. Click on the text file (or Word file) to see diffs of the content from last version
   5. COMMIT with comment (git command line equivalent is “git commit –m “comment text”)
4. DevCS
   1. Note that we are in a DevCS project called Jules 1
   2. Show each of the main tabs
   3. Show help
5. DevCS / Code tab (= centralized git repository, or Git hub)
   1. Can have any number of repositories to suit your working method
   2. Explain repo structure (matching our local one)
   3. Show scripts, that call sql files
   4. End showing \ dbcs \ sql
6. Sourcetree
   1. Push (note the number of changes to be pushed)
   2. Show full Output (quick)
   3. Click on new most recent line, showing commit comment.
   4. Click on the GSE Pwords.txt file to show diffs.
7. DevCS / Home
   1. Note recent actions
   2. Click on commit number
   3. Shows content changes (may just be the date in the sql file)
   4. Click Code – dbcs – sql. Note commit comment and source user.

**Automated Builds (= Hudson continuous deployment engine)**

A DevCS (Hudson) build is a piece of processing triggered from within DevCS. It could be anything written in any language. Multiple builds can run in series to complete a complex workflow including dependencies and parameters. Here we have a series of builds as follows; *Build 🡪 shell script 🡪 (in most cases) SQL Plus script.* The SQL Plus scripts were all created using the appropriate APEX DevOps utility. Shell scripts were written by hand.

1. Right click Build tab to open it in a new browser tab.
   1. Explain execution summary and history.
      1. 1st 2 builds have not yet been run from DevCS
      2. Status relates to the most recent execution
      3. Weather shows general history (hover over the cloudy symbol for 03 or 04)
      4. Hover over any build to show description. Note that 08 is triggered by new commit of the updated app file.
   2. Click on *build 03\_CopySQLFiles*
      1. Shows full history for this build and relates builds (projects)
      2. Click on the console for any run to see details. The errors were due to a wrong SSH key preventing access to the target DBCS.
      3. Click Back
   3. Click Configure
      1. Main – basic info
      2. Parameters – the target DBCS IP address
      3. Source Control – location of files used in the build
      4. Triggers - other builds that trigger this one. None here, so this one is triggered manually
      5. Environment - Specify timeout etc for this build
      6. Build Steps - what this build actually executes. This is a shell script with the target DBCS IP address parameter passed to it
      7. Post Build - build *04\_createApexWorkspace* is to be triggered after this one has completed
      8. Advanced - click “?” (help) to review these options
      9. Click Build tab to return to build steps summary
   4. Right click on the Code tab to open the repo in a new tab
      1. Review dbcs/scripts/*copySQLFiles.sh*
      2. Receives the target IP address
      3. SCPs all the sql files from a specific folder in our DevCS Git repo to a directory called oracle on the target DBCS environment and then a data file from another folder. These are all executed by subsequent builds.
      4. Click Jobs Overview (or Build)
   5. Click on build *04\_createApexWorkspace*
   6. Click Configure
      1. Main – basic info
      2. Parameters – none
      3. Triggers - *03\_copySQLFiles*
      4. Build Steps - this build executes the copySQLFiles.sh script. Required parameter values of target DBCS IP address, user id and passwords hard coded.
      5. Post Build - build *05\_crerateDBSchemaAndObjects* is to be triggered after this one has completed
      6. Switch to the code tab
   7. Review dbcs/scripts/*createApexWorkspace.sh*
      1. Receives 3 parameters from the build
      2. Executes a SQL Plus script called *createApexWorkspace.sql*
   8. Review dbcs/sql/*createApexWorkspace.sql*
      1. This is APEX Workspace export file created by the developer in the dev environment. By calling the APEX API to import it into the target environment APEX will recreate the Workspace and APEX users. Note lines 29-32.
2. Click on 03\_CopySQLFiles
   1. Click Build Now and accept default parameter value
   2. Click Jobs Overview and watch the Build Queue and Job Stats chart (you *may* need to refresh page initially)
   3. Builds 03 to 07 should run in series.
   4. While this process is starting;
      1. Go to target APEX environment as Internal / Admin / Pitta111! to show no Workspace called JULES.
      2. Once job 04 has completed you can refresh the APEX page to see that it has appeared.

**See results in target environment**

1. Go to target APEX environment and refresh to show new workspace
2. Login to workspace jules / jules / Pitta111 (will need to change pwd – use Pitta222)
3. Use SQL Workshop to show tables with data.
4. View / run Climbing Logbook app – same as it was in the dev environment
5. Optionally also view new Schema (User) and Database objects in SQL Dev.

**Update just the App and move to Test**

1. Show the build *08\_updateApexApp*
   1. Source Code 🡪 Branches 🡪 Advanced 🡪 Included Regions
   2. Triggers – based on SCM Polling (as defined above).
2. Make a change to the dev version of the app (e.g. change title to v1.3 etc and switch to a different ThemeRoller style)
   1. Export – save to desktop and rename as *updateApexApp.sql* (CASE sensitive)
   2. Move file to C:\JL\APEXDevCS\dbcs\sql – REPLACE what is there(overwrites the old version)
3. Sourcetree - commit and then push
4. View DevCS home page to note the new commit
5. Quickly go to the Build tab and note / await 08 being queued and run
6. After it has completed;
   1. Click Git Logs icon on RHS to see it was triggered by SCM polling.
   2. Click Console output to see output from shell script and SQL Plus.
7. In APEX workspace refresh the App Builder page to show and run the updated app.
   1. Note that App is replaced (not a new one added) because the APEX import process called by the SQL script specifies to use the same Application ID as the exported app, so it is the same App ID as was already there, so the app is replaced. If it specified to use a new App ID we would have a new version of the app created in addition to the one already there.

**Builds, shell scripts and SQL scripts**

|  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- |
| **Build** | **Parameters** | **Triggers**  (= triggered by) | **Build Steps** | **SQL Script** | **Post Build**  (= triggers**)** | **Notes** |
|  |  |  | Shell scripts located in dbcs/scripts | Located in dbcs/sql |  |  |
| 01\_createStorage  Optional | OPC ID  OPC User  OPC Pwd  SC Name | None  (Manual) | createStorageContainer.sh | None | - | Takes ? Mins  Only needed if Db will require back up. |
| 02\_createDBCS  Optional | OPC ID  OPC User  OPC Pwd  Db Name  Sys Pwd  SC Name  JSON File | None  (Manual) | createDBCS.sh | None | - | Takes 20 – 30 mins |
| 03\_copySQLFiles | DBCS Public IP | None  (Manual) | copySQLFiles.sh | None | 04 |  |
| 04\_createApexWorkspace | - | 03 | createApexWorkspace.sh  Parameters hard coded | createApexWorkspace.sql | 05 |  |
| 05\_createDBSchemaAndObjects | - | 04 | createDBSchemaAndObjects.sh  Parameters hard coded | createDBSchemaAndObjects.sql | 06 |  |
| 06\_buildApexApp | - | 05 | buildApexApp.sh  Parameters hard coded | buildApexApp.sql | 07 |  |
| 07\_loadApexData | - | 06 | loadApexData.sh  Parameters hard coded | loadApexData.sql | - |  |
| UpdateApexApp | - | None  (Manual) | updateApexApp.sh  Parameters hard coded | updateApexApp.sql | - | Ideally this could be triggered by a updated file being pushed into the DevCS repo. |
| Dropall | - | None  (Manual) | dropAll.sh | dropAll.sql | - | Removes workspace and therefore app, users and schema |

**Demo Details**

| **View** | **Click stream** | **Talk stream** |
| --- | --- | --- |
|  | Login to your Dev APEX;  **Note** – Feel free to use your own APEX environment.  . | We log in to the APEX development Environment.  Note the workspace we are using |
|  | Go to the Application Builder |  |
|  | Click on the xxx App |  |
|  | Click Run |  |
|  | Login to the target APEX admin environment.  Internal / admin / <db pwd> |  |
|  | Show that our workspace does not exist yet. | We want to use DevCS to recreate this workspace in a target environment (most likely, test) and then also migrate the app into it. |
|  | Back in our dev environment |  |
|  | Export the App to a file.  Locally rename the file to be; |  |
|  | Scripts  Edit and then download the DDL for tables needed for the app. |  |
|  | Export data for a table to a file.  Show other data files already unloaded. |  |
|  |  |  |
|  |  |  |
|  |  |  |
|  |  |  |
|  |  |  |
|  |  |  |
|  |  |  |
|  |  |  |
|  |  |  |
|  |  |  |