Self Service Application Deployment in a Runtime Instance

Jason A. Straub
Oracle America, Inc.
Dublin, Ohio, USA

Keywords:

APEX
Runtime Only
Self Service
Development
Application Deployment
IT Operations

Introduction

Oracle best practice is to have your production Application Express instance configured as a runtime only instance. A runtime only instance requires any application changes be done through either SQL*Plus or SQLDeveloper, but can all your developers be trusted with such access? This paper will describe a solution developed internally at Oracle to solve this problem and allow developers to deploy application changes to a staging instance and then the production instance all through a web based (APEX) application. This paper will also look under the covers to see what makes the Self Service Deployment Application work.

Problem Statement

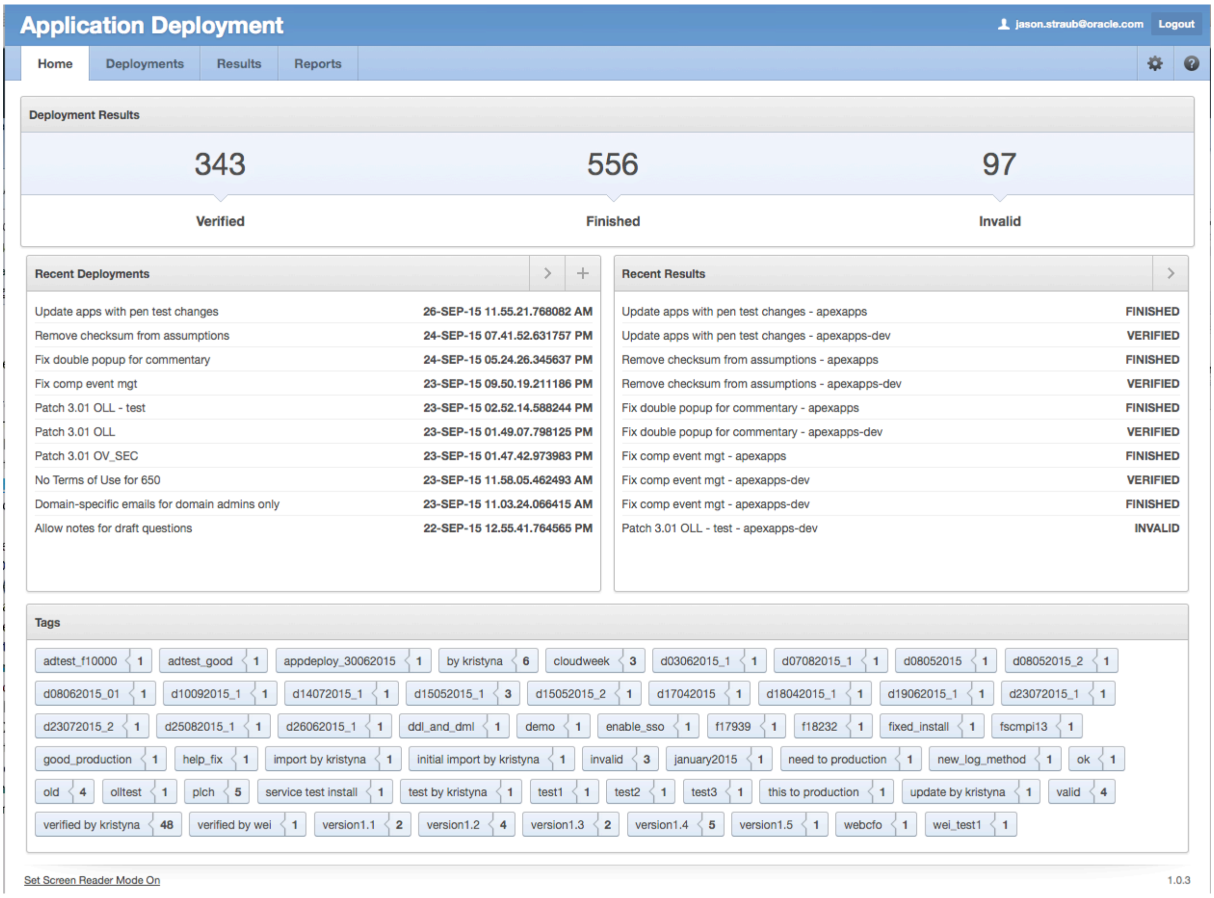
At Oracle, we have found ourselves deploying more and more customer facing Internet applications as Application Express applications. Working with our Product Development IT (PDIT) organization, we developed a professionally managed, security hardened environment for applications delivered on the Internet. Some of these applications include:

- AskTom, https://asktom.oracle.com
- The PL/SQL Challenge, https://plsqlchallenge.oracle.com
- Oracle Learning Library (OLL), https://apexapps.oracle.com/pls/apex/f?p=44785:1:0

So what's the problem? As more and more Internet facing applications were being developed in Application Express, we needed to provide a dedicated instance, and move these off of apex.oracle.com, which is intended as a demo instance. We needed a more durable infrastructure for these applications. Of course this instance needed to be a runtime only instance, as all production internet facing Application Express instances should be, to maximize security and minimize the attack surface. But since this instance was runtime only, and we could not give direct SQL*Plus or SQLDeveloper access to the instance we had to provide some type of self service infrastructure for developers to update their applications in the stage instance, and then the production instance.

How Oracle Does IT

Introducing, the Application Deployment Self-Service application! The Application Deployment Self-Service application provides a way for application owners to deploy application updates to stage for verification and once verified apply the same updates to production. Application owners write an installation script to invoke SQL files consisting of Oracle Application Express export files, as well as scripts to make DDL or DML changes to the applications. The developer then zips the scripts into a file and creates a deployment, by uploading the zip throught the Application Deployment Self-Service application. Once the zip file is uploaded, they can apply the deployment to the stage environment. The developer then verifies the changes on the stage system, changes the status of the deployment to verified, and then has the ability to apply the changes to production. The developer also gets access to the full installation log of their script running on each instance.



Deployment Results

343	556	97
Verified	Finished	Invalid

Recent Deployments

Update apps with pen test changes	26-SEP-15 11.55.21.768082 AM
Remove checksum from assumptions	24-SEP-15 07.41.52.631757 PM
Fix double popup for commentary	24-SEP-15 05.24.26.345637 PM
Fix comp event mgt	23-SEP-15 09.50.19.211186 PM
Patch 3.01 OLL - test	23-SEP-15 02.52.14.588244 PM
Patch 3.01 OLL	23-SEP-15 01.49.07.798125 PM
Patch 3.01 OV_SEC	23-SEP-15 01.47.42.973983 PM
No Terms of Use for 650	23-SEP-15 11.58.05.462493 AM
Domain-specific emails for domain admins only	23-SEP-15 11.03.24.066415 AM
Allow notes for draft questions	22-SEP-15 12.55.41.764565 PM

Recent Results

Update apps with pen test changes - apexapps	FINISHED
Update apps with pen test changes - apexapps-dev	VERIFIED
Remove checksum from assumptions - apexapps	FINISHED
Remove checksum from assumptions - apexapps-dev	VERIFIED
Fix double popup for commentary - apexapps	FINISHED
Fix double popup for commentary - apexapps-dev	VERIFIED
Fix comp event mgt - apexapps	FINISHED
Fix comp event mgt - apexapps-dev	VERIFIED
Fix comp event mgt - apexapps-dev	FINISHED
Patch 3.01 OLL - test - apexapps-dev	INVALID

Tags

adtest_f10000	1
adtest_good	1
appdeploy_30062015	1
by kristyna	6
cloudweek	3
d03062015_1	1
d07082015_1	1
d08052015	1
d08052015_2	1
d08062015_01	1
d10092015_1	1
d14072015_1	1
d15052015_1	3
d15052015_2	1
d17042015	1
d18042015_1	1
d19062015_1	1
d23072015_1	1
d23072015_2	1
d25082015_1	1
d26062015_1	1
ddl_and_dml	1
demo	1
enable_sso	1
f17939	1
f18232	1
fixed_install	1
fscmpi13	1
good_production	1
help_fix	1
import by kristyna	1
initial import by kristyna	1
invalid	3
january2015	1
need to production	1
new_log_method	1
ok	1
old	4
olltest	1
plch	5
service test install	1
test by kristyna	1
test1	1
test2	1
test3	1
this to production	1
update by kristyna	1
valid	4
verified by kristyna	48
verified by wei	1
version1.1	2
version1.2	4
version1.3	2
version1.4	5
version1.5	1
webcfo	1
wei_test1	1

Illustration. 1: Application Deployment Self Service Application

How it Works

The Application Deployment Self-Service application is made up of three main components. They are:

1. Oracle Application Express (APEX)
2. Oracle Database

3. Shell scripts (C shell)

Oracle Application Express of course is the web application framework used for this application. What better platform for deploying APEX applications than an APEX application itself! The original deployment system was built using APEX 4.2.x, but now is running happily on 5.0.1.

Next we have the Oracle Database, which as you know, is required to run APEX. But this deployment system also uses two very important features of the Oracle database to make it work.

Firstly we have proxy connect. Proxy connect allows us to connect to a database as a user, while providing the credentials for a completely different user. There is simply an alter user statement to do, and then you can use the connect syntax you see above.

In designing this application, we considered various methods of what we could use to connect as all these application users and run SQL. We did not want to have to manage the passwords of every user. For one, that would be a security challenge. Proxy connect allows us to only have to manage one password.

Consider the following example:

```
alter user JASON1 grant connect through APPDEPLOY;

connect APPDEPLOY[JASON1]/oracle@localhost/prod
```

In the above example, a grant is made for JASON1 to allow proxy connection to the database through the APPDEPLOY database user. The APPDEPLOY database user is the user that owns the Application Deployment Self-Service Application. This means we can connect to the database, parse SQL as JASON1, but only need to have the credentials for APPDEPLOY.

Next is the powerful feature of DBMS_SCHEDULER, which allows us to run jobs in the database. These jobs don't have to run PL/SQL. They can execute shell scripts for example. To execute a shell script, we first have to provide an OS credential. Typically you would use a minimally privileged OS user (which the production Application Deployment Self Service application does), but for simplicity in the following listing, the credentials for the OS user that owns the Oracle software is provided.

```
sys.dbms_scheduler.create_credential(
        credential_name => 'OS_Credential',
        username        => 'oracle',
        password        => 'oracle');
```

The application uses create_program of DBMS_SCHEDULER to define the shell script we want to invoke, and we define arguments for the script, and create the job to invoke the script. Notice that we are indicating there are two parameters in the listing. The example below is just one of the shell scripts invoked for this application. The application has an administrative interface to allow new workspaces to use the application. The first step is to do the proxy connect grant, which is done in the application using the DBMS_SCHEDULER method.

```
sys.dbms_scheduler.create_program(
        program_name            => 'proxy_grant',
        program_type            => 'executable',
        program_action          => '~/ad/grant.csh',
        number_of_arguments     => 2,
        enabled                 => false);

sys.dbms_scheduler.define_program_argument(
        program_name            => ’proxy_grant',
        argument_position       => 1,
        argument_name           => 'grant_schema',
        argument_type           => 'VARCHAR2',
        out_argument            => false);

sys.dbms_scheduler.define_program_argument(
        program_name            => 'proxy_grant',
        argument_position       => 2,
        argument_name           => 'deploy_path',
        default_value           => '~/ad/deploy',
        argument_type           => 'VARCHAR2',
        out_argument            => false);

sys.dbms_scheduler.enable(
        name            => 'proxy_grant');

sys.dbms_scheduler.create_job(
        job_name        => 'grant_job',
        program_name    => 'proxy_grant',
        credential_name => 'OS_Credential',
        enabled         => false);
```

The last component is a shell script that runs on the operating system of the database server. This application uses the C shell, but bash could have easily been used. Below are the contents of grant.csh.

```
#!/bin/csh

setenv DEPLOY_PATH $2
source $DEPLOY_PATH/env.csh
cd $DEPLOY_PATH

sqlplus -r 2 /nolog @grant.sql $1 >> $APPDEPLOY_PATH/grant.log
```

The SQL script invoked by the shell script is grant.sql, below.

```
set define '^' verify on feedback on termout on
--------------------------------------------------------------------
--
--  Copyright (c) Oracle Corporation 2014. All Rights Reserved.
--
--    NAME
--      grant.sql
--
--    DESCRIPTION
--            This script grants the connect through for proxy
connections.
```

```
--
--
--    MODIFIED   (MM/DD/YYYY)
--    jstraub     02/13/2014 - Created
--     jstraub       02/17/2014 - Replaced hardcoded APPSTORE with
APPUN set to APPDEPLOY
--
-------------------------------------------------------------------
define PROXYUSER = '^1'
define APPUN = 'APPDEPLOY'

@@connect_stage.sql ^APPUN

alter user ^PROXYUSER grant connect through ^APPUN;

@@connect_prod.sql ^APPUN

alter user ^PROXYUSER grant connect through ^APPUN;

exit
```

The administrative page in the Application Deployment Self-Service application simply accepts the database username associated with a workspace, and then calls a page process to run the job defined above.

```
    begin

        sys.dbms_scheduler.set_job_argument_value(
            job_name            => 'APPDEPLOY.grant_job',
            argument_position   => 1,
            argument_value      => :P17_SCHEMA);

        sys.dbms_scheduler.run_job(
            job_name                => 'APPDEPLOY.grant_job',
            use_current_session     => true );

    end;
```

Conclusion

Combining the power of the Oracle Database and shell scripting, very complex business problems can be solved by an Oracle Application Express application. Although this paper did not describe exactly how to build the Application Deployment Self-Service application, it did provide enough information and examples to show the combination of using DBMS_SCHEDULER with shell scripts, the same combination that makes the application work.

Contact address:

Jason Straub
Principal Member of Technical Staff
Oracle Application Express
+1.937.431.4574
Oracle America, Inc.
5650 Blazer Parkway, Dublin, Ohio USA 43017-3562
jason.straub@oracle.com
@JasonAStraub