Lab 1 - Steps 2D, 2E

Like | Updated 25 January 2016 by Sunil K. Jha | Tags: *None*

DevOps Technical Workshop

(D) Creating a MobileFirst Platform environment

Previously, you made a Liberty environment and dropped the Jenkins application into it. This was tedious as you had to type out commands manually, hit go on them, move files, configure the server.xml file, and not to mention, you had to download the files first! If only there was a way to do it all in a managed way, so the next time you have to do this you can just execute one command, and it does it all for you. Puppet and Chef are the tools for this. For the next activity, you will only use Puppet, but the principles are the same whether using it or Chef. You could have used puppet to set up all of the configuration of Jenkins, but you did it manually. The Puppet scripts you will be executing will create a Liberty profile in a similar manner to what has been done previously. In this task, you will use puppet, not only to create the Liberty Profile to house your MobileFirst Platform, but also to install the product, create the databases, and configure it to run in the newly created environment.

There are some prepared puppet scripts for execution. Move to the puppet directory on the terminal .
cd ~/Documents/DevOps/lab1/puppet

Here, you will find a collection of puppet (.pp) scripts. Each of these contains a basic rule, such as *ensure a file or directory exists*, *ensure it has specific contents* or *permissions,* or even *execute some command*. The puppet manifest files have been broken into the following tasks (Puppet terminology : a manifest is just a file containing a collection of instructions.):

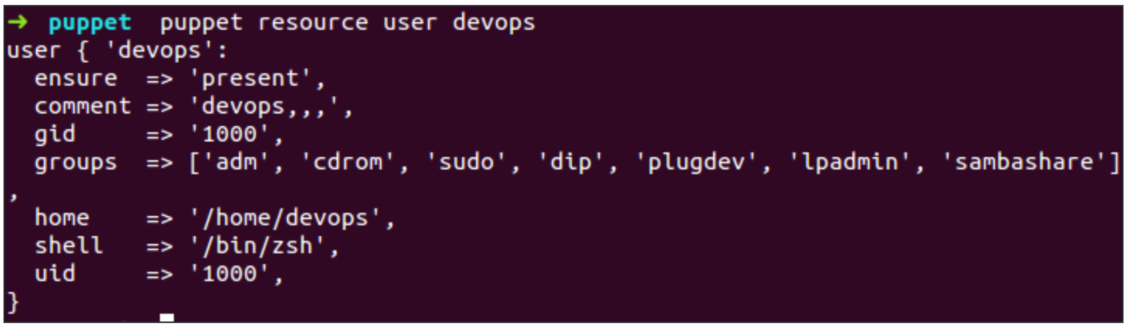
- Install liberty server
- Create a liberty server
- Install MobileFirst Platform Server into that liberty server
- Start and stop tasks

There is a Puppet clean-up script (clean.pp) which resets all these files to how they should be for when setting up the CI environment. Puppet is designed for use on distributed systems and managing dependencies such as *user accounts and other services (eg ntp)*. The example scripts use puppet to allow you to create MobileFirst Platform Server environments in the pattern you'd expect to find across many systems. Instead of installing multiple servers on multiple VMs, we'll install them in multiple directories to *simulate* the behavior of being on different systems.

Puppet has some key functions:
`puppet parser validate $FILENAME` - This checks your script for errors
`puppet apply --noop $FILENAME` - This stages your changes and simulates what would occur if you applied your changes
`puppet apply $FILENAME` - This writes your changes to the machine

Puppet allows you describe the state of the system. Try running the command:

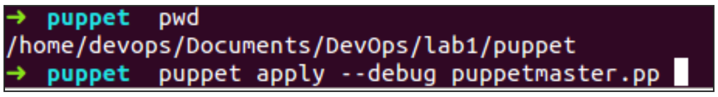
`puppet resource user devops` - This will show you the information about the user the VM is logged in as.



```
➜  puppet  puppet resource user devops
user { 'devops':
  ensure  => 'present',
  comment => 'devops,,,',
  gid     => '1000',
  groups  => ['adm', 'cdrom', 'sudo', 'dip', 'plugdev', 'lpadmin', 'sambashare']
,
  home    => '/home/devops',
  shell   => '/bin/zsh',
  uid     => '1000',
}
```

To run the Puppet scripts for this lab, all that is required is to execute this script from the Puppet directory inside lab1. For additional debugging and logging information, add `--debug` to the below command:
`puppet apply puppetmaster.pp` This may take a while to run.

(*pwd is a linux command to check what directory you are in and to ensure you are running a command from where you should be.)*



```
➜  puppet  pwd
/home/devops/Documents/DevOps/lab1/puppet
➜  puppet  puppet apply --debug puppetmaster.pp
```

This will then execute all the puppet manifests required to install a MobileFirst Platform (new name for Worklight) server into Liberty and configure the Derby database it requires. The reason MFP has been chosen to use is because it was the product used on Homebase, where all of the material and exercises in this course have come from.

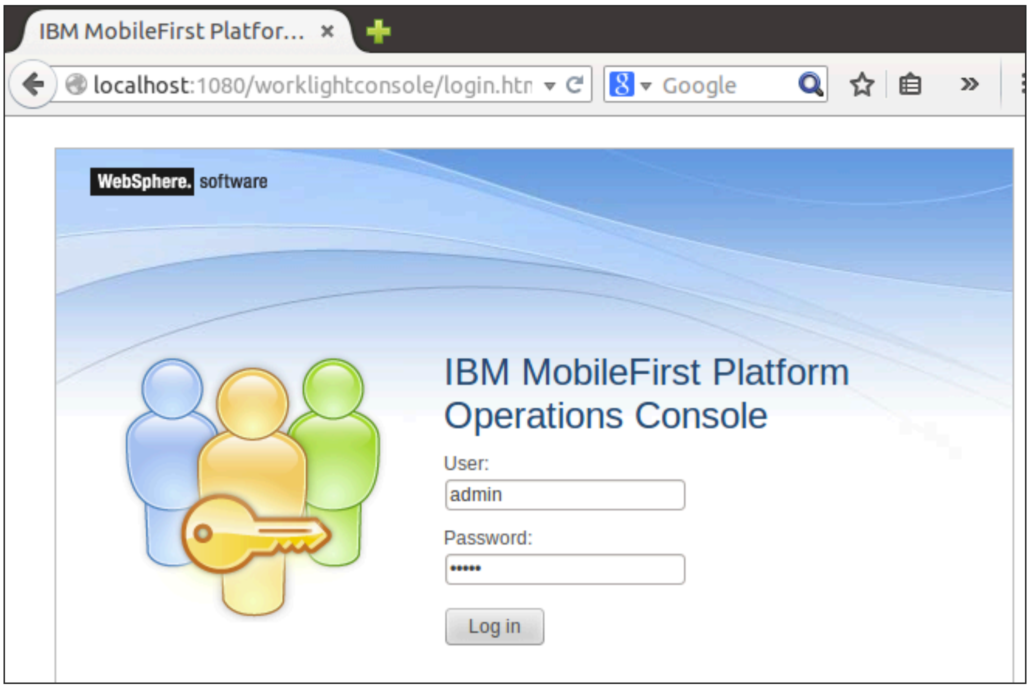
To change the endpoints and install locations of the Liberty/MFP instance, open the params.pp file located in the puppet directory in your text editor.

The parameters you might want to configure are outlined below as these are configured on a per install basis. In our VM we will have to ensure the port is changed if we build multiple MFP servers as new servers should not conflict with existing servers.

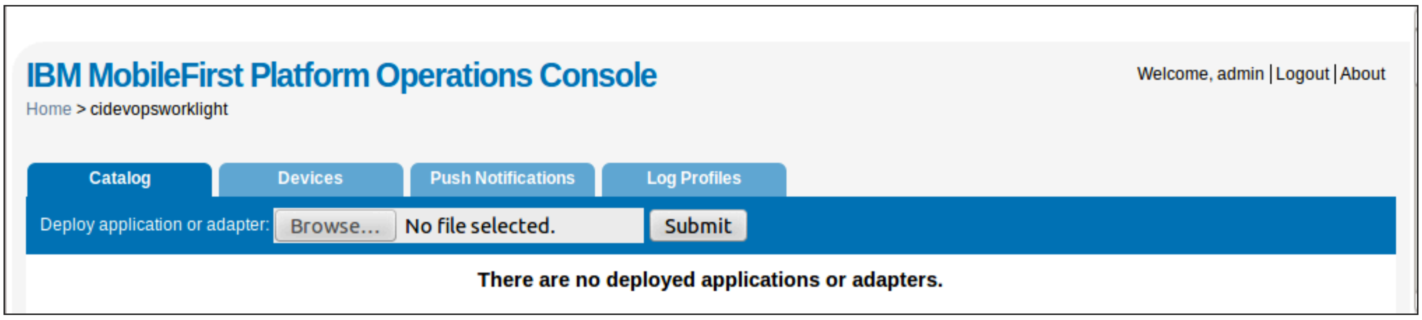
For the purpose of this lab, the following can be ignored:

```
$LIBERTY_TARGET= "/home/devops/ci/Liberty"
$SERVER_NAME= "cidevopsworklight"
$CONTEXT_ROOT= "/cidevopsworklight"
$DERBY_ADMIN_DB= "WLADMIN"
$WORKLIGHT_DB= "WORKLIGHT"
$WORKLIGHT_REP= "WLREPORTS"
$defaultHttpPort = "1080"
$defautlHttpsPort = "1443"
```

As well as installing MFP, the Puppet script will start your MFP server. To verify this happened correctly, once the Puppet script has completed with no errors, go to the MFP server in your browser: http://localhost:1080/worklightconsole/index.html
Log in with **username**: admin **password**: admin


IBM MobileFirst Platfor...
localhost:1080/worklightconsole/login.htm
Google
WebSphere. software
IBM MobileFirst Platform Operations Console
User:
admin
Password:
•••••
Log in

If you are able to log in successfully, the following screen will appear:


IBM MobileFirst Platform Operations Console
Welcome, admin | Logout | About
Home > cidevopsworklight
Catalog
Devices
Push Notifications
Log Profiles
Deploy application or adapter:
Browse...
No file selected.
Submit

There are no deployed applications or adapters.

Note: If the script generates an error in the middle through execution or does not run as expected :

1. Remove any files created midway through the process to clean out your VM. I have written a script to stop the MFP server if it has been started and to delete the contents of the CI directory. It is located in the `$SCRIPT_DIR`. Move into it and execute `remove-mfp.sh`:

```
cd $SCRIPT_DIR
./remove-mfp.sh CI
```

2. Run the puppet apply clean.pp command to clean the files and try again with puppet apply puppetmaster.pp. Note if you use the clean.pp file; it will reset the values for the environment to correspond to the CI environment ie port 1080 and context route /cidevopsworklight.

To become more familiar with Puppet, it is recommended to download the learning VM from the following URL:

https://puppetlabs.com/download-learning-vm.

(E) Creating a (useful) Jenkins / MFP job

We have created an Mobile First Platform (MFP) server environment, and a Jenkins server (*and have checked they are up and running)*. Jenkins can now be used to build and deploy code that runs on MFP. For MFP, this could be cross-platform hybrid code or it could be an adapter that mobile apps connect to (for example, integrate into existing back-end systems)

I have created an MFP project with a single adapter and app stored on the VM so now lets wire it all together!

Use the cheat sheet supplied to guide you if you need to stop / start your servers.

Create a following the same method as before (freestyle project). Give it a sensible name this time, such as Continuous-Integration. This job is going to be used to build and deploy the develop branch of code of the locally stored Git project. You have a prepared script that will be used to execute the job; its contents can be found at `/home/devops/Documents/WL_Build_Deploy/build-main.sh.`

Open the file in the same text editor you used above and you'll see the script is designed to be executed with parameters. This future proofing of the script will be discussed later but for now, you will just get Jenkins to execute the script. The main function in the script builds the adapter and deploys it to your local server. In order for it to know the server name/endpoint in which to deploy to, it is configured in the script. In a real world scenario, this script would be checked into source code and pulled out on each build, this allows us to manage the build scripts as we would any other source code. It also provides a backup for scripts, which when designing these systems, will evolve and improve. An important thing to note here is the name of the server to which you are deploying. *NOTE: If you were feeling adventurous and changed any of the params.pp file in a previous lab, you should ensure that these values match the ones exposed in the script.*

Back on the Jenkins console, in the job configuration page for the job you just created ,
we are going to point Jenkins to the above script and have it execute it. To do this we will add a build step to Execute Shell.

```
export DEPLOYER_ENV="CI"
/home/devops/Documents/WL_Build_Deploy/build-main.sh
```

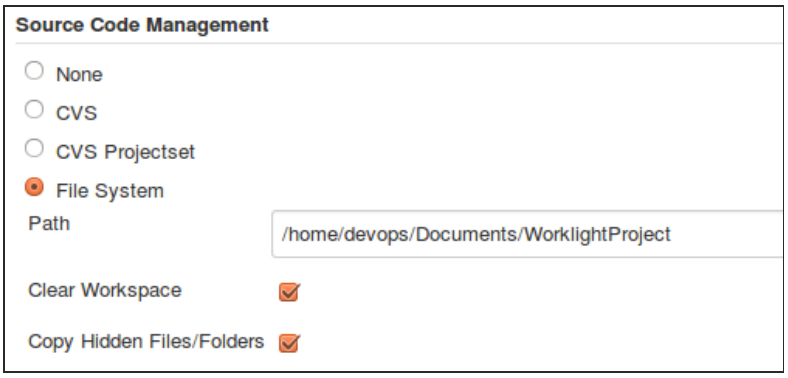
This will cause Jenkins to call the script and execute it. You could copy all the contents of the script into the Jenkins execute shell box, but this is not advisable. If you have multiple jobs with similar purposes, you want to minimize the amount time you have to re-creating it in each job. By keeping configuration stored on Jenkins to a minimum and checking scripts into source code, you have better control and reliability of the environment.

The location of the MFP project it will build can be hardcoded into the build-main.sh script. For an example, you will check the code out of a location in the *filesystem*. On the Source Code Management label (back on Jenkins, under Job Config), select **File System** and enter the **location** (NOTE : If the File System is not available; ensure that you have installed the plugin for it... It's "File System SCM"):

```
/home/devops/Documents/WorklightProject
```

Tick the boxes for **Clear Workspace** and **Copy Hidden Files.**

You are selecting these options to ensure that no fragments of previous builds remain when building each job. Worklight projects contain some hidden files, which are also necessary for the build to be successful.


Source Code Management

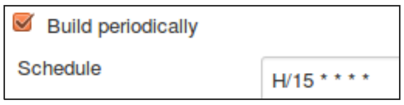
None [✓ File System]
CVS
CVS Projectset
File System

Path: /home/devops/Documents/WorklightProject

Clear Workspace: ✓

Copy Hidden Files/Folders: ✓

Now, add a build trigger to your job **Build Triggers**. You will configure it so the build will execute every 15 minutes.

To do this add, select **Build periodically** and enter `H/15 * * * *` in the text box. 

✓ Build periodically

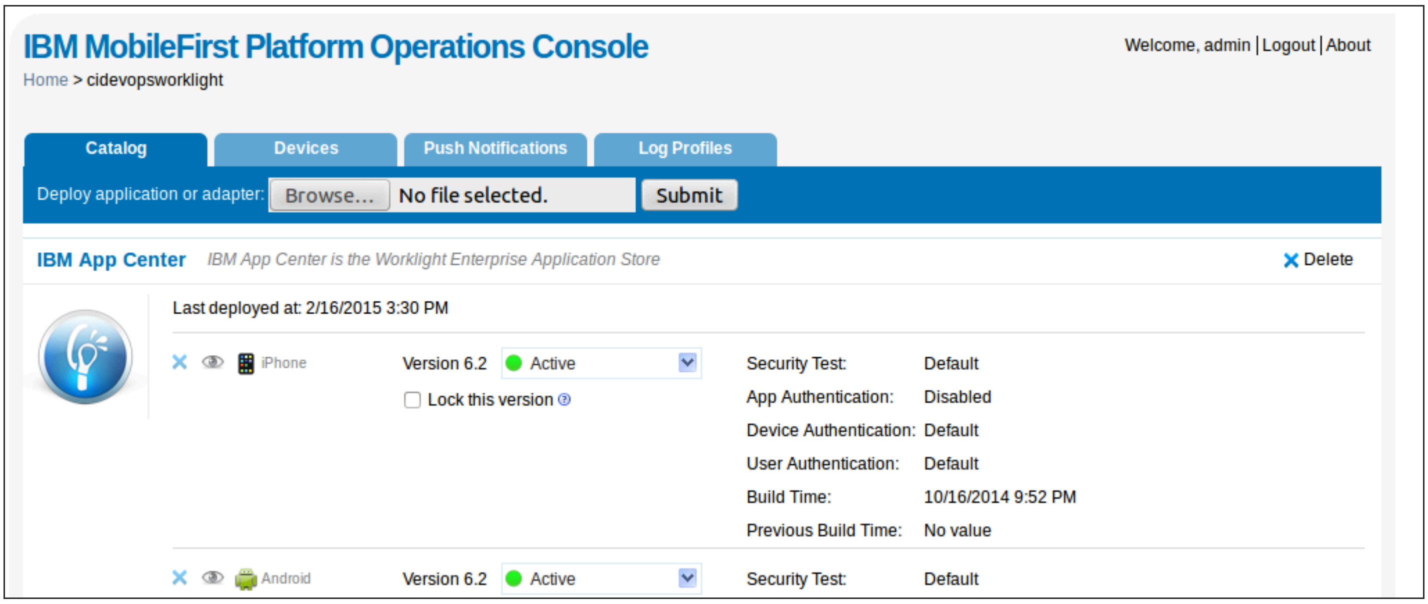
Schedule: H/15 * * * *

For more information on what these values mean, click the **Help** icon beside the text box.

In a real world example we may not want to have code build and deploy happening every 15 minutes, but for the purposes of this VM, we are setting this threshold. It will allow your Jenkins instance quickly generate a lot of build data which at the end of the labs will show a complete story. In that real world, it is likely that every time a user commits code to the repository, a build should be automatically triggered. If this were a release train, then a time frame such as every hour would be appropriate whereby changes are just taken and pushed directly to production.

When you are ready, click **Apply** and then **Save** and build ("Build Now") your first real Jenkins job. This will take a while. Once successful, check your MFP console to find the time the artifacts deployed.

The console will look like this once successful:


IBM MobileFirst Platform Operations Console

Welcome, admin | Logout | About

Home > cidevopsworklight

IBM App Center *IBM App Center is the Worklight Enterprise Application Store*

Delete

Last deployed at: 2/16/2015 3:30 PM

iPhone Version 6.2 Active

Lock this version

Security Test:	Default
App Authentication:	Disabled
Device Authentication:	Default
User Authentication:	Default
Build Time:	10/16/2014 9:52 PM
Previous Build Time:	No value

Android Version 6.2 Active

Security Test:	Default

Run a second build of the Jenkins job.

Question for your consideration: Why has the iPhone environment not updated? Investigate the console output for the first task to identify the reasons and discuss with your table. This should **not** be fixed as part of this lab.

Back

Next

Comments

There are no comments.

Back to Top