Exercise – Jenkins and Git Pipeline

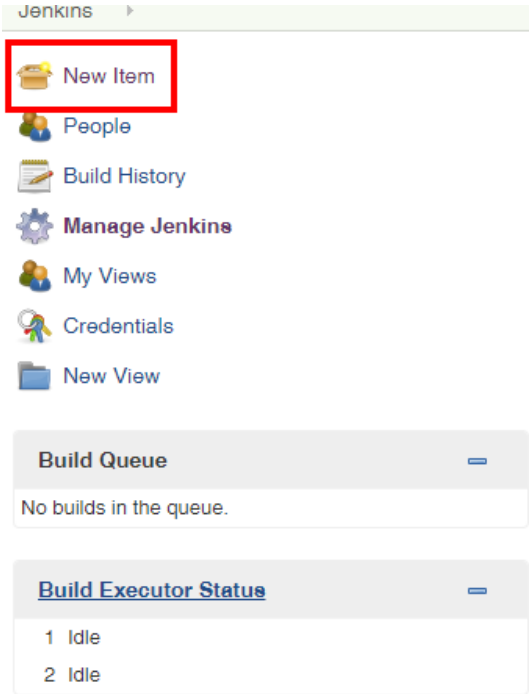
Objective

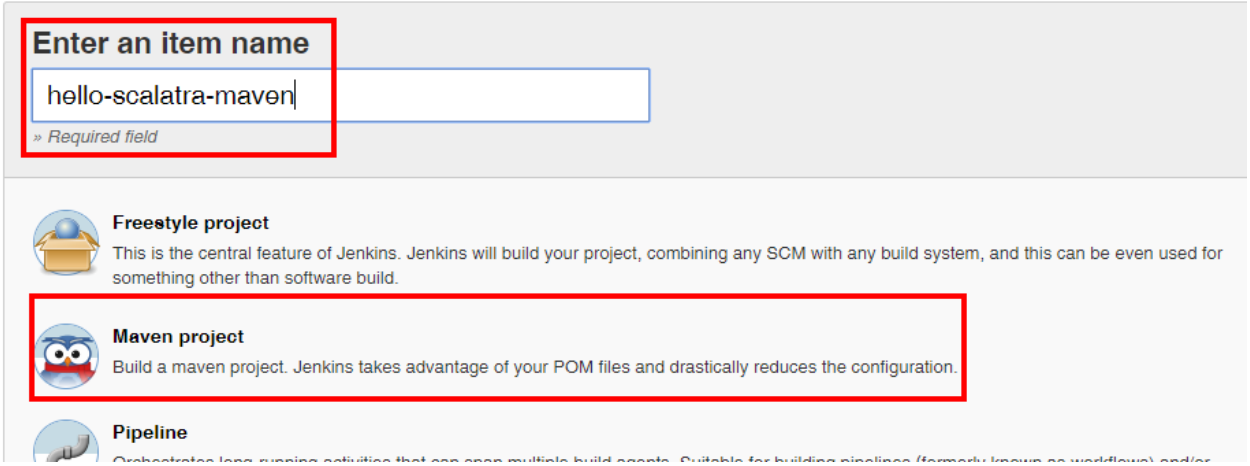
The objective of this exercise is to setup the git hooks and triggers for a project, secure Jenkins and setup source control backup for the Jenkins files.

Overview

Part 1 - Create a Maven Project in Jenkins

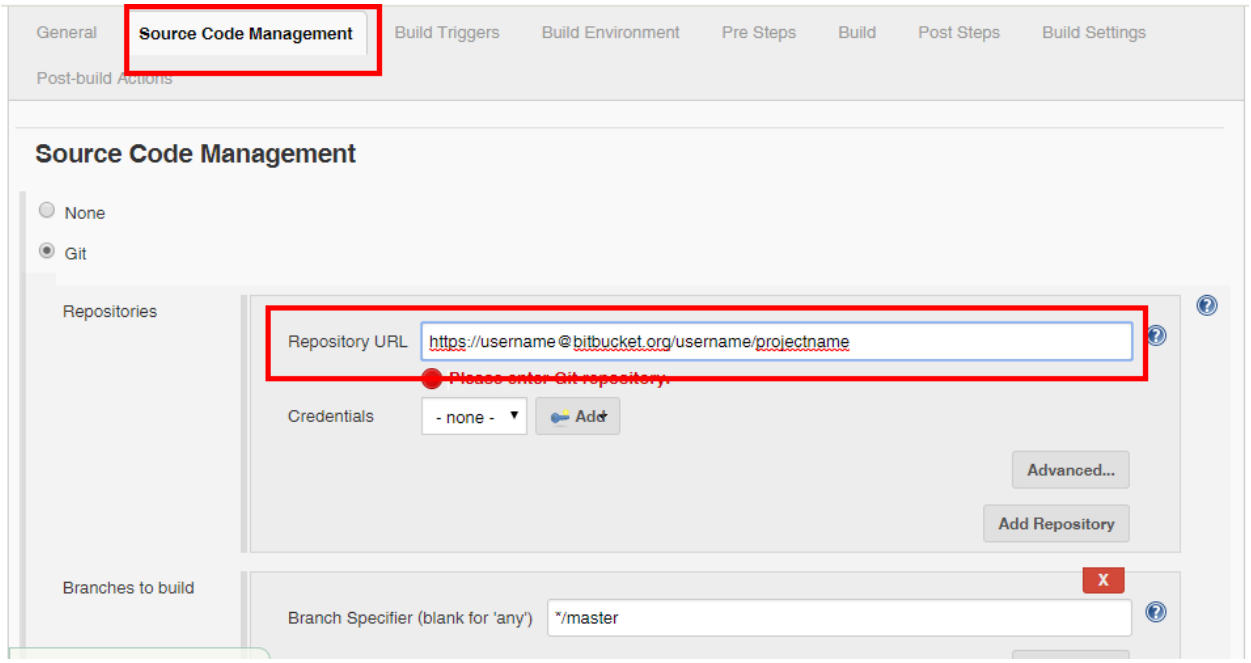
Click on “New Item" to start creating a new job. You need to give the job a name - use "hello-scalatra-maven" and select the "Maven Project" option from the list of project types below.



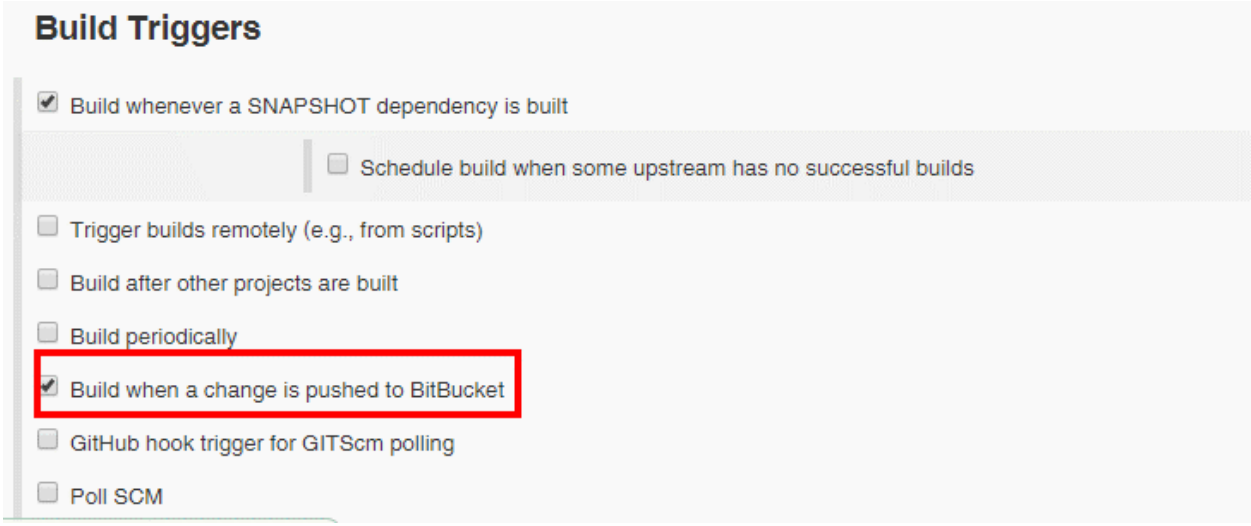


Click **"OK"** to create the project. You will be taken to a set of configuration options.

Under source code management select "Git" and add **your** hello-scalatra git repository to this **(not the one shown!)**

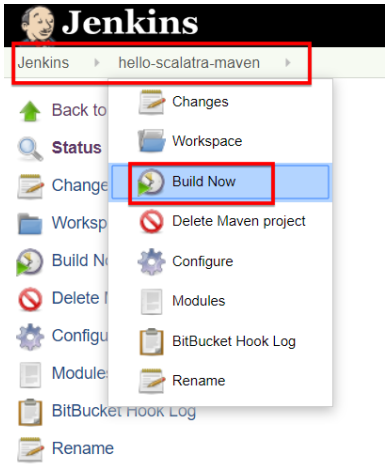


Scroll down to build triggers and select the box for "Build when a change is pushed to Bitbucket".

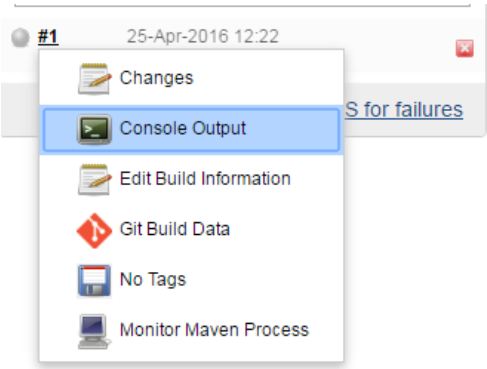


Click **"Save"** to finish.

Manually build the project by clicking "Build now" from the top bread-crumb menu.



You can see the log files by hovering over the build number and clicking the small downward arrow to the right, then selecting "Console output".



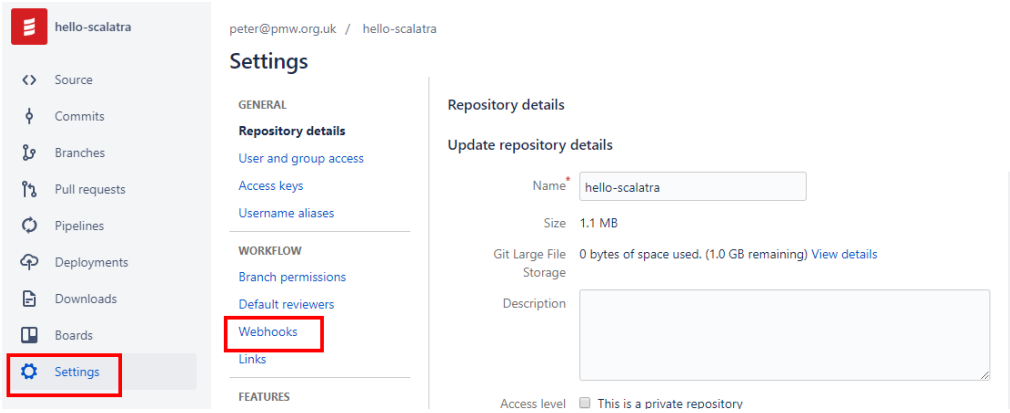
Example output:

```
[INFO] ------------------------------------------------------------------------
[INFO] BUILD SUCCESS
[INFO] ------------------------------------------------------------------------
[INFO] Total time:  03:33 min
[INFO] Finished at: 2019-05-04T14:32:58Z
```

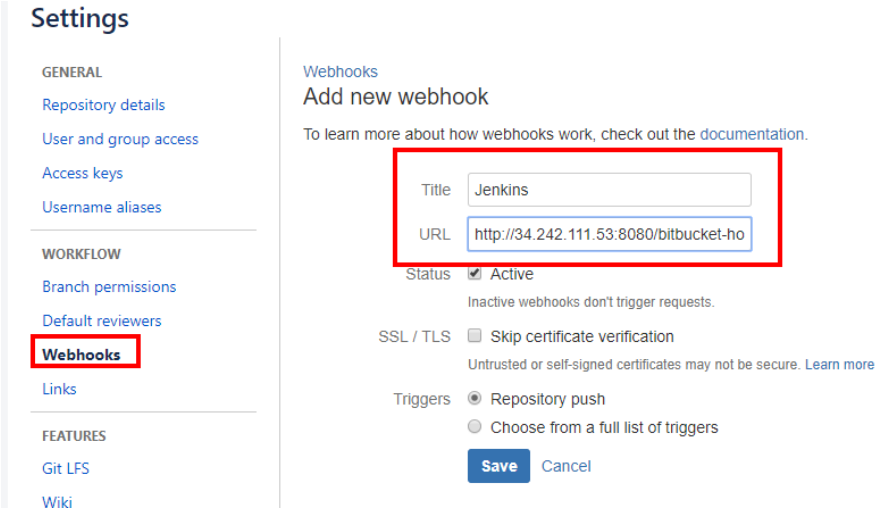
Bitbucket:

Now we have setup the listening side we need to configure Bitbucket to push a notification to Jenkins on a new commit.

1. Go to your Bitbucket repository. Select your “hello-scalatra” repo. Click on the settings icon on the left of the screen.



2. Click on the webhooks menu option



3. Add a new webhook. Call it Jenkins and add the URL in the form:

   http://[jenkinsip]:8080/bitbucket-hook/

   The end ‘/’ is very important. If you forget this then it will not work.

Test the webhook on the maven EC2 Machine.

4. Go to your cloned copy of the hello-scalatra project on the maven machine. We can add a comment into one of the Java or Scala files by starting to line with `//`. Or edit the following file:

```
cd ~
sudo vi hello-scalatra/src/main/scala/com/pmw/MainServlet.scala
```

Add a comment to the main servlet class, for example:

```
package com.pmw
import org.scalatra.ScalatraServlet
import org.scalatra.scalate.ScalateSupport

//File edited by ...

class MainServlet extends ScalatraServlet with ScalateSupport {
  before() {
    contentType = "text/html"
  }
  get("/") {
    layoutTemplate("/WEB-INF/templates/views/index.ssp")
  }
  get("/dinosaur") {
    layoutTemplate("/WEB-INF/templates/views/dinosaur.ssp")
  }
}
```

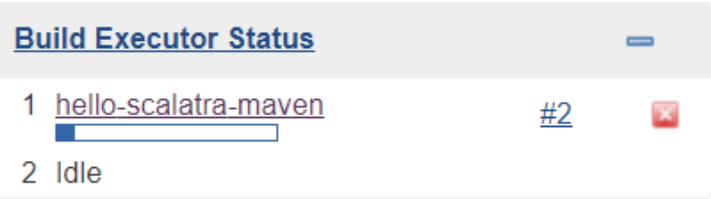
5. Move back to the hello-scalatra directory. Commit your file to git and push it to the repo.

```
$ cd ~
$ cd hello-scalatra
$ git add .
```

```
$ git status
$ git commit -m 'changes to the message'
$ git status
$ git push origin master
# enter your bitbucket password, if repo is private
```

6. Watch to see if Jenkins picks up the change and builds your new file!

Example from the Build Queue:

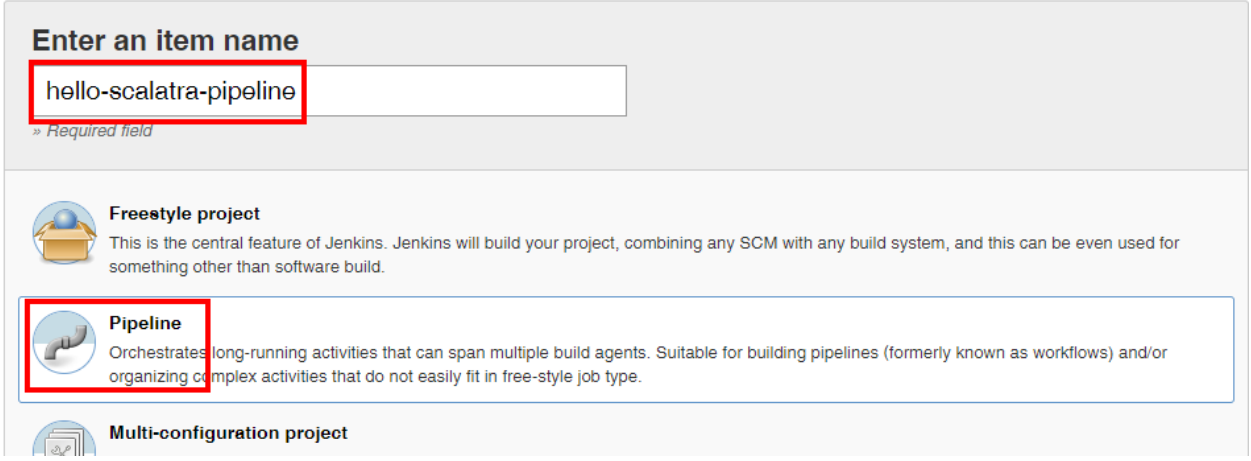


Examine the Bitbucket Hook Log.

Part 2: Create a Pipeline Project

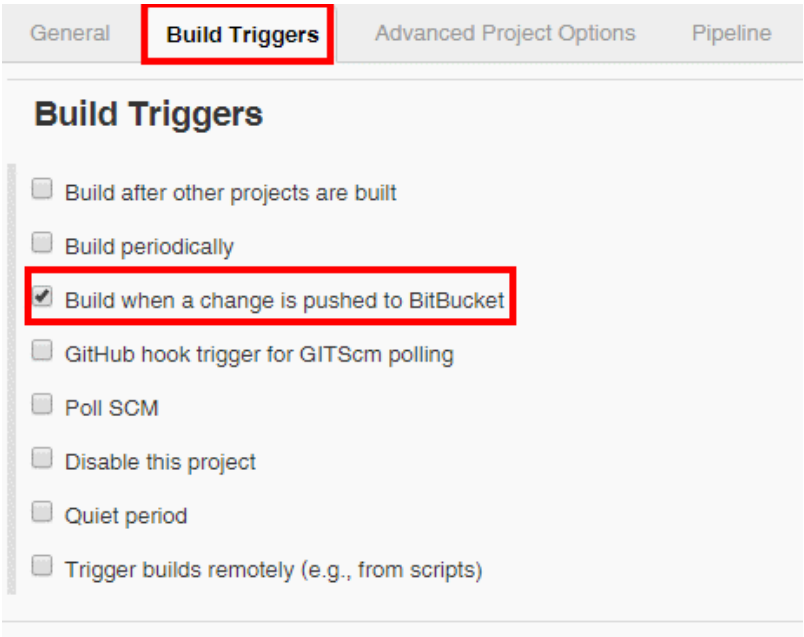
Jenkins 2.0 comes with a new set of options for pipeline projects - this is to tie in with existing DevOps pipelines.

From the dashboard create a new item. Call this one hello-scalatra-pipeline and select the pipeline project type.



Select **"OK".**

Under Build Triggers select the checkbox for build when a change is pushed to bitbucket.



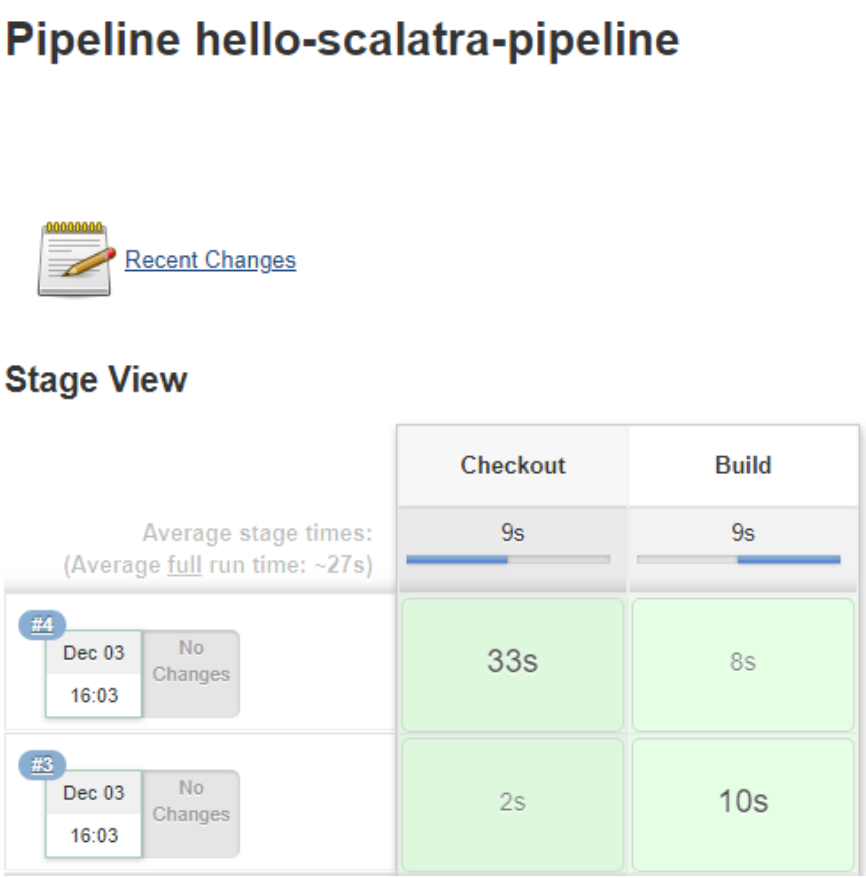
In the pipeline script box, we need to write a script. Enter the code below and change the git line to use your Bitbucket project. Then click **“Save”.**

Your sample script to look like this:

```
node {
   // Mark the code checkout 'stage'....
   stage 'Checkout'

   // Get some code from a GitHub repository - remove the
username@ portion of the url, if your repo is public
   git url: 'https://username@bitbucket.org/username/hello-
scalatra.git '

   // Get the maven tool.
   // ** NOTE: This 'M3' maven tool must be configured
   // **       in the global configuration.
   def mvnHome = tool 'M3'

   // Mark the code build 'stage'....
   stage 'Build'
   // Run the maven build
```

```
    sh "${mvnHome}/bin/mvn clean install"
}
```

Click build now to see your project build and the pipeline stages complete.



Part 3: Deploy with a post-build action

The **hello-scalatra-maven** project can be deployed to a tomcat sever. Jenkins can take care of this for us. In this part of the exercise will look at configuring and deploying to a locally based server.

1. Install tomcat on your Jenkins machine.

   On the command line type the following:

```
$ sudo yum install tomcat7-webapps tomcat7-docs-webapp tomcat7-admin-webapps -y
```

2. We need to tell Tomcat to run on a different port to Jenkins, to do this we need to edit the /etc/tomcat7/server.xml file

```
$ sudo vi /etc/tomcat7/server.xml
```

Change all instances of port 8080 to 9090

3. We also need to add a new tomcat user who is able to deploy the WAR file. Edit the **/etc/tomcat7/tomcat-users.xml** file and add the following before the …**</tomcat-users>** closing tag

```
$ sudo vi /etc/tomcat7/tomcat-users.xml

<user name="deployer" password="deployer" roles="manager-
script,manager-gui" />
```

```
in-script,manager-gui,manager-script,manager-jmx,manager-status" />-->
<user name="deployer" password="deployer" roles="manager-script,manager-gui" />
</tomcat-users>
```

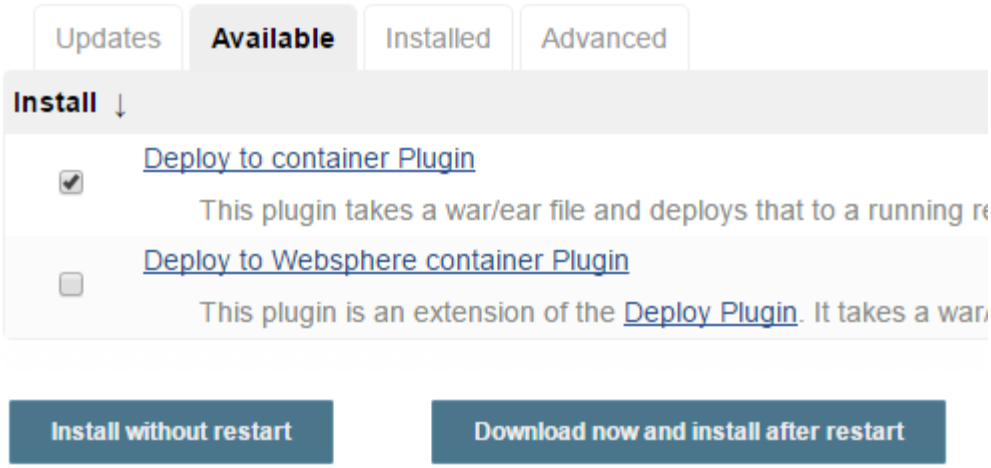
4. Restart Tomcat

```
$ sudo service tomcat7 stop

$ sudo service tomcat7 start

$ sudo fuser -v -n tcp 9090 #check if it is running
```

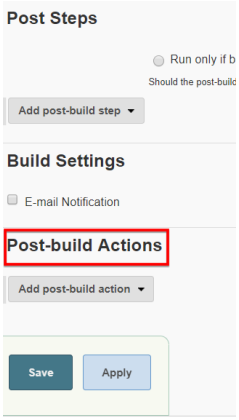
5. Verify you can access the tomcat manager gui at see the tomcat 7 splash screen on **http://[jenkinsip]]:9090/manager** (your username and password are both deployer, as set above)

Once Tomcat is set up then can tell Jenkins how to deploy to it; to do this we need another plugin.

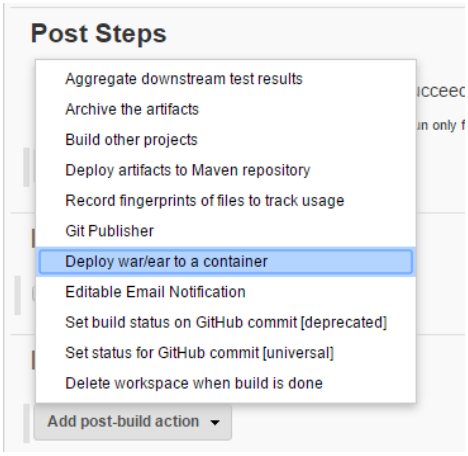
6. Install the “Deploy to Container” plugin. You want to download and install after restart. Remember to check the box to restart the server.



7. Go to your hello-scalatra-maven project and click configure. Scroll down to post-build action area.

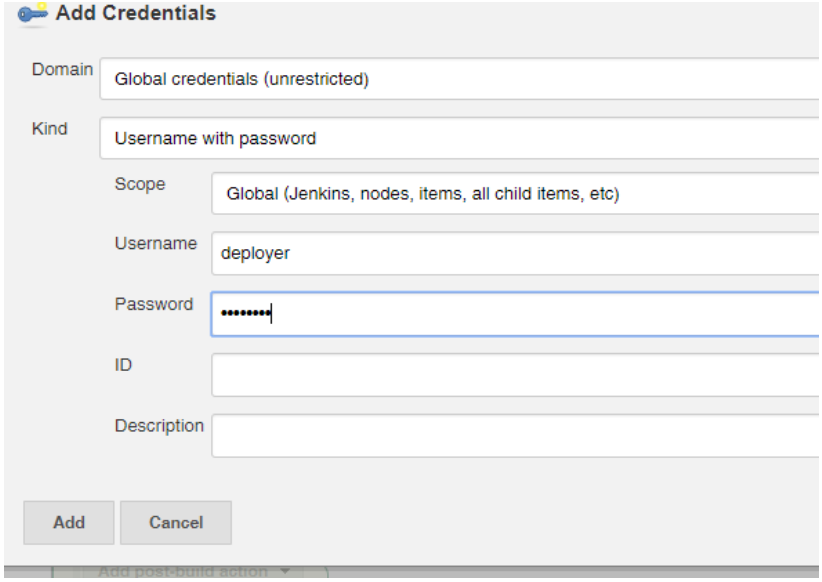


8. Add a new **post-build action** – “deploy war/ear to container”

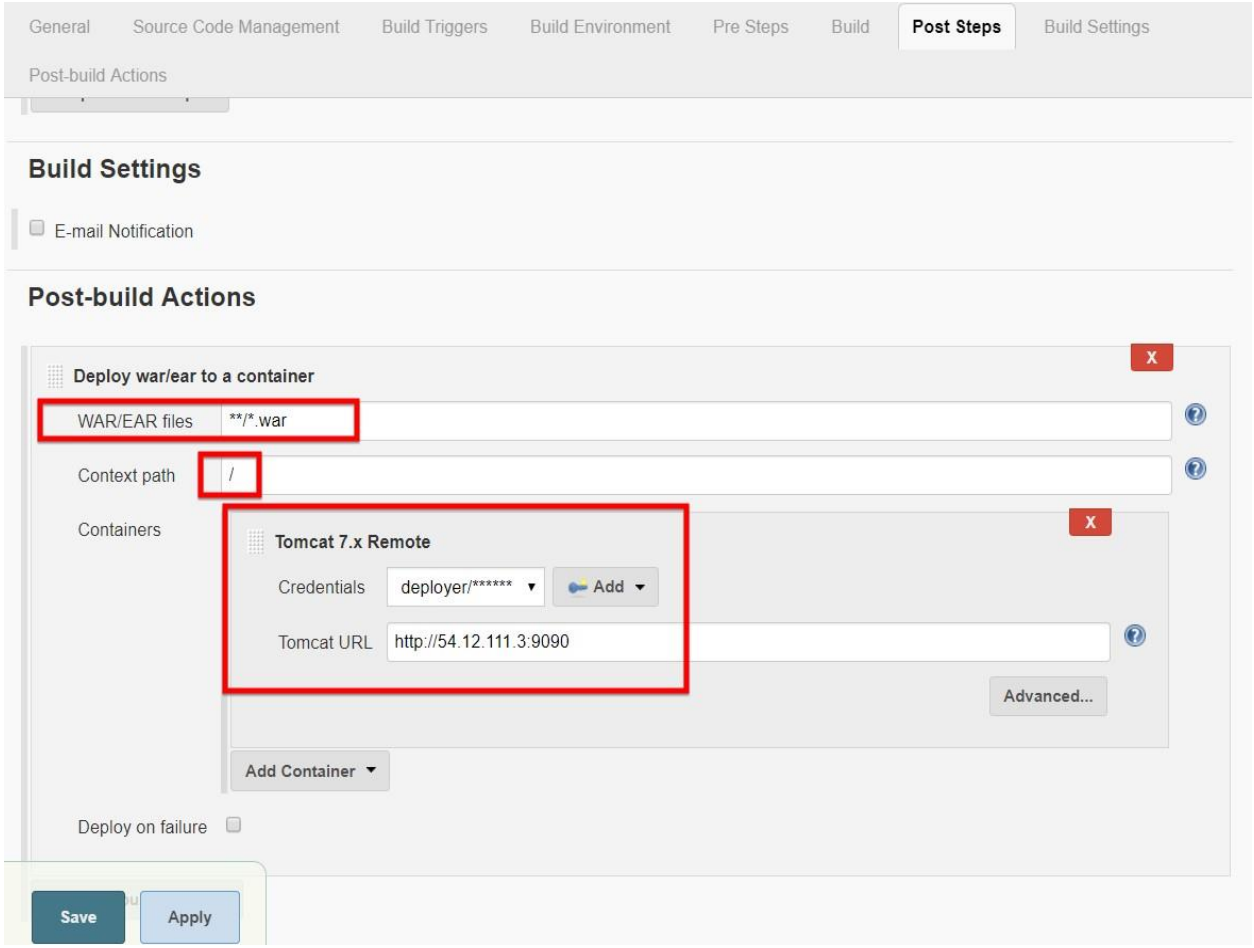


9. Under WAR/EAR file put: ****/*.war**

10. Click “Add Container” and select Tomcat 7.x
11. Click Add to add a credentials. The username and password are both “deployer”



12. The Tomcat URL is your Jenkins server on port 9090.
13. The finished setup should look similar to this:



14. Click save and start a build on **hello-scalatra-maven** project. When it completes you should be able to go to http://[jenkinsip]:9090/hello-scalatra/ to see your deployed project.