Hewlett Packard Enterprise

Codar

Software version: 1.70

Continuous Integration, Deployment and Testing by Codar using ALM and Jenkins

Document release date: July 2016

Software release date: July 2016

Contents

What is Codar?

Codar is a continuous delivery solution that provides deployment and release management of complex multi-tier applications across the application lifecycle. It automates the deployment of applications by embracing existing content from Chef, HPE SA, and so on and representing this content as components. These components can be used in a graphical topology designer to create an application model.

One of the important features of Codar is that the model is used to trigger deployments automatically with Jenkins, trigger test cases that are on the deployed instances in ALM (Application Lifecycle Management), and update the results in ALM.

This document provides information about integrating Codar with ALM.

Why is Codar required?

Software engineering builds are subject to continuous deployment and testing on the principles of frequent code commits, build automation, faster and frequent builds, automated application deployment, and test automation. On top of continuous integration, software development teams also continuously deliver qualified software applications to their test and production teams. One of the challenges that most software development teams face in the process of continuous integration and continuous delivery is the ability to automate the deployment of applications in a simple and consistent manner and run tests on the deployed instance. Codar is built to solve this problem.

Using Codar, users can deploy the application and run tests automatically by integrating with ALM. This white paper describes how Codar can be integrated with ALM. Codar is integrated with ALM through Jenkins. Jenkins acts as orchestrator between Codar and ALM.

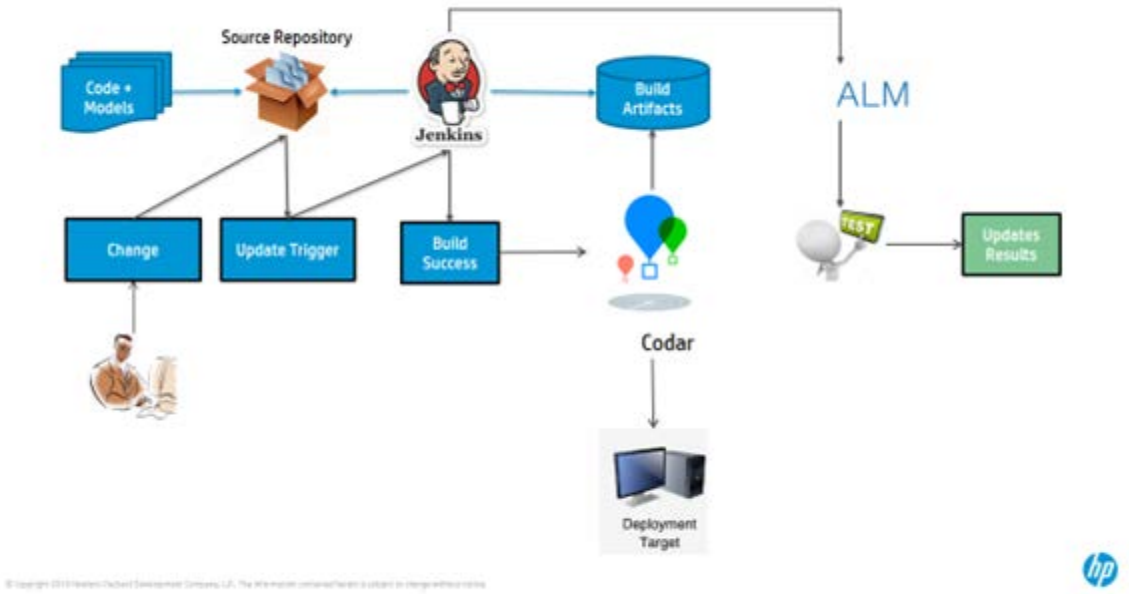
HPE Application Lifecycle Management

ALM is a set of software products designed to accelerate the delivery of secure, reliable, and modern applications. It is a combination of a common platform, several key applications, and a dashboard targeted at managing the core lifecycle of applications.

End-to-end flow

Figure 1 shows the end-to-end to flow of how Codar and ALM are integrated through Jenkins. In this use case, Jenkins is the orchestrator.

Figure 1: End to end workflow of the Codar-ALM integration



The developer makes a code change and checks it into source control. The source control system sends an update trigger and Jenkins triggers a build. After the build is successful, Jenkins invokes the Codar plugin which in turn invokes the Codar API and triggers a deployment.

After the deployment is complete, Jenkins creates a JSON file that contains the details of the deployed instance. It then invokes the ALM plugin. The ALM plugin logs on to ALM with the details provided during configuration. It invokes the test case and updates the result in ALM.

Continuous integration and deployment using Codar and Jenkins (build tool)

Software required for the integration

Jenkins must be integrated with Codar for the ALM integration to work. To integrate Jenkins with Codar, you must install the following software:

1. Install Jenkins from jenkins-ci.org/
2. Install the JDK version 1.7x on the Jenkins server.
3. Install Collabnet Subversion Edge from collab.net/support/documentation
4. Install TortoiseSVN from tortoisesvn.net. Install the latest version and use the default settings. After the installation, you can see new options when you right-click a file or folder in Windows Explorer.
5. Download and install Maven from maven.apache.org

Configuring Jenkins

After installing the software listed at Software required for the integration, configure Jenkins as follows (the following steps are for Jenkins version 1.583):

1. Ensure that JDK and Maven are installed.
2. Open Jenkins and click the **Manage Jenkins** option in the Jenkins dashboard.
3. Click **Configure System**.
4. In the **JDK** section, click **JDK installations** and then **Add JDK**.
5. Enter the name and path of the JAVA_HOME environment variable.
6. Deselect the **Install automatically** check box.
7. In the Maven section, click **Maven installations** and then **Add Maven**.
8. Enter the name and path of the MAVEN_HOME environment variable.
9. Deselect the **Install automatically** check box.
10. Enter the value of the MAVEN_OPTS environment variable.
11. Click **Save**.

Uploading the Codar plugin on the Jenkins server

You must now upload and enable the Codar plugin on the server in which Jenkins is installed.

Installing the Codar plugin

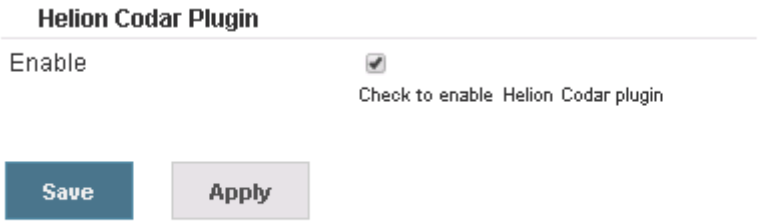
1. Log on to the Jenkins dashboard using the http://<host>:<port>/ URL. Use the host and port information appropriate for your Jenkins environment.
2. Click **Manage Jenkins** on the Jenkins dashboard.
3. Click **Manage Plugins**.
4. Select the **Advanced** tab.
5. In the **Upload Plugin** section, browse to the path of the Codar plugin file at HPE\Codar\CSAKit-4.2\Content Archives\topology\Jenkins plugin\HP_Helion_Codar.hpi
6. Click **Upload**.

Enabling the Codar plugin

1. Click **Manage Jenkins** on the Jenkins dashboard.
2. Click **Configure System**.
3. Scroll down to the **HP Helion Codar Plugin/ HP Codar Plugin** section and select the **Enable** check box.
4. Click **Save**.

Note: For security reasons, you must configure the Codar plugin with SSL and HTTPS enabled and with TLS Protocol Version 1.2 only.

Figure 2: Enabling the Codar plugin

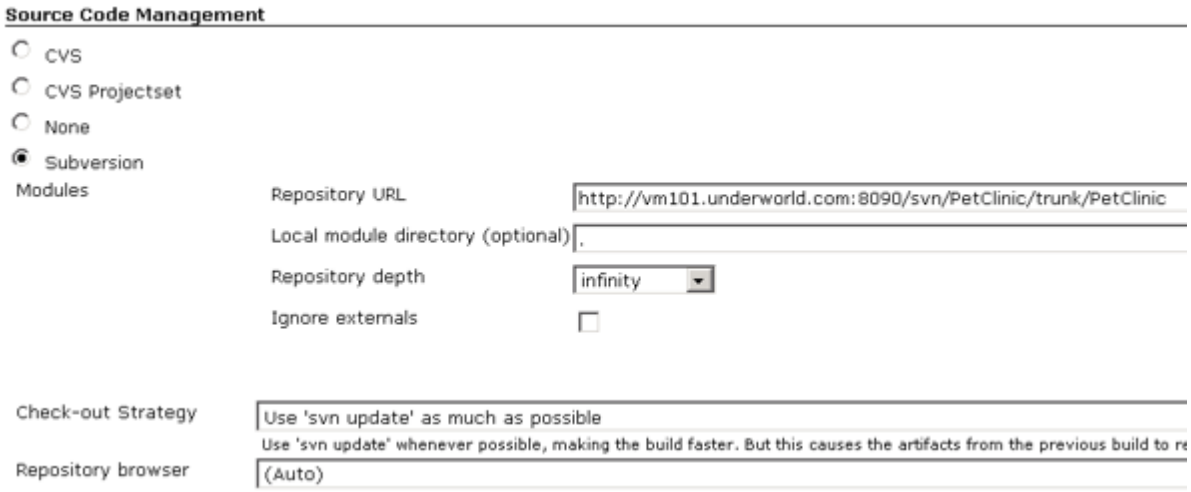


Configuring the Pet Clinic sample application project

This section describes how to configure the Codar-ALM integration by means of a sample application called Pet Clinic.

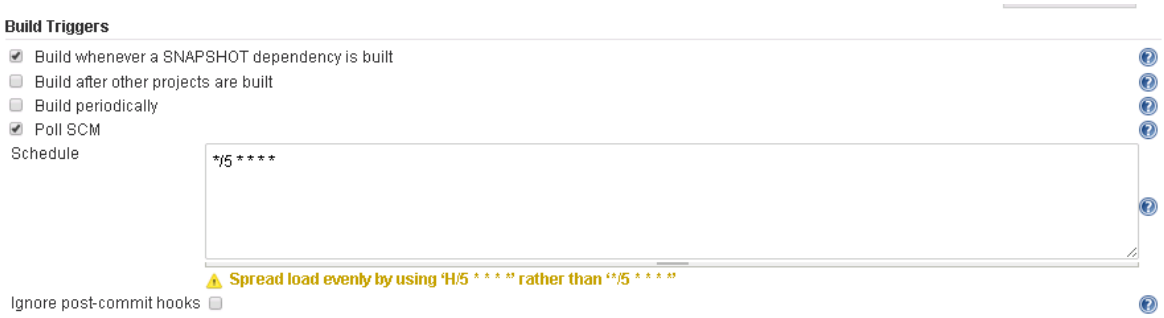
1. Download the Pet Clinic source code from GitHub.
2. Check in the source code for the Pet Clinic project into the SVN server.
3. Create a new Pet Clinic project in the Jenkins dashboard:
   a. Click **New Item** -> **Maven project**.
   b. Enter Pet Clinic in the **Item name** text box.
   c. Click **OK**.
   The Pet Clinic link is displayed in the Jenkins dashboard.
4. Click the Pet Clinic link on the Jenkins dashboard, and then click the **Configure** link in the page that opens.
5. Configure SVN for the Pet Clinic project by choosing the **Subversion Modules** option in the **Source Code Management** section area and adding the SVN Pet Clinic source code URL in the **Repository URL** field.
6. After saving, update the SVN credentials as shown in Figure 3.

Figure 3: Source code management

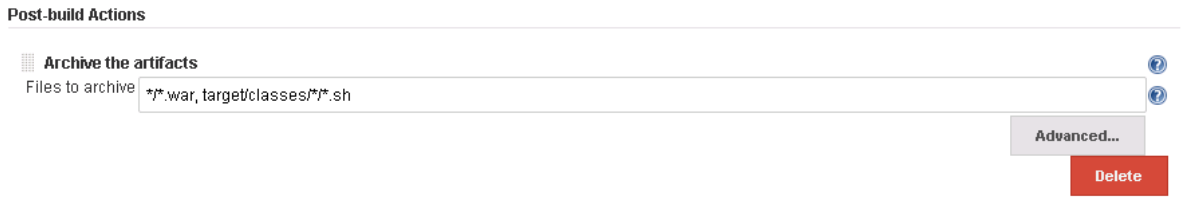


7. Configure Jenkins to automatically trigger a build if some code is checked in by selecting the **Poll SCM** check box and adding */5 * * * * in the **Schedule** field. This indicates the poll every five minutes if a code commit happens.

Figure 4: Triggering a build



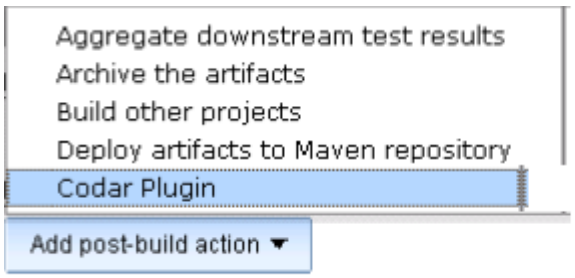
8. Scroll down to the **Post-build Actions** section and click **Add post-build action**, select **Archive the artifacts**, and then enter */*.war, target/classes/*/*.sh in the **Files to archive** text box.



Configuring the Codar plugin for the Pet Clinic sample application

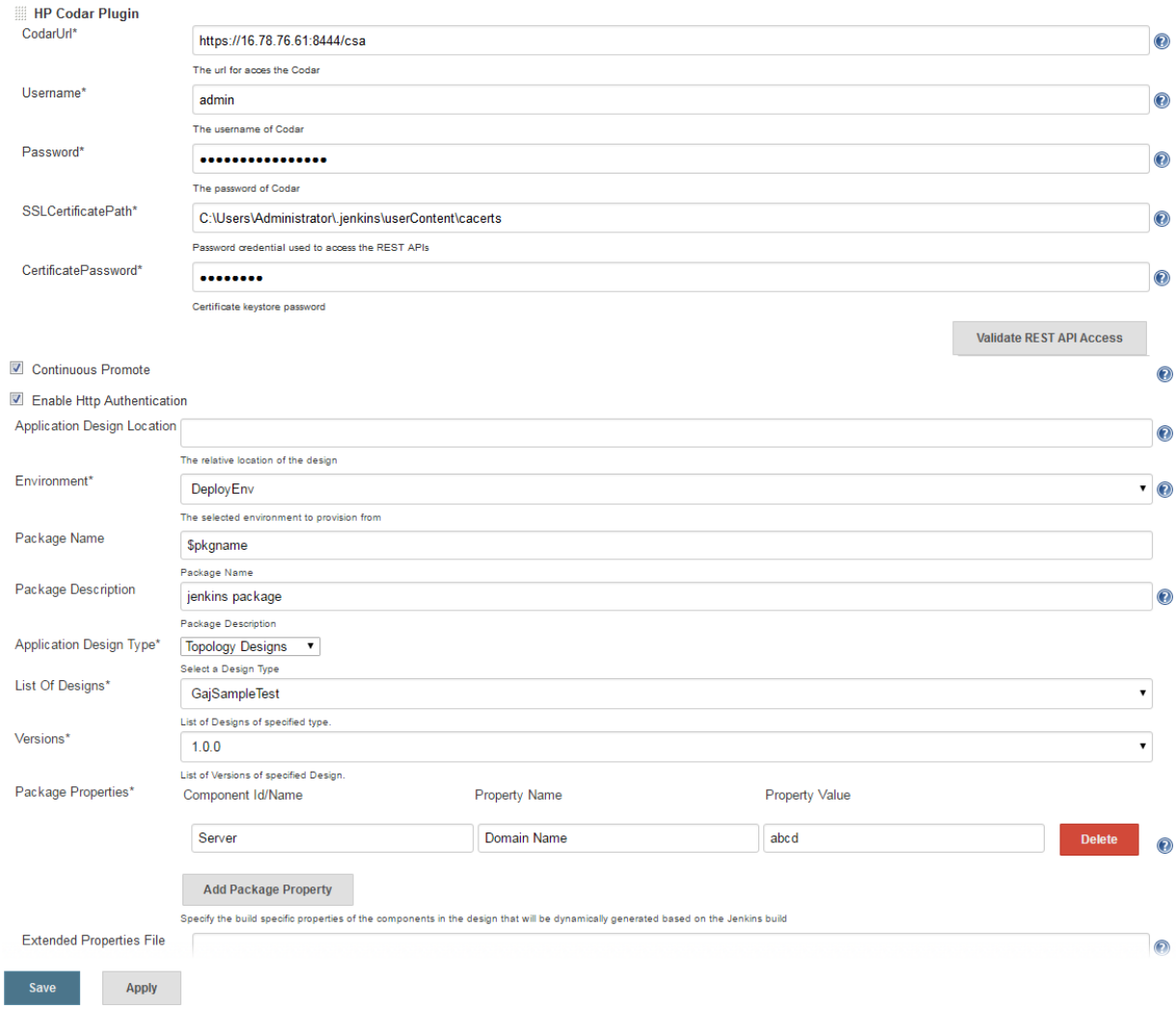
1. Click the Pet Clinic link on the Jenkins dashboard, and then click the **Configure** link on the page that opens.
2. Click **Add post-build action** and select **Codar Plugin**.

Figure 5: Configuring the Codar plugin



3. Enter the Codar plugin properties as follows:

   Figure 6: Codar properties



a. CodarUrl – URL used to log on to Codar.

b. Username – Name of the user that has administrative privileges in Codar.
Do not use the default administrator user because it may cause a security issue. After installing Codar and configuring LDAP, a user is added to the Application Architect role. Use the credentials for that user here.

c. Password – Password of the Codar user.

d. SSLCertificatePath – Enter SSL certificate path for Codar and pick up the certificate from the Codar setup. If open JRE is used during the Codar installation, then it is in the computer in which Codar is installed (in the Program Files\Hewlett-Packard\Codar\openjre\lib\cacerts path). Details about the JRE used during installation is located in the **csa.properties** file.

e. CertificatePassword – Enter the SSL certificate keystore password for Codar.

f. Continuous Promote – If this option is selected, packages are created in the first stage and release gate actions are executed. If all actions are successful, only then is the package promoted to the next stage and so on till it reaches the last stage. In the last stage, the package is executed after all actions are executed. If this option is not selected, then the package is created and deployed. Release gate actions are not executed and not promoted.

g. Enable Http Authentication – Select to enable the Jenkins user name and password. This is required for Codar to pull deployment artifacts from Jenkins.

h. HttpUsername - User name for accessing artifacts from the HTTP location. For example, if the artifacts are at a location in Jenkins, enter the user name of the Jenkins server.

i. HttpPassword – Password for accessing artifacts from the HTTP location. For example, if the artifacts are at a location in Jenkins, enter the password of the Jenkins server.

j. Application Design Location – this can be configured in 3 ways:

- Relative path and file name of the application design JSON file from the source repository URL. The relative path must be separated by a slash. This design will be deployed by Codar. For example, designs\PetClinicApp.json.

For example, if the source repository is https://myrepo.mydomain.com/mypetclinicapp, the application JSON can exist in a directory named `designs` and the JSON file can be created with the any name as required. The JSON file is a part of the repository in which the application source code is located.

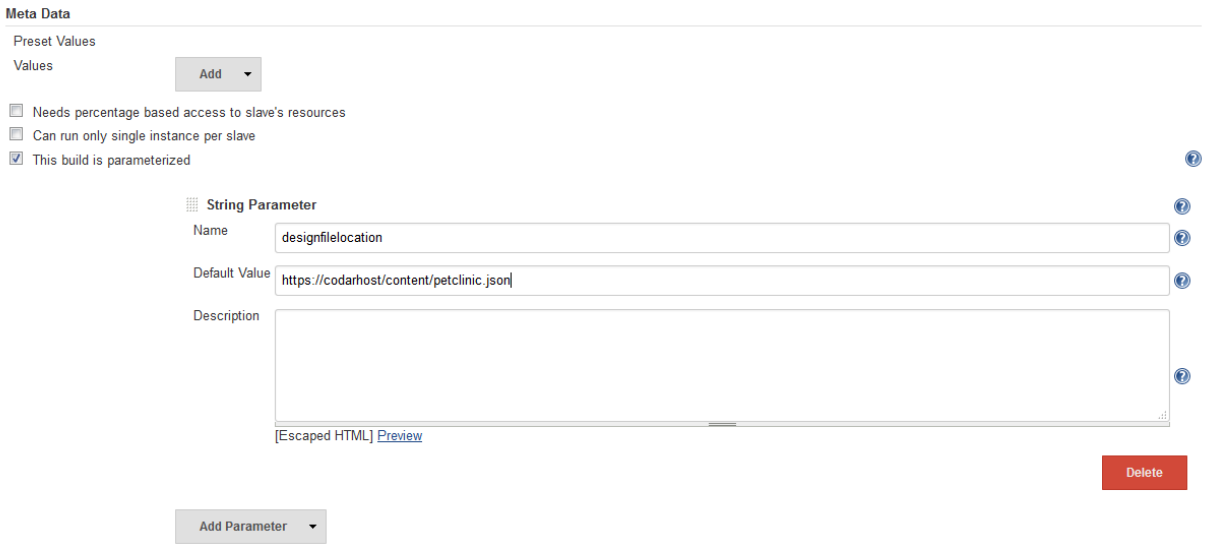
- URL of design location can be specified. For example, you can specify the location of design file **http://<hostname>/<designfile.json**>

- You can create an environment variable and pass it as a variable.

k. Application Design Type – From the list, select either Topology Designs or Sequenced Designs. If you select 'Topology Designs', then the List of Designs drop-down lists all topology designs for selection. If you select 'Sequenced Design', then the List of Designs drop-down lists all sequenced designs.

l. List of Designs – Select the design from the list. The list shows topology designs or sequenced designs based on the application design type you selected.

m. Versions – Select the version number of the design. The list shows all versions of the design selected above.



n. Environment – Environment in which the provider to be used for deployment is contained. Environments are created in the **Resource provider** tile under Codar. For details about environments, see the Codar Administration Guide.

o. Package Name – Name of the package

p. Package Description - Description of the package. The Jenkins URL is appended to the package description.

q. Package Properties – You can specify the value of the property in the following ways:

- Artifact name like petclinic.war that resolves to http://<jenkins_host>/job/Petclinic/<buildno>/artifact/target/petclinic.war
- Property value can be parameterized in one of the following ways:
- Property value – The property values allows user to specify parameterized values which allows users to dynamically reference the current build artifacts that built by the Jenkins Job
    - Within the property values users will be able to specify the Jenkins Parameters in the following notation ${Jenkins_param} where Jenkins_param is the parameter created in Jenkins. This allows the possibility of injecting the Jenkins environmental variables as part of the package property. For e.g Users can create a Jenkins String Parameter called BUILD_NUM and provide the value of this property to the environmental variable provided by jenkins $BUILD_NUMBER .Users can then specify this Jenkins Parameter ${BUILD_NUM} in the property value and this value will be assigned to the package property value. When a package is created in Codar, it will substitute the current Build number in Jenkins as the value of the property.
    - Users can also directly specify the Jenkins artifact name as depicted below. The Petclinic.war will then automatically be resolved to the http url of the Jenkins artifact, and substituted in the package property value.

For example, the package property can be:

PetClinc App	artifacturl	petclinic.war

The ID can also be used instead of the name:

PetClinic Application VERSION 1 GROUPID com.hp.csa.type0001	artifacturl	petclinic.war

Use only the package ID instead of the package name.

r. Extended Properties File –Enter the name of the properties file. This is an optional field. Specify this file only if you want to specify a different CI process than what is provided by default. The properties file can specify a different OO flow containing the necessary CI logic. You can also specify a different flow D by creating a properties file with the UUID as the key and UUID of the OO flow as the value. For example, uuid=asdaasdasdsdasdad99f. You can also specify the required properties to this flow as key value pairs in the properties file.

4. Node ID – Enter the component ID for which the component properties have to be extracted. The component name can be obtained from the topology design or from the JSON file. Multiple components names or IDs can be provided by clicking the **Add** button. For example, to retrieve the IP address and host name of the vCenter, the component name can be vCenter Server and the component ID can be: vCenterServerType__VERSION__04.20.0000__GROUPID__com.hp.csa.type0002

   The following is an example of the extended properties file of the Pet Clinic application:

   ## Properties accessed by the ARA API to invoke OO flows.

   ##This properties file contains the oo flow id (uuid) as well as the relevant parameters to be passed to the oo flow.

   #uuid of the oo flow. This flow contains the necessary logic for the Continuous integration process.

   #uuid=377898bc-d92e-4e6a-b542-718539fdcb9a

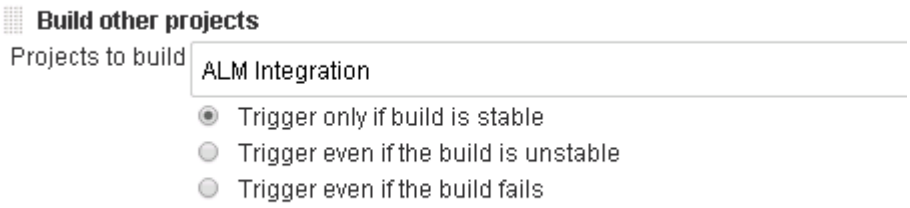
   #user can add more parameters in key value format

Adding another project

Add a step to build a downstream project (ALM Integration) as shown in Figure 7:

1. Open the Pet Clinic job in Jenkins and click **Configure**.
2. Scroll down to the **Post-build Actions** section click the **Add post-build actions** drop-down list.
3. Select **Build other projects**.
4. Enter the project name (in this case, ALM Integration) in the **Projects to build** field.
   This project triggers the ALM Integration project after the Pet Clinic project is complete.

Figure 7: Building another project



Continuous testing of the application using Codar and ALM through Jenkins

Install and configure the Jenkins – ALM plugin

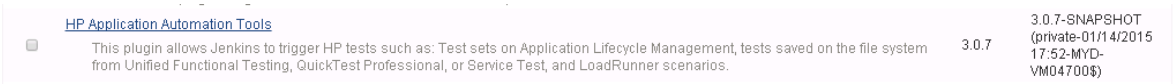
Install the Jenkins plugin for ALM from the following location:

https://wiki.jenkins-ci.org/display/JENKINS/HP+Application+Automation+Tools

Enable the Jenkins plugin for ALM

Go to **Jenkins** -> **Manage Jenkins** -> **Manage Plugins** and verify that the following entry appears:

Figure 8: Verifying the Jenkins – ALM plugin



Go to **Jenkins** -> **Manage Jenkins** -> **Configure System** and provide the details of the ALM server.

Figure 9: Providing details of the ALM server

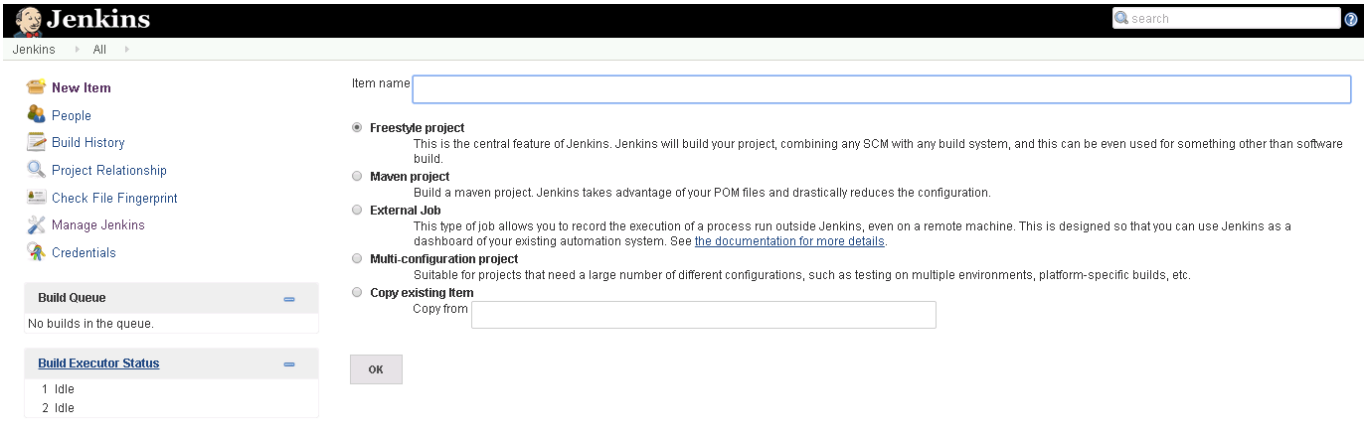
Enabling the Artifact Deployer Plugin

Install (if not available) the artifact plugin from **Manage Jenkins** –> **Manage Plugins** –> **Available** tab –> **Artifact Deployer Plugin**.

Configure the Jenkins plugin for ALM

1. *Create a new freestyle project and name it ALM Integration.*

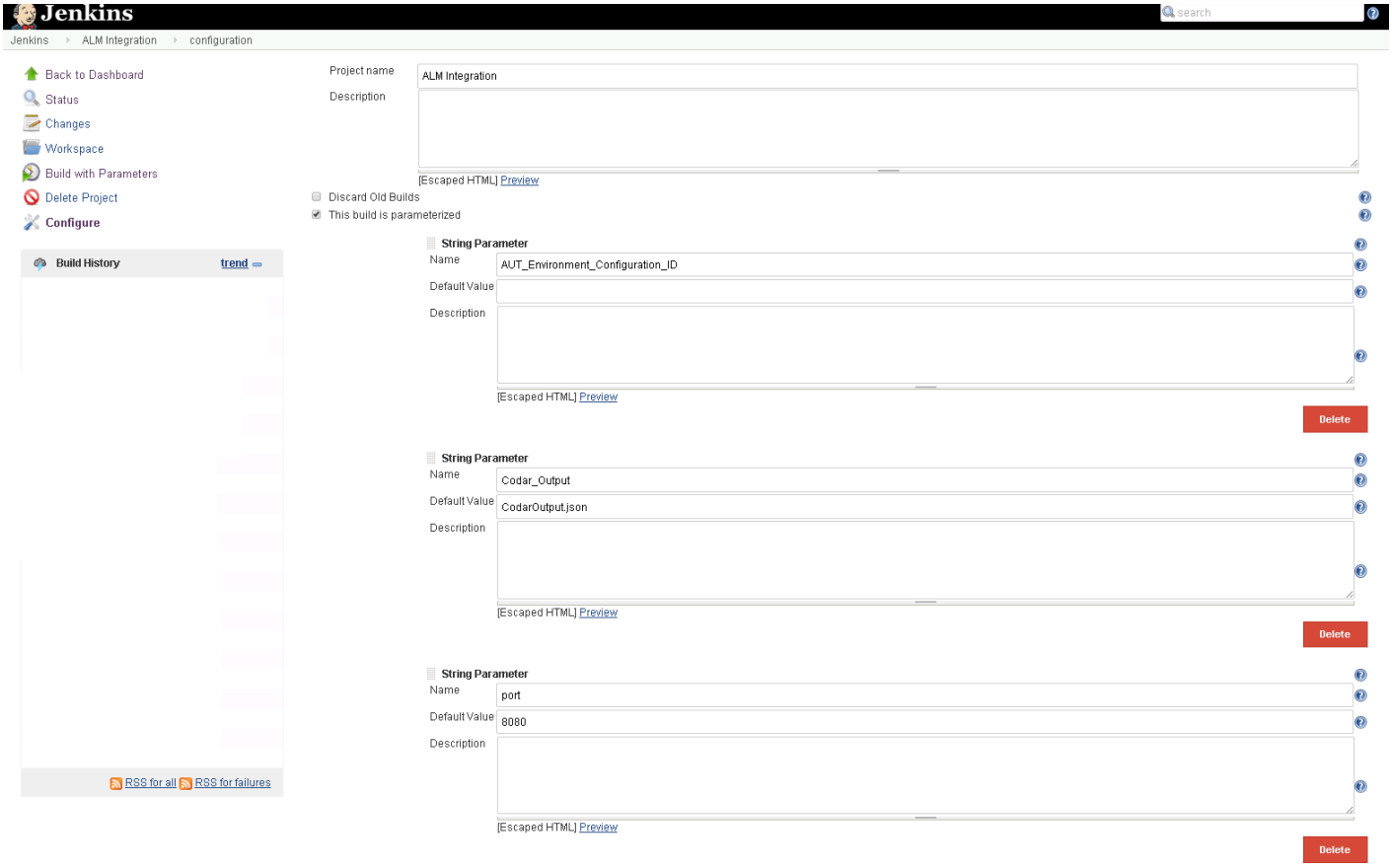
   Figure 10: Creating a project



2. Click the **Configure** link in the ALM Integration page.
3. Click the **This build is parameterized** check box.
4. Click **Add Parameter** three times and enter the details as listed in Parameter details. All parameters must be string parameters. These parameters are environment variables that are used when configuring the build steps.
   Table 1: Parameter details

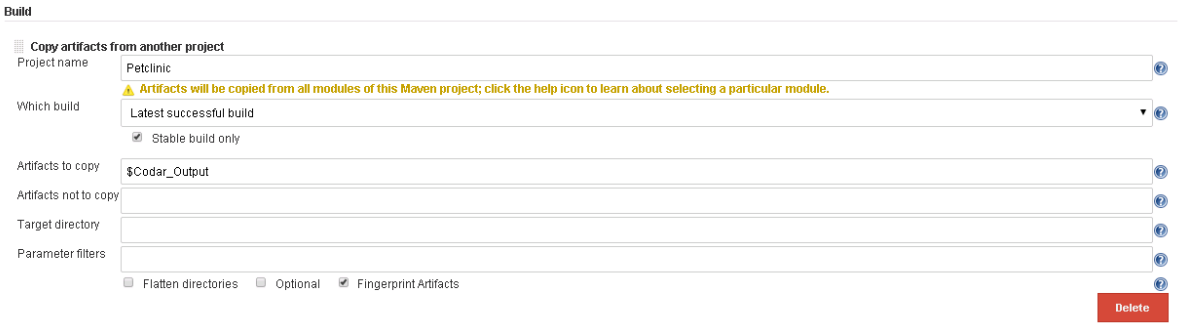
Parameter name	Default value	Description
AUT_Environment_Configuration_ID	Blank	Blank
Codar_Output	CodarOutput.json	Blank
Port	8080	Blank

Figure 11: Parameter details



5. Click **Add build step**.
6. In the **Copy artifacts from another project** section, enter Pet Clinic in the **Project name** field. This copies the output of the file that is generated to the ALM Integration job's workspace. The copied file contains information such as host name, user name and so on of the node.

   Figure 12: Configuring the values of the ALM Integration project



7. Click **Add build step**.
8. Select **Execute AUT Environment Preparation using HP ALM Lab Management**.
9. Enter the following values:

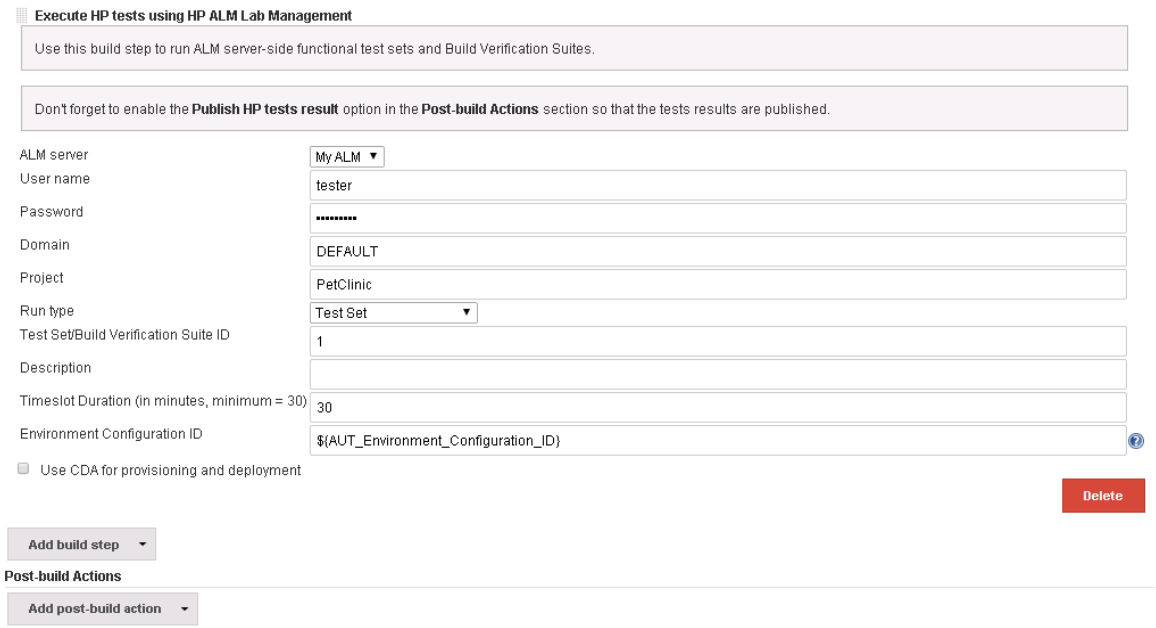
   Figure 13: Specifying values for the Execute AUT Environment Preparation using ALM Lab Management plugin

Table 2: Values of the ALM Lab Management plugin

Field	Description
ALM server	Name of the ALM plugin configuration
User name	Name of the ALM user who has permission to trigger the test suite
Password	Password of the ALM user who has permission to trigger the test suite
Domain	ALM domain name
Project	Name of the project that is associated with the domain and that which contains the test suite
AUT Environment ID	ALM environment ID
AUT Environment Configuration	Select this check box and enter a unique name with the build ID. You can see the value of the build ID from the $BUILD_NUMBER environment variable. For example, TriggeredByJenkins $BUILD_NUMBER
Path to JSON file	Path in which the JSON file is copied during the Copy artifacts from another project step.
Assign AUT Assignment Configuration ID to	Parameter that is passed to the test script. This parameter is created in the AUT environment. Enter Parameters/App Server/ipAddress
AUT Environment Parameters	Parameter type: From JSON Parameter name: Parameters/App Server/ipAddress Parameter value: $.Nodes[?(@.id==<VcenterServerType>__<VERSION>__04.20.0000__GROUPID__com.hp.csa.type0002)].properties.[?(@.id==ipAddress)].values[0] You can obtain these values from the JSON file that refers to the node component which contains the application server details.

10. Click **Add build step**.
11. Select the **Execute HP Tests using HP ALM Lab Management** plugin and enter the values. For information about these values, see the Jenkins ALM plugin documentation.

Figure 14: Values of the Execute Tests using ALM Lab Management plugin

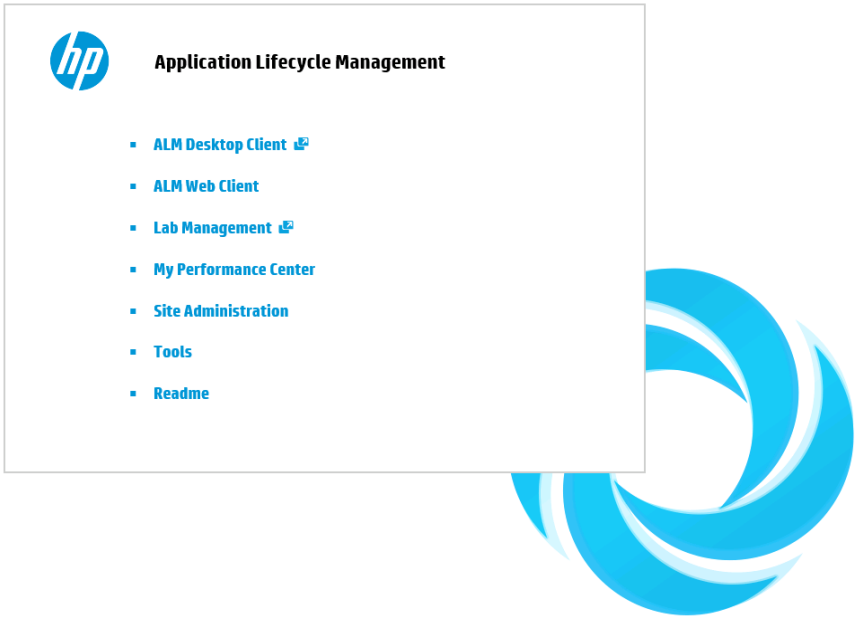


Create the test environment in ALM

The Jenkins ALM integration is supported only in ALM version 12.20. See the ALM Installation Guide for information about installing ALM 12.20.

After installing ALM, the configurations required for the Jenkins-ALM integration are shown in Figure 16.

Figure 15: Configurations required for the integration

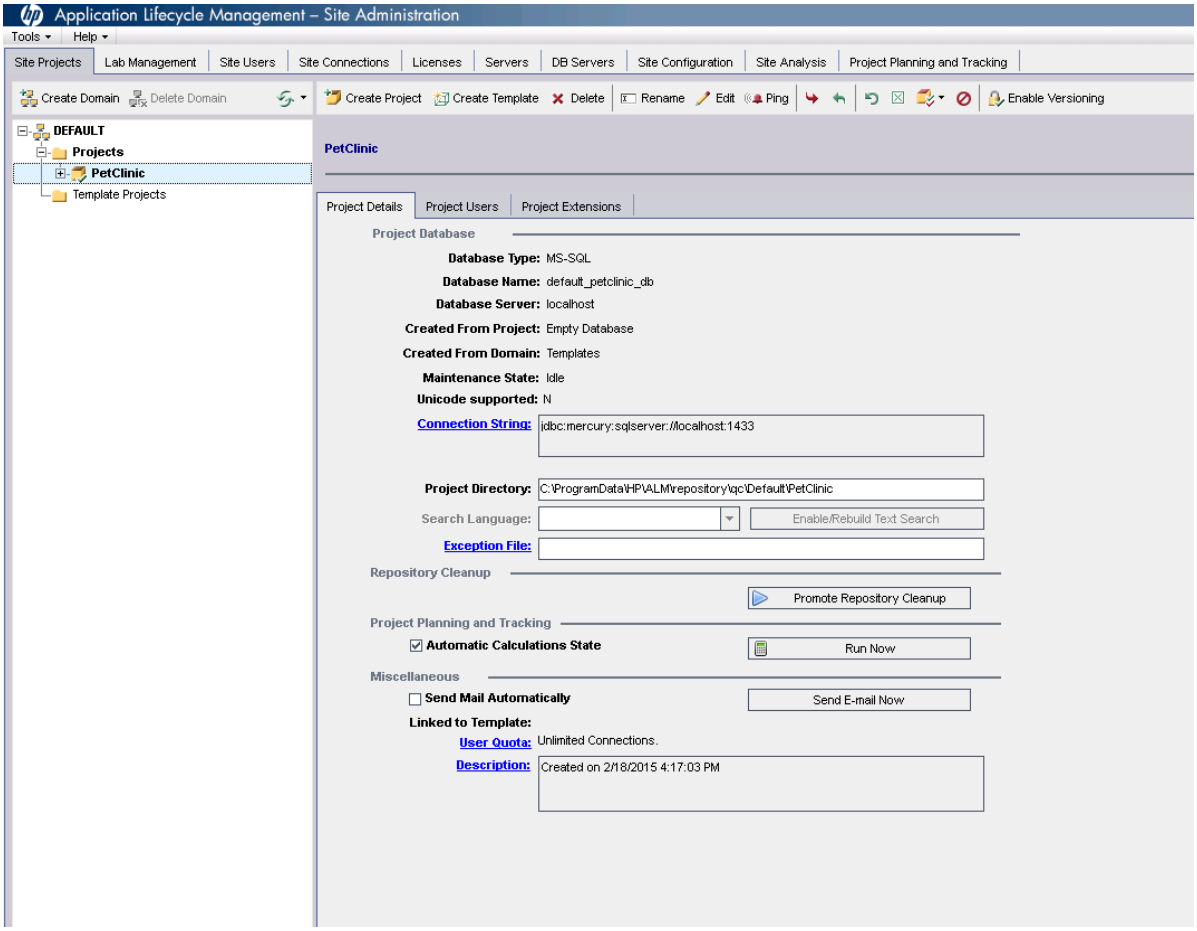


After installing ALM, open the http://localhost:8080/qcbin/ link using Internet Explorer.

Site Administration

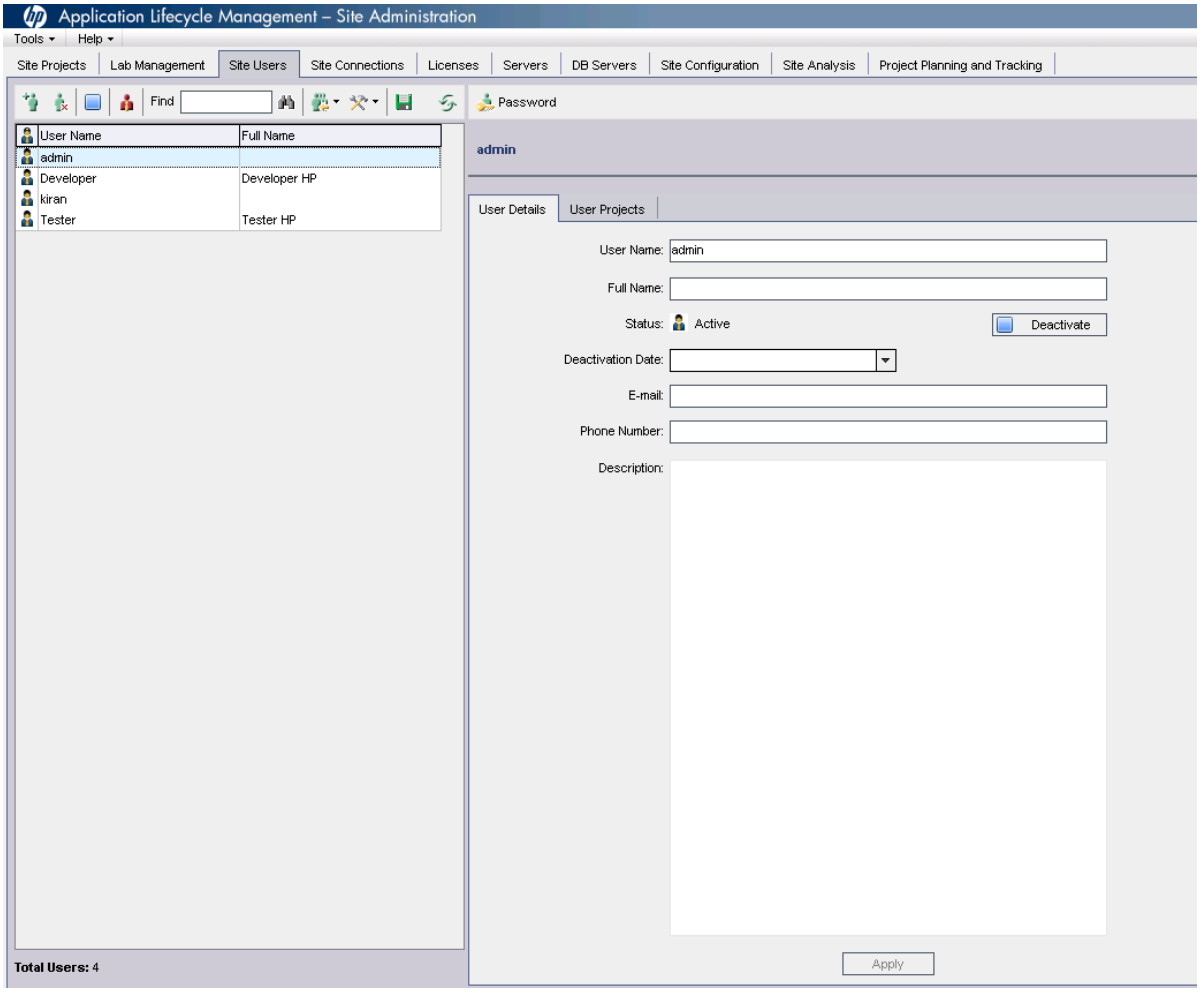
Use Site Administration to create projects and users. The users should be associated with the project. The following screens show how the Pet Clinic project is created.

Figure 16: Entering the project details



After entering the project details, users have to be created and associated with the Pet Clinic project. In this example, a user called Tester has been created and Tester has been associated with the Pet Clinic project. In this case, Tester is also a project administrator (to avoid any permission issues).

Figure 17: Creating a user



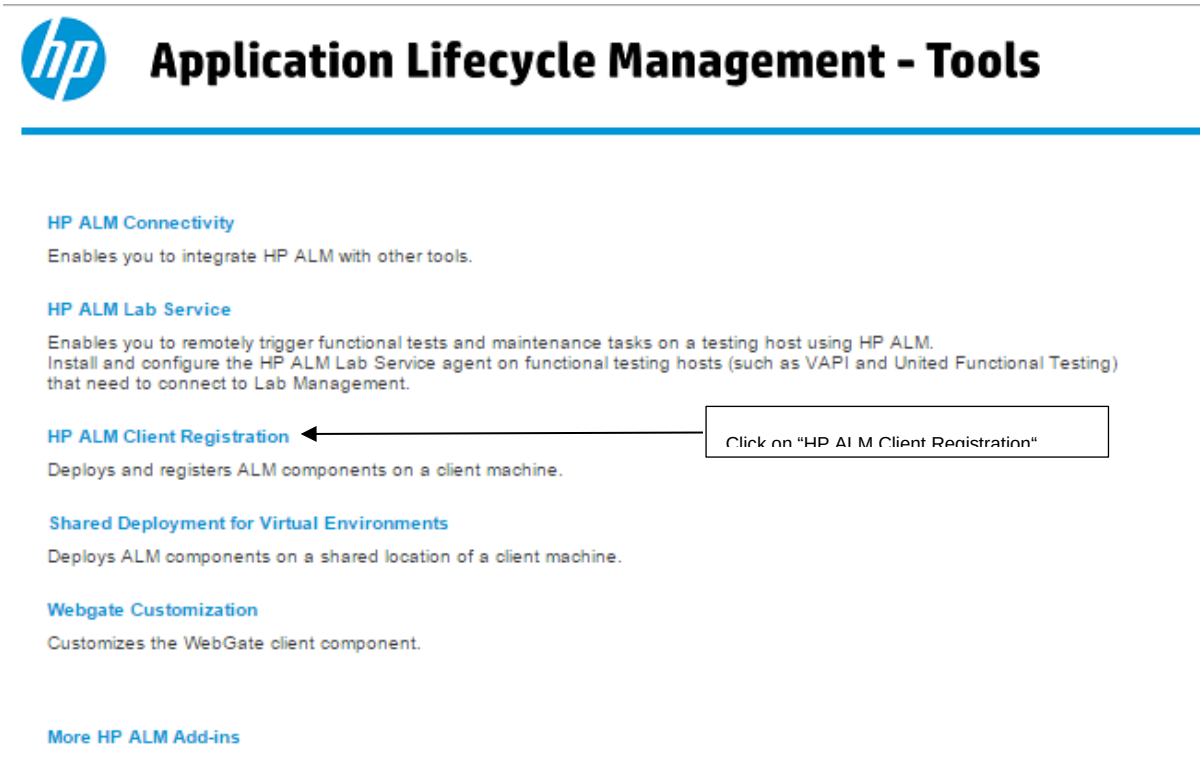
For more information about using Site Administration, see the ALM Administration Guide.

ALM Lab Service

You can install ALM Lab Service either on the same computer in which ALM 12.20 is installed or on a different computer. Ensure that ALM Lab Service is running. For information about installing and configuring ALM Lab Service, see the ALM Guide.

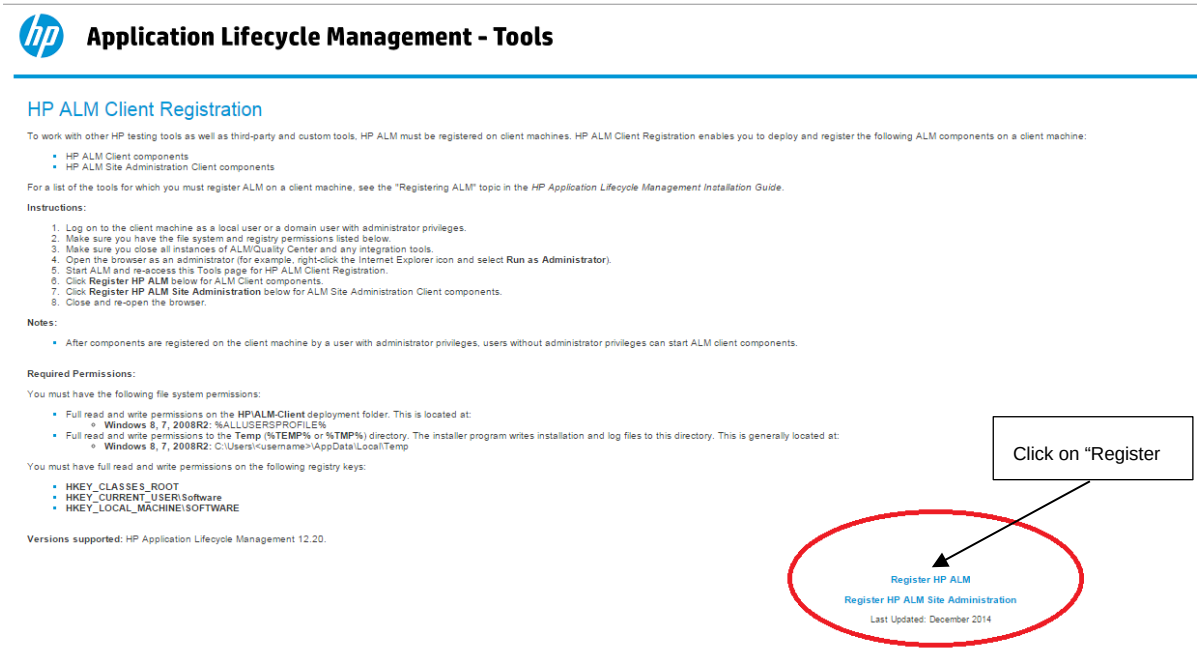
Note – If ALM Lab Service is installed on a different computer, then VAPI-XP must be installed. To do this, open Internet Explorer on the computer in which ALM Lab Service is installed and type http://<alm server hostname>:<port>/qcbin/addins.html

Figure 18: Installing VAPI-XP



To download the VAPI-XI libraries, register the client with ALM as shown in Figure 20.

Figure 19: Downloading libraries

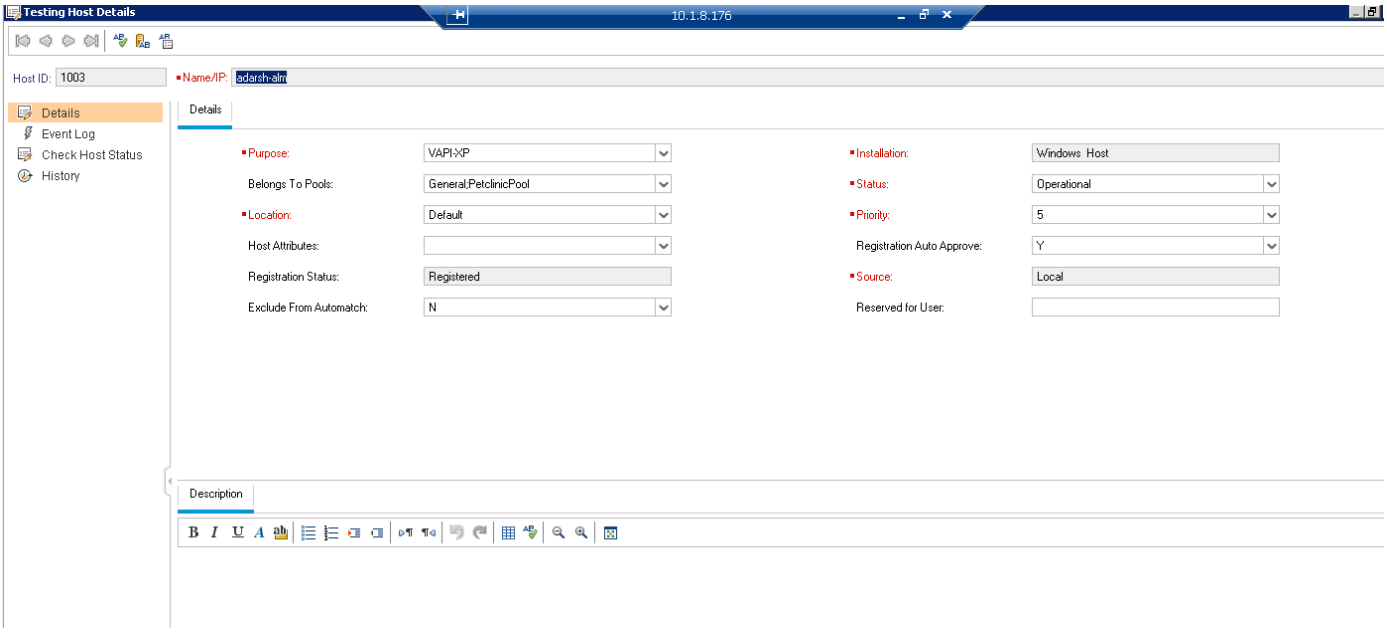


ALM Lab Management

1. Open ALM Lab Management from the computer in which ALM 12.20 is installed. Login as an administrator.

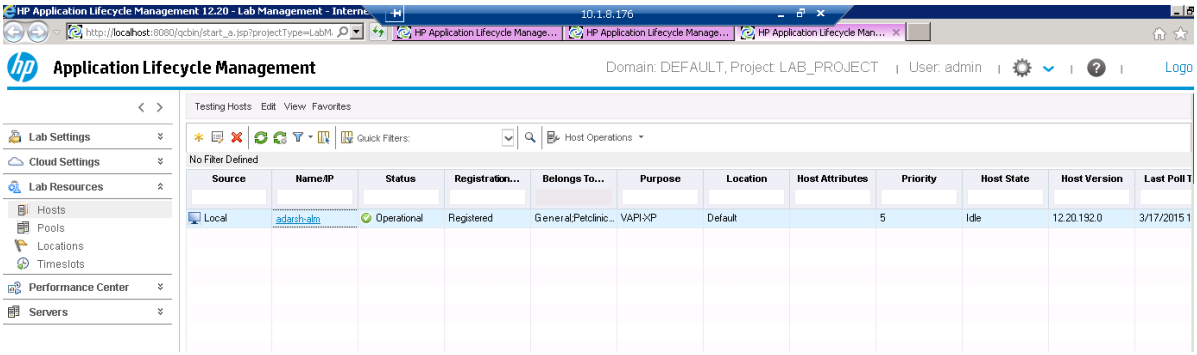
2. Go to **Testing Host** –> **New Testing host**.

Figure 20: Testing hosts



3. Ensure that the purpose is VAPI-XP.
4. After adding the new host, it must appear as shown in Figure 22.

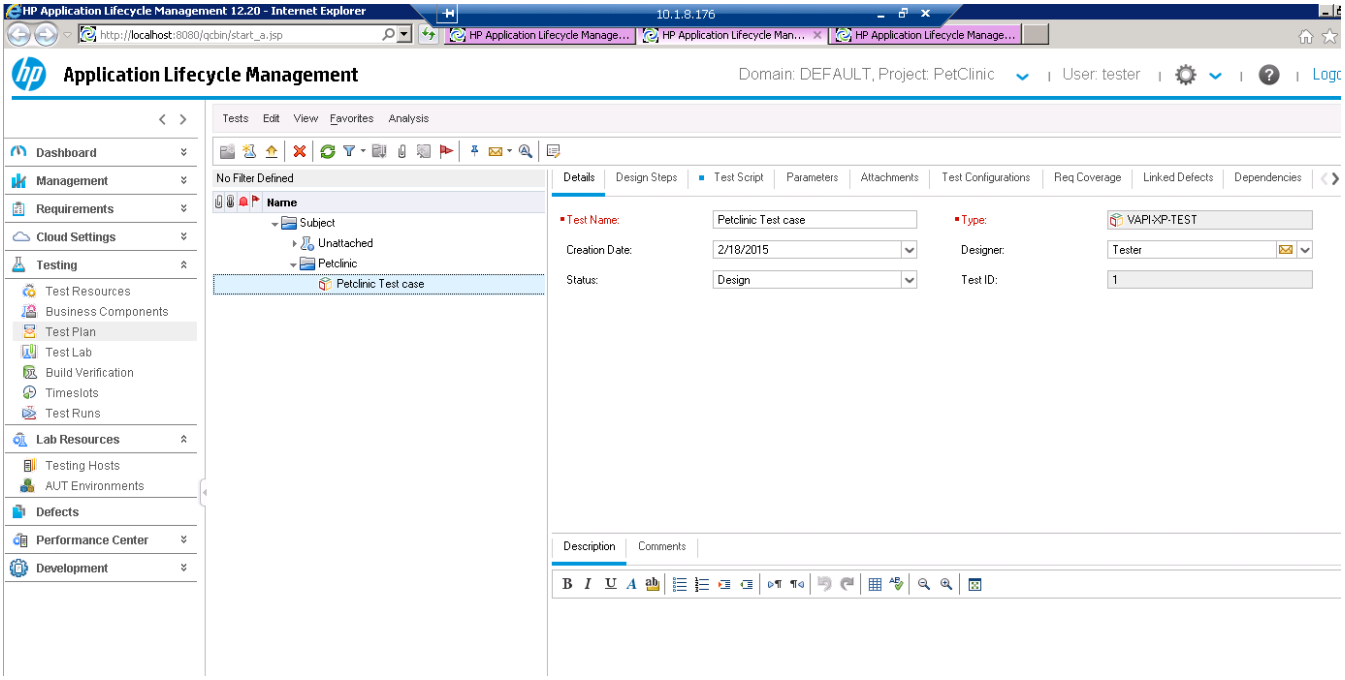
Figure 21: New host



ALM Desktop Client

1. Open ALM Desktop Client from the computer in which ALM 12.20 is installed.
2. Login as Tester (domain is default and the project is Pet Clinic for this example)
3. Go to **Test Plan** –> **Create a New Test Case**.

Figure 22: Creating a test case



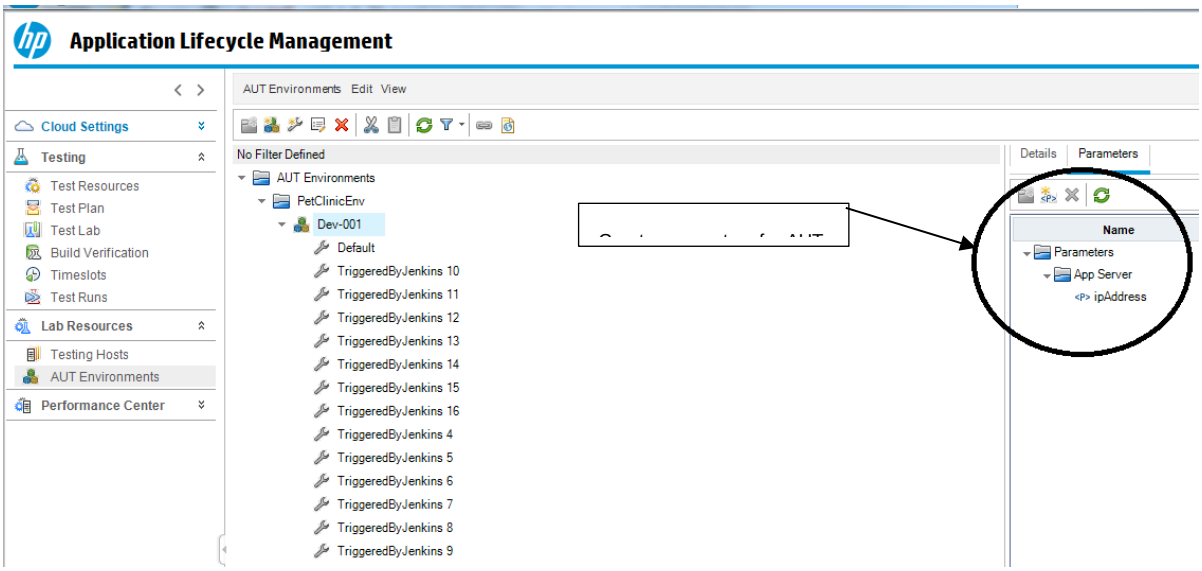
4. Ensure that the type is VAPI-XP-TEST.
5. Add the following script on the **Test Script** tab.

```
' Petclinic Test [VBScript]
' Created by Application Lifecycle Management
' 2/9/2015 3:25:28 AM
' ====================================================




' ----------------------------------------------------
' Main Test Function
' Debug - Boolean. Equals to false if running in [Test Mode] : reporting to Application Lifecycle Management
' CurrentTestSet - [OTA COM Library].TestSet.
' CurrentTSTest - [OTA COM Library].TSTest.
' CurrentRun - [OTA COM Library].Run.
' ----------------------------------------------------
Sub Test_Main(Debug, CurrentTestSet, CurrentTSTest, CurrentRun)
  ' *** VBScript Limitation ! ***
  ' "On Error Resume Next" statement suppresses run-time script errors.
  ' To handle run-time error in a right way, you need to put "If Err.Number <> 0 Then"
  ' after each line of code that can cause such a run-time error.
  On Error Resume Next


  ' clear output window
  TDOutput.Clear
```

```
  ' TODO: put your code here
      'Dim ipAdd : ipAdd = "10.1.9.136"

'url_base= "10.1.9.136"
url_base=CurrentRun.getRunTimeParameterByName("ipAddress")
url= "http://" & url_base & ":8080/petclinic"

  set ie = CreateObject("InternetExplorer.Application")
'     ie.Navigate "http://10.1.9.136:8080/petclinic"
      ie.Navigate url
     ie.Visible = true

  If Not Debug Then
  End If
  ' handle run-time errors
  If Err.Number <> 0 Then
    TDOutput.Print "Run-time error [" & Err.Number & "] : " & Err.Description
    ' update execution status in "Test" mode
    If Not Debug Then
      CurrentRun.Status = "Failed"
      CurrentTSTest.Status = "Failed"
    End If
  End If
End Sub
```

6. Ensure that Testing Hosts is updated with the information added in Lab Management.
7. Verify that the Pet Clinic test case is successful by manually running the test case in the test lab.
8. Go to **Lab Resources** -> **AUT Environments** and add a new AUT parameter called App Server as shown in Figure 24.

Figure 23: Added the App Server parameter



End-to-end flow

Now Jenkins is set up for the Pet Clinic project and the ALM_Integration project. The ALM_Integration project is configured as a dependent project for Pet Clinic. The following steps provide the end-to-end flow:

1. Jenkins pulls in Pet Clinic changes from SVN.
2. The Pet Clinic build is triggered.
3. After the build is complete, Jenkins archives the artifacts.
4. The Codar plugin that is configured as a post build action is triggered as show in Figure 25.

Figure 24: Triggering the post build action

```
Archiving artifacts
Jenkins URL: http://10.1.6.54:8088/
Codar Deploy URL: https://10.1.6.54:8444/csa/api/codar/codar-oointegration/invokeFlow
Codar Deploy Status URL: https://10.1.6.54:8444/csa/api/codar/app-deploy/
No Properties file.
JSON Input to Codar :: {
"designurl":"http://10.1.6.54:8088/job/Petclinic/29/artifact/PetclinicApp.json","environment":"De
9fa310fd__VERSION__1__GROUPID__com.hp.csa.type0001=[{artifacturl:http://10.1.6.54:8088/job/Petcli
[{vmNamePrefix:ChangedDBName_}],VcenterServerType__VERSION__04.20.0000__GROUPID__com.hp.csa.type0
Output from the Codar REST CALL ....
Received Output from the Codar REST CALL ...

{

{
  "serviceInstanceId" : "8a8186b64b11370b014bda5aa02d3612"
```

5. The Codar plugin calls an API in Codar and triggers the deployment.
6. It keeps polling till the deployment is complete as shown in Figure 26.

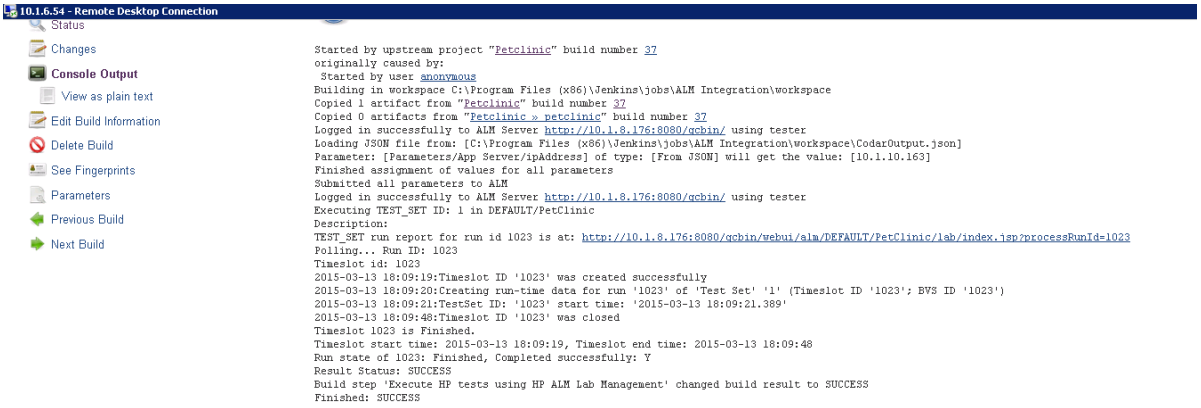
Figure 25: Deployment complete

```
Final Deployment Status :DEPLOYING
getserviceInstance from the Codar REST CALL ....   "status" : "ACTIVE",
Deployment Status :ACTIVE
Final Deployment Status :ACTIVE
Writing Codar output on this folder:C:\Program Files (x86)\Jenkins\jobs\Petclinic\builds\2015-03-02_17-27-38\archive
Codar output File Location:C:\Program Files (x86)\Jenkins\jobs\Petclinic\builds\2015-03-02_17-27-38\archive
Warning: you have no plugins providing access control for builds, so falling back to legacy behavior of permitting any downstream builds to be triggered
Triggering a new build of ALM_Integration
Finished: SUCCESS
```

7. After the deployment is complete, the status of the final deployment becomes active.

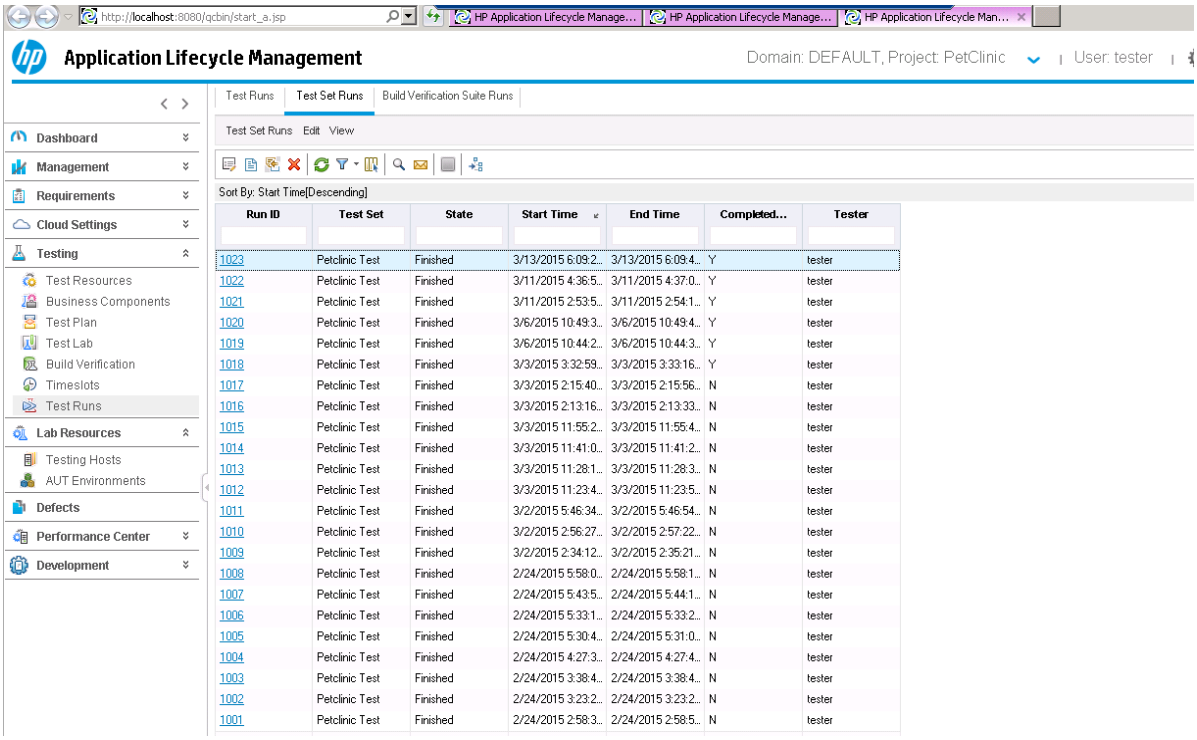
8. The Codar plugin creates a JSON file that contains the IP address, user name, password, and other details of the deployed instance. The JSON file is created based on the node ID input given during the configuration of the plugin.
9. The JSON file is stored in the workspace as CodarOutput.json. This is the integration point between Codar and ALM.
10. Because ALM_Integration is configured as a subsequent project, the ALM_Integration build job is triggered after the Pet Clinic job is successful.
11. Open the output console of ALM_Integration to see the status of that job. It must be successful as shown in Figure 27.

Figure 26: Successful job status



12. The ALM_Integration job copies the JSON file from workspace. It then reads the JSON file and looks for parameters such as IP address.
13. The ALM_Integration job logs on to ALM with the credentials provided during configuration. It triggers the test case and executes the test script.
14. The test script is executed in the computer that is configured in Lab Management. VAPI tests are executed. In this case, it opens the Pet Clinic link http://<localhost>:8080 in a browser and verifies that the application is deployed successfully.
15. The test results are updated back into Test Run as shown in Figure 28.

Figure 27: Updated test results

1. 

Send documentation feedback

If you have comments about this document, you can send them to clouddocs@hpe.com.

Legal notices

Warranty

The only warranties for Hewlett Packard Enterprise products and services are set forth in the express warranty statements accompanying such products and services. Nothing herein should be construed as constituting an additional warranty. Hewlett Packard Enterprise shall not be liable for technical or editorial errors or omissions contained herein. The information contained herein is subject to change without notice.

Restricted rights legend

Confidential computer software. Valid license from Hewlett Packard Enterprise required for possession, use or copying. Consistent with FAR 12.211 and 12.212, Commercial Computer Software, Computer Software Documentation, and Technical Data for Commercial Items are licensed to the U.S. Government under vendor's standard commercial license.

Copyright notice

Trademark notices

Adobe® is a trademark of Adobe Systems Incorporated.

Microsoft® and Windows® are U.S. registered trademarks of Microsoft Corporation.

Oracle and Java are registered trademarks of Oracle and/or its affiliates.

UNIX® is a registered trademark of The Open Group.

RED HAT READY™ Logo and RED HAT CERTIFIED PARTNER™ Logo are trademarks of Red Hat, Inc.

The OpenStack word mark and the Square O Design, together or apart, are trademarks or registered trademarks of OpenStack Foundation in the United States and other countries, and are used with the OpenStack Foundation's permission.

Documentation updates

The title page of this document contains the following identifying information:

- Software Version number, which indicates the software version.
- Document Release Date, which changes each time the document is updated.
- Software Release Date, which indicates the release date of this version of the software.

To check for recent updates or to verify that you are using the most recent edition of a document, go to the following URL and sign-in or register: https://softwaresupport.hpe.com.

Select Manuals from the Dashboard menu to view all available documentation. Use the search and filter functions to find documentation, whitepapers, and other information sources.

You will also receive updated or new editions if you subscribe to the appropriate product support service. Contact your Hewlett Packard Enterprise sales representative for details.

Support

Visit the Hewlett Packard Enterprise Software Support Online web site at https://softwaresupport.hpe.com.