Hewlett Packard
Enterprise

Codar

Software version: 1.70.0001

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

Document release date: September 2016

Software release date: September 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.

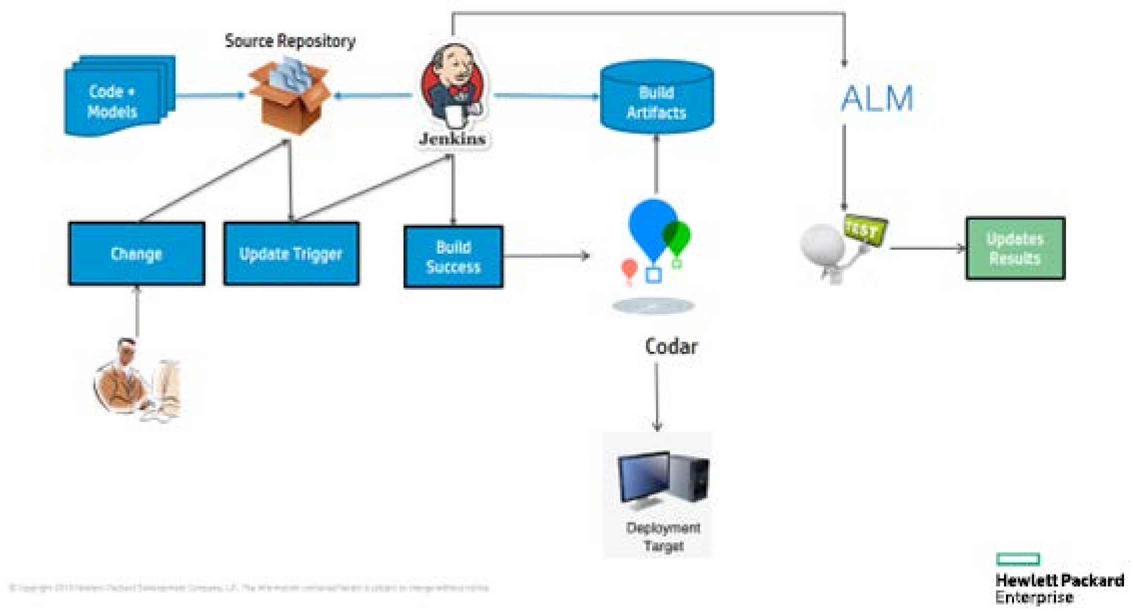
HPE Application Lifecycle Management

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.

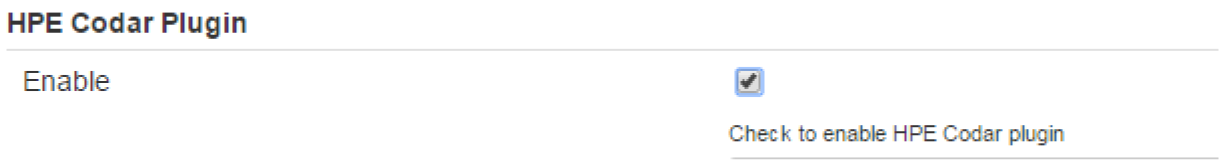
Installing the Codar plugin

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-4.7\Content Archives\topology\Jenkins plugin\HPE_Codar.hpi
6. Click **Upload**.

   Upon successful upload, the system returns a 'Success' message.

Enabling the Codar plugin

1. Click **Manage Jenkins** on the Jenkins dashboard.
2. Click **Configure System**.
3. Scroll down to the **HPE Codar Plugin** section and select the **Enable** check box, if not selected by default.


HPE Codar Plugin

Enable ✓

Check to enable HPE Codar plugin

4. Provide the following details:
   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. .
   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 on the computer on which Codar is installed (Example: Windows: In the C:\ProgramFiles\HPE\Codar\openjre\lib\security\cacerts path. Linux: In the /root/temp/cacerts path. Ensure that 'cacerts

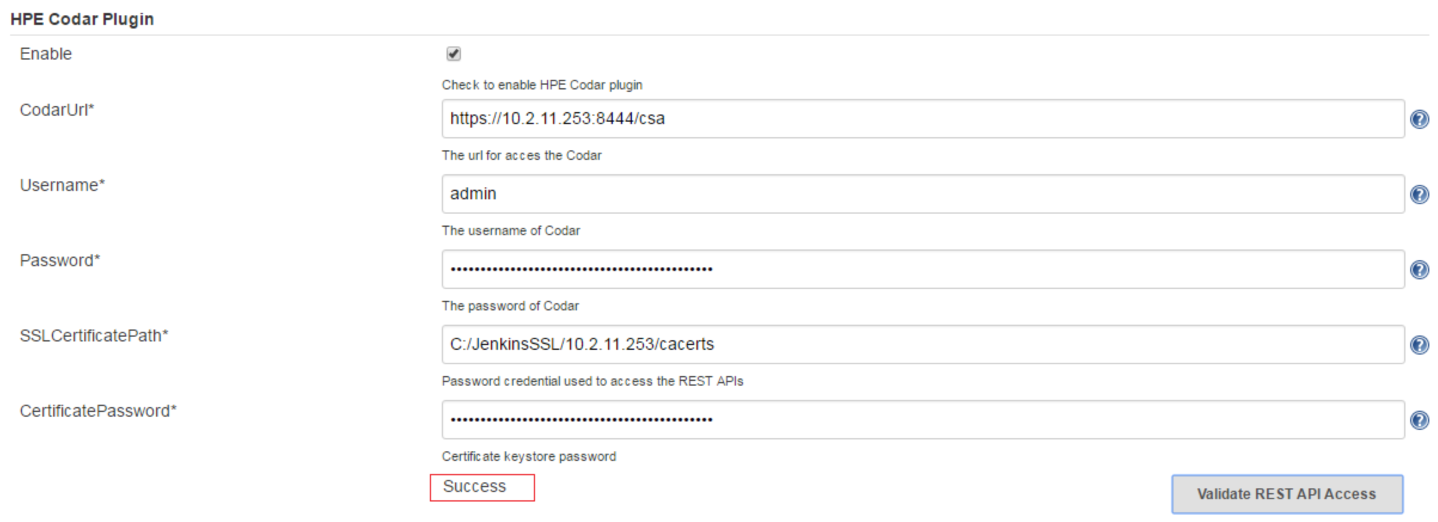
path' has the read/write permission on the server on which Codar is installed). Details about the JRE used during installation is located in the **csa.properties** file.

e. CertificatePassword – Enter the SSL certificate keystore password for Codar. The default password is 'changeit'.

5. Test the link by clicking 'Validate REST API Access'.
6. Ensure that the validation returns 'Success' as shown in the screenshot below, by resolving connectivity issues, if any.
7. 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


HPE Codar Plugin

Enable	✓ Check to enable HPE Codar plugin
CodarUrl*	https://10.2.11.253:8444/csa The url for acces the Codar
Username*	admin The username of Codar
Password*	•• The password of Codar
SSLCertificatePath*	C:/JenkinsSSL/10.2.11.253/cacerts Password credential used to access the REST APIs
CertificatePassword*	•• Certificate keystore password

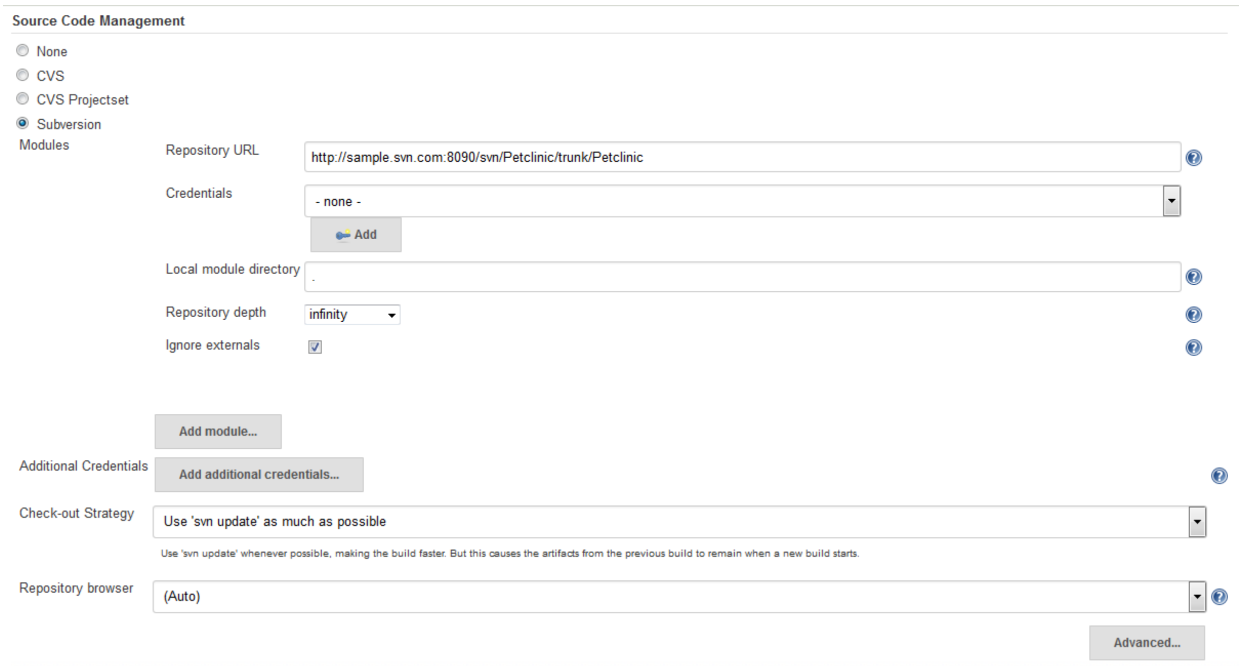
Success

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


Source Code Management

- ( ) None
- ( ) CVS
- ( ) CVS Projectset
- (●) Subversion

Modules

Repository URL	http://sample.svn.com:8090/svn/Petclinic/trunk/Petclinic
Credentials	- none -
Local module directory	.
Repository depth	infinity
Ignore externals	✓

Additional Credentials

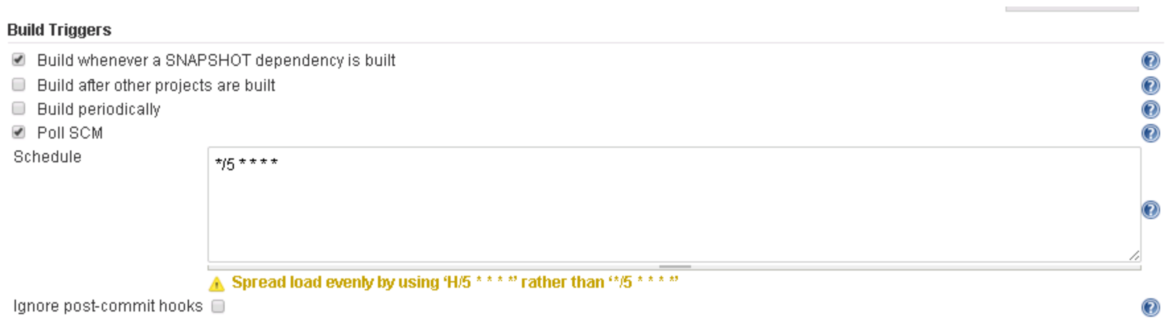
Check-out Strategy: Use 'svn update' as much as possible

Use 'svn update' whenever possible, making the build faster. But this causes the artifacts from the previous build to remain when a new build starts.

Repository browser: (Auto)

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


Build Triggers

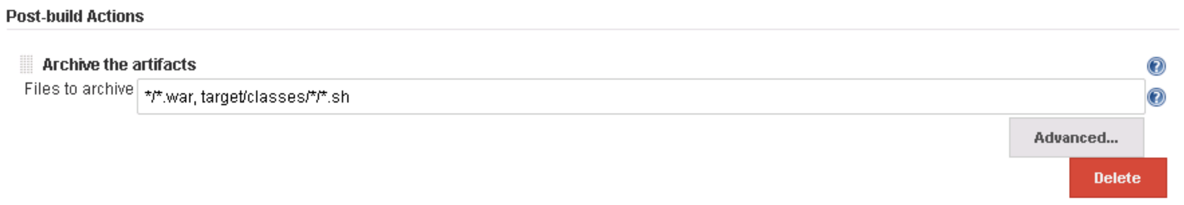
- ✓ Build whenever a SNAPSHOT dependency is built
- Build after other projects are built
- Build periodically
- ✓ Poll SCM

Schedule: */5 * * * *

Spread load evenly by using 'H/5 * * * *' rather than '*/5 * * * *'

Ignore post-commit hooks

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.


Post-build Actions

Archive the artifacts

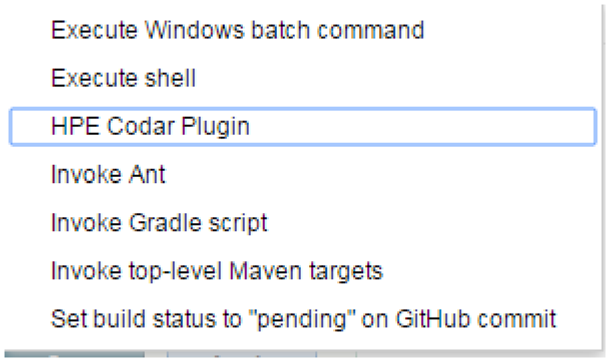
Files to archive: */*.war, target/classes/*/*.sh

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.

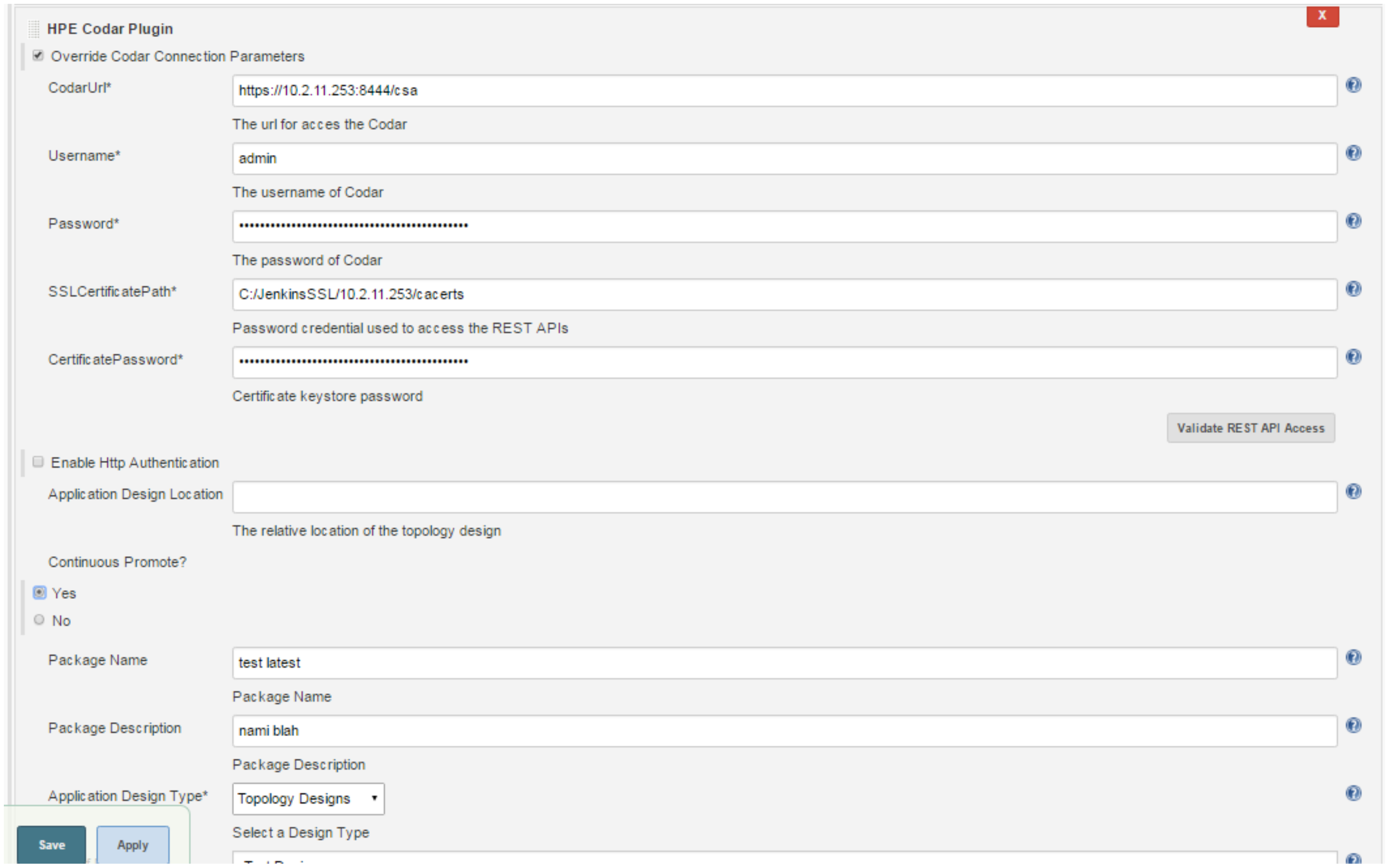
2. Click **Add build step** and select **HPE Codar Plugin**.
   Note: For builds with pre-build and post-build actions, you must configure the Codar plugin as part of post-build actions. For example, in a Maven project, configure the Codar plugin-in as a post-build action.

Figure 5: Configuring the Codar plugin


Execute Windows batch command
Execute shell
HPE Codar Plugin
Invoke Ant
Invoke Gradle script
Invoke top-level Maven targets
Set build status to "pending" on GitHub commit

3. Enter the Codar plugin properties as follows:

   Figure 6: Codar properties



HPE Codar Plugin

[✓ Override Codar Connection Parameters]

CodarUrl* https://10.2.11.253:8444/csa
The url for acces the Codar

Username* admin
The username of Codar

Password* ••
The password of Codar

SSLCertificatePath* C:/JenkinsSSL/10.2.11.253/cacerts
Password credential used to access the REST APIs

CertificatePassword* ••
Certificate keystore password

Enable Http Authentication

Application Design Location
The relative location of the topology design

Continuous Promote?
[✓ Yes] No

Package Name test latest
Package Name

Package Description nami blah
Package Description

Application Design Type* Topology Designs
Select a Design Type

Note: If you select the 'Override Codar Connection Parameters' option, you must provide the details from steps 'a' to 'e'. Skip to step 'f', if you do not select this option.

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. The default password is 'changeit'.

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

g. 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.

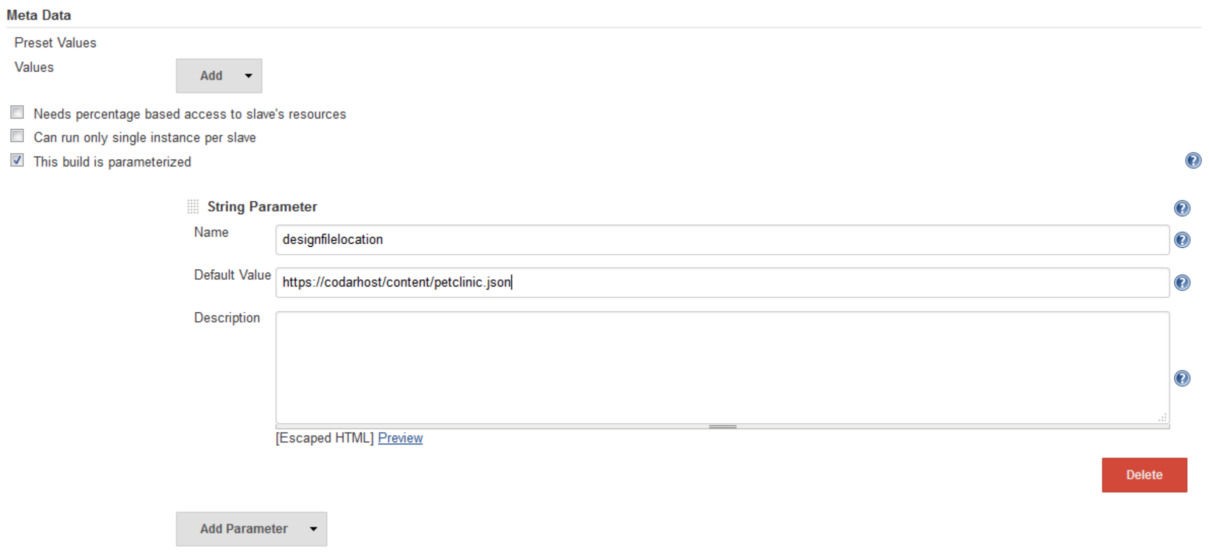
h. HttpPassword – Password for accessing artifacts from the HTTP location. For example, if the artifacts are at a location in Jenkins, enter the password of the Jenkins server.

i. Application Design Location – This is an optional parameter which 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. 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.


Meta Data
Preset Values
Values

☐ Needs percentage based access to slave's resources
☐ Can run only single instance per slave
☑ This build is parameterized
String Parameter
Name: designfilelocation
Default Value: https://codarhost/content/petclinic.json
Description:
[Escaped HTML] Preview

j. 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.

- If this option is selected as 'Yes', you cannot specify the environment.

- If this option is selected as 'No', you must specify the environment.

k. Package Name – Name of the package

Note: Package Name can be a build parameter or Jenkins inbuilt variables like $BUILD_ID. For example, package name can be derived from the build parameter 'PackageName' and the value can be 'Package: $BUILD_ID'. You can set the package name as '$PackageName'.

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

m. 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.

n. 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.

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

p. 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.

Note: A resource environment is required for the Codar plug-in to work.

q. Package Properties – The following Jenkins environment variables are supported:

- ${build_id}
- ${build_number}
- ${build_url}
- ${job_name}

You can specify the value of the property in the following ways:

- Artifact name like petclinic.war

  Note: An artifact is a group of files with a .war .jar or .zip. extension. You can directly specify the Jenkins artifact name as 'petclinic.war' for Jenkins. The 'petclinic.war' will then automatically be resolved to the http URL of the Jenkins artifact, and substituted in the package property value. For artefacts other than Jenkins, you need to specify the complete http URL.

- Property value – The property values allows user to specify parameterized values which allows users to dynamically reference the current build artifacts built by the Jenkins Job.

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
```

Integrating ALM with Codar

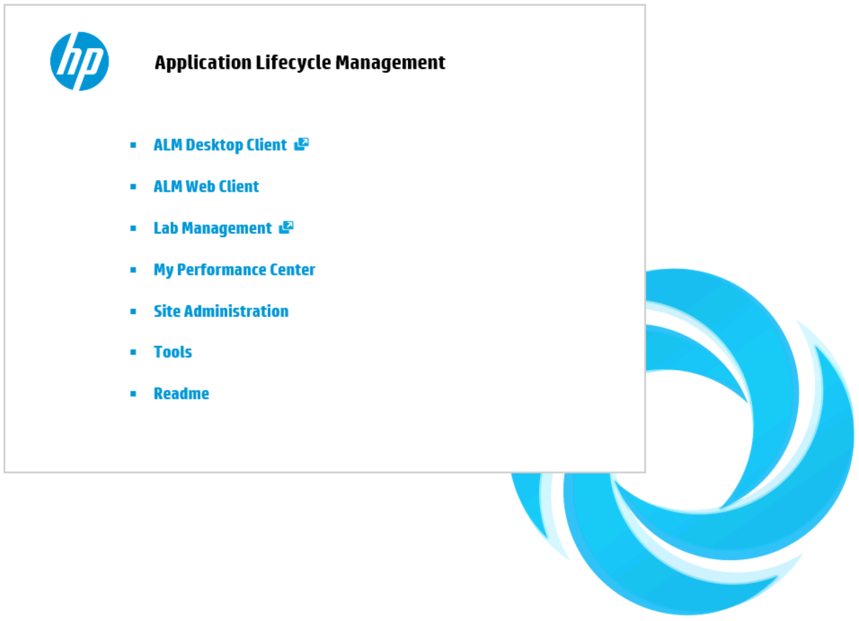
Codar supports native ALM integration. To setup native ALM integration in Codar, please refers Online Help Guide.

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 7: Configurations required for the integration

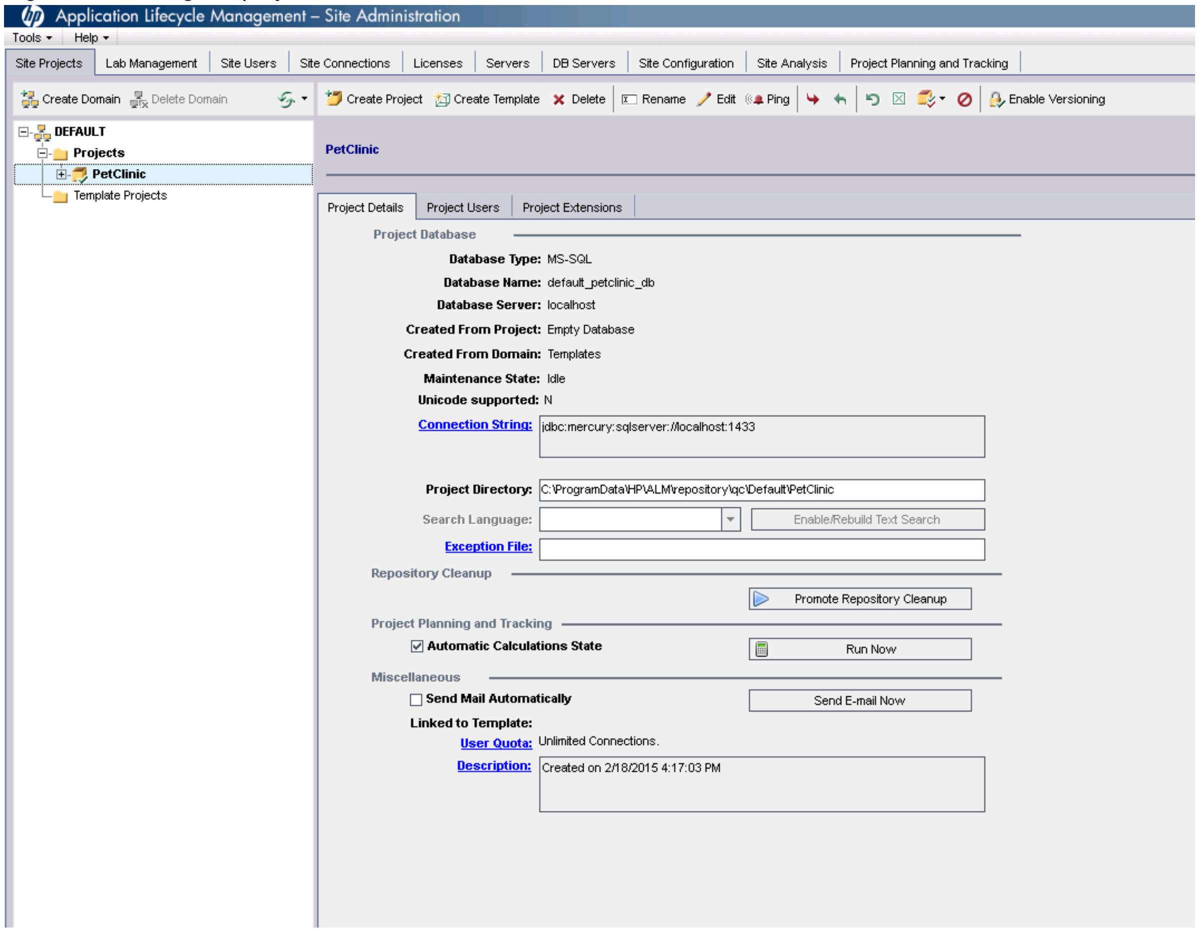


After installing ALM, open the http://localhost:8080/qcbin/ link using Internet Explorer.

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 8: Entering the project details



Project Database

Database Type: MS-SQL
Database Name: default_petclinic_db
Database Server: localhost
Created From Project: Empty Database
Created From Domain: Templates
Maintenance State: Idle
Unicode supported: N
Connection String: jdbc:mercury:sqlserver://localhost:1433

Project Directory: C:\ProgramData\HP\ALM\repository\qc\Default\PetClinic
Search Language:
Exception File:

Repository Cleanup

Project Planning and Tracking

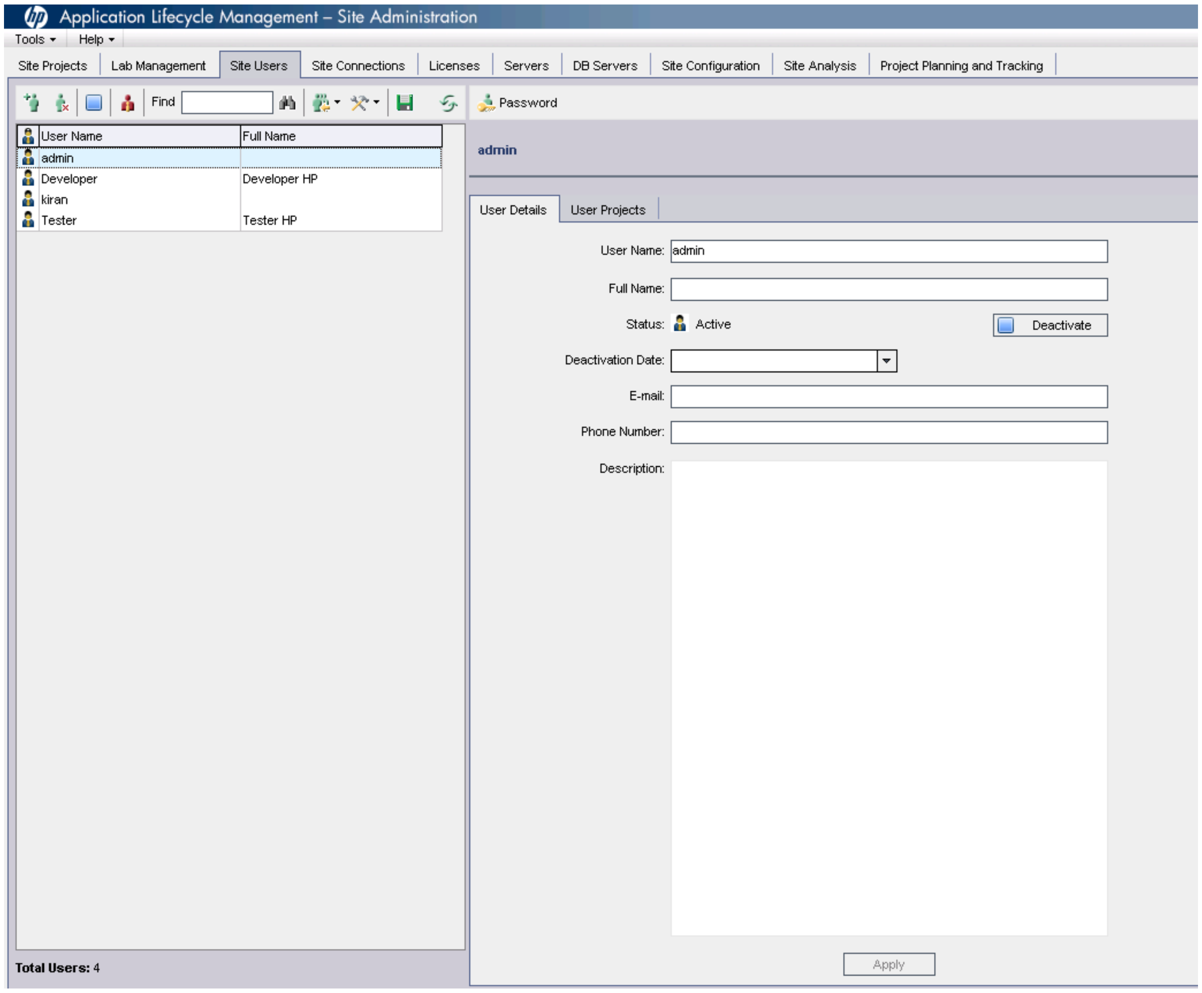
☑ Automatic Calculations State

Miscellaneous

☐ Send Mail Automatically
Linked to Template:
User Quota: Unlimited Connections.
Description: Created on 2/18/2015 4:17:03 PM

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 9: Creating a user



User Name	Full Name
admin	
Developer	Developer HP
kiran	
Tester	Tester HP

Total Users: 4

admin

User Name: admin
Full Name:
Status: Active
Deactivation Date:
E-mail:
Phone Number:
Description:

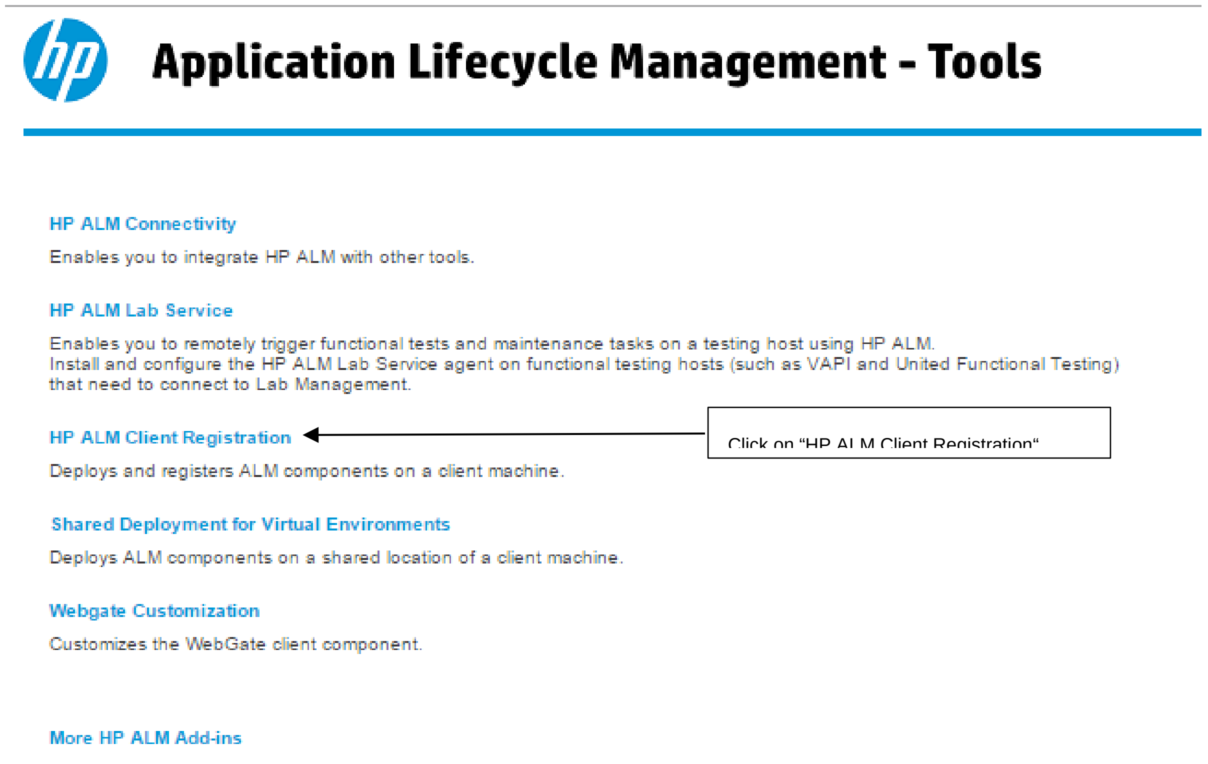
For more information about using Site Administration, see the ALM Administration Guide.

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 10: Installing VAPI-XP



Application Lifecycle Management - Tools

HP ALM Connectivity

Enables you to integrate HP ALM with other tools.

HP ALM Lab Service

Enables you to remotely trigger functional tests and maintenance tasks on a testing host using HP ALM.
Install and configure the HP ALM Lab Service agent on functional testing hosts (such as VAPI and United Functional Testing) that need to connect to Lab Management.

HP ALM Client Registration

Deploys and registers ALM components on a client machine.

Shared Deployment for Virtual Environments

Deploys ALM components on a shared location of a client machine.

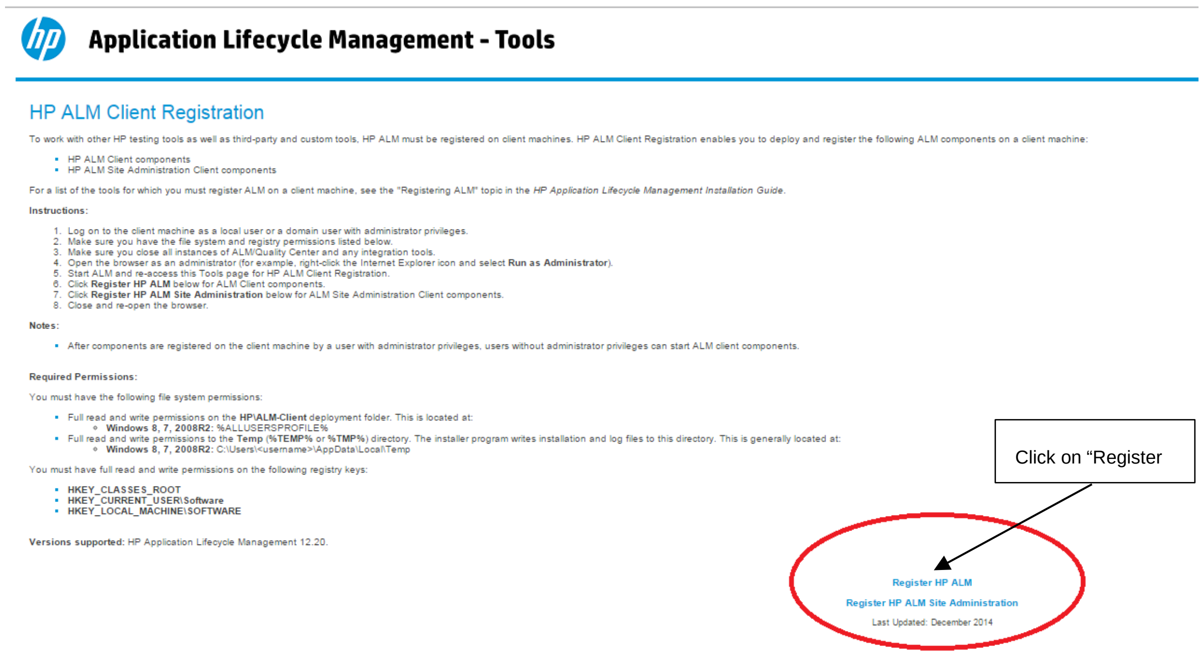
Webgate Customization

Customizes the WebGate client component.

More HP ALM Add-ins

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

Figure 11: Downloading libraries



Application Lifecycle Management - Tools

HP ALM Client Registration

To work with other HP testing tools as well as third-party and custom tools, HP ALM must be registered on client machines. HP ALM Client Registration enables you to deploy and register the following ALM components on a client machine:

- HP ALM Client components
- HP ALM Site Administration Client components

For a list of the tools for which you must register ALM on a client machine, see the "Registering ALM" topic in the *HP Application Lifecycle Management Installation Guide*.

Instructions:

1. Log on to the client machine as a local user or a domain user with administrator privileges.
2. Make sure you have the file system and registry permissions listed below.
3. Make sure you close all instances of ALM/Quality Center and any integration tools.
4. Open the browser as an administrator (for example, right-click the Internet Explorer icon and select **Run as Administrator**).
5. Start ALM and re-access this Tools page for HP ALM Client Registration.
6. Click **Register HP ALM** below for ALM Client components.
7. Click **Register HP ALM Site Administration** below for ALM Site Administration Client components.
8. Close and re-open the browser.

Notes:

- After components are registered on the client machine by a user with administrator privileges, users without administrator privileges can start ALM client components.

Required Permissions:

You must have the following file system permissions:

- Full read and write permissions on the **HP\ALM-Client** deployment folder. This is located at:
  - **Windows 8, 7, 2008R2:** %ALLUSERSPROFILE%
- Full read and write permissions to the **Temp** (**%TEMP%** or **%TMP%**) directory. The installer program writes installation and log files to this directory. This is generally located at:
  - **Windows 8, 7, 2008R2:** C:\Users\<username>\AppData\Local\Temp

You must have full read and write permissions on the following registry keys:

- **HKEY_CLASSES_ROOT**
- **HKEY_CURRENT_USER\Software**
- **HKEY_LOCAL_MACHINE\SOFTWARE**

Versions supported: HP Application Lifecycle Management 12.20.

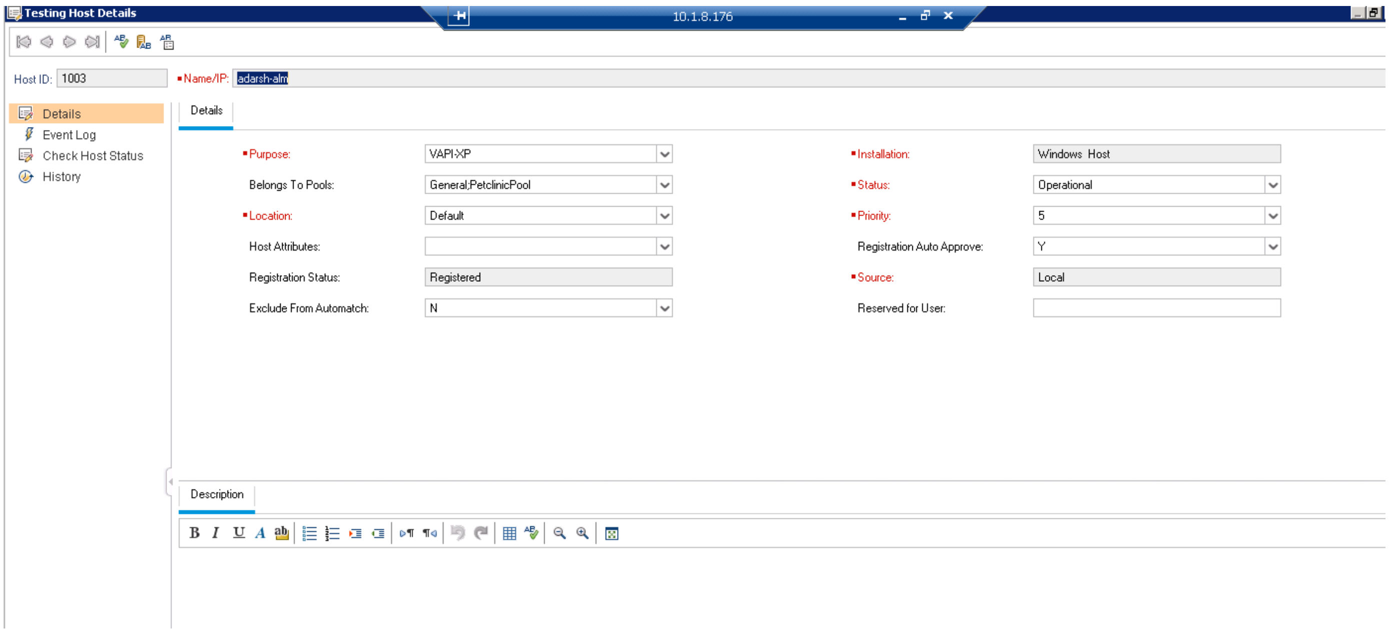
Click on "Register
Register HP ALM
Register HP ALM Site Administration
Last Updated: December 2014

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 12: Testing hosts

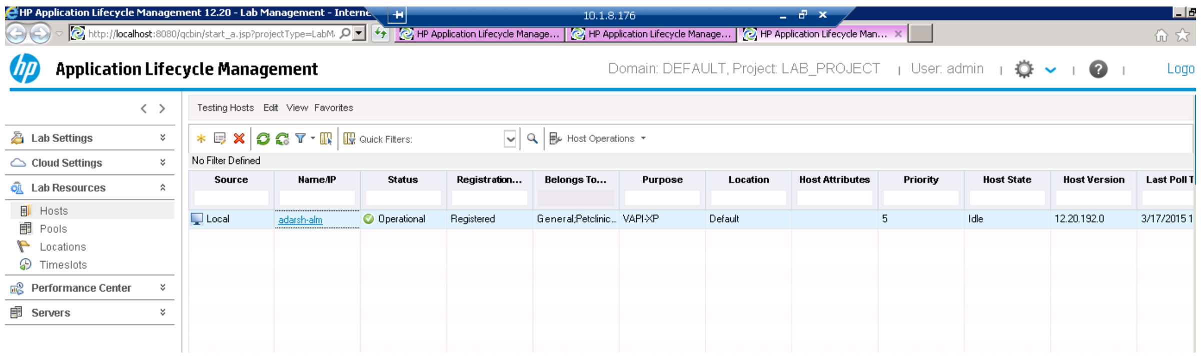

Testing Host Details

Host ID: 1003 ▪Name/IP: adarsh-alm

▪Purpose:	VAPI-XP	▪Installation:	Windows Host
Belongs To Pools:	General;PetclinicPool	▪Status:	Operational
▪Location:	Default	▪Priority:	5
Host Attributes:		Registration Auto Approve:	Y
Registration Status:	Registered	▪Source:	Local
Exclude From Automatch:	N	Reserved for User:	

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

Figure 13: New host


HP Application Lifecycle Management 12.20 - Lab Management - Interne

http://localhost:8080/qcbin/start_a.jsp?projectType=LabM

Application Lifecycle Management — Domain: DEFAULT, Project: LAB_PROJECT | User: admin

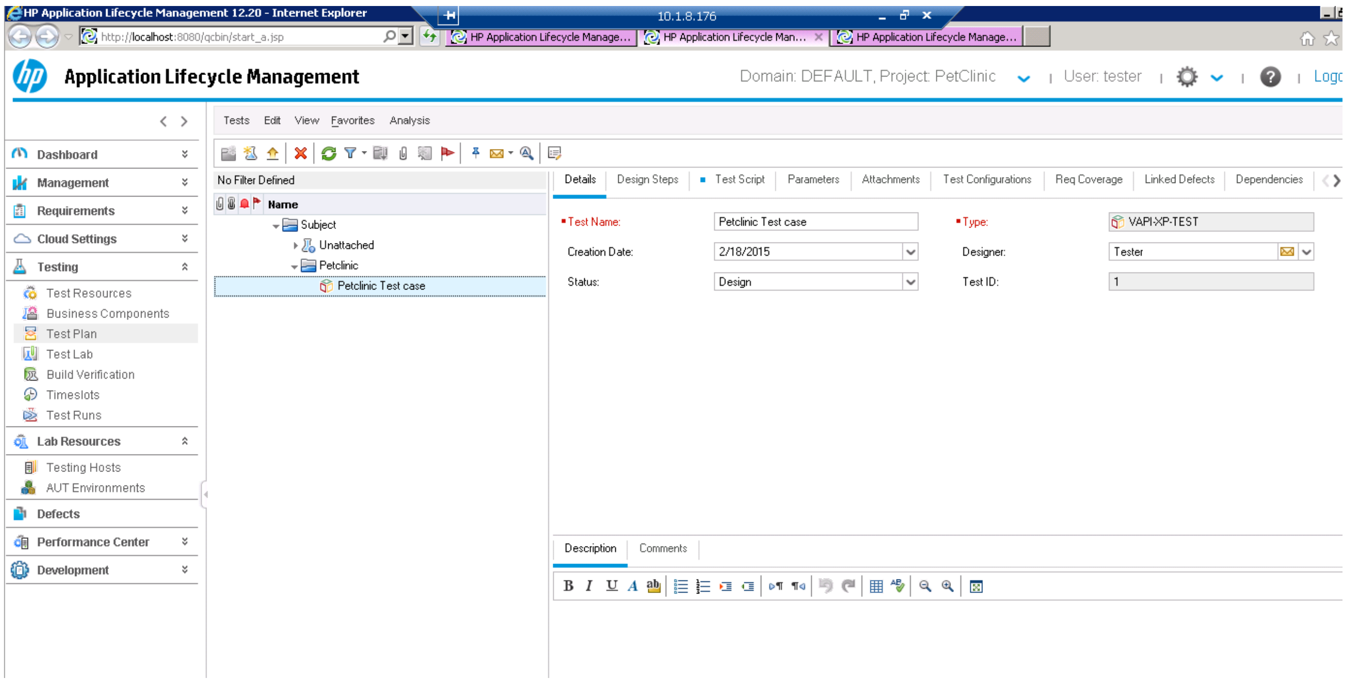
No Filter Defined

Source	Name/IP	Status	Registration...	Belongs To...	Purpose	Location	Host Attributes	Priority	Host State	Host Version	Last Poll T
Local	adarsh-alm	Operational	Registered	General;Petclinic...	VAPI-XP	Default		5	Idle	12.20.192.0	3/17/2015 1

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 14: Creating a test case



HP Application Lifecycle Management 12.20 - Internet Explorer

http://localhost:8080/qcbin/start_a.jsp

Application Lifecycle Management — Domain: DEFAULT, Project: PetClinic | User: tester

No Filter Defined

- Subject
  - Unattached
  - Petclinic
    - Petclinic Test case

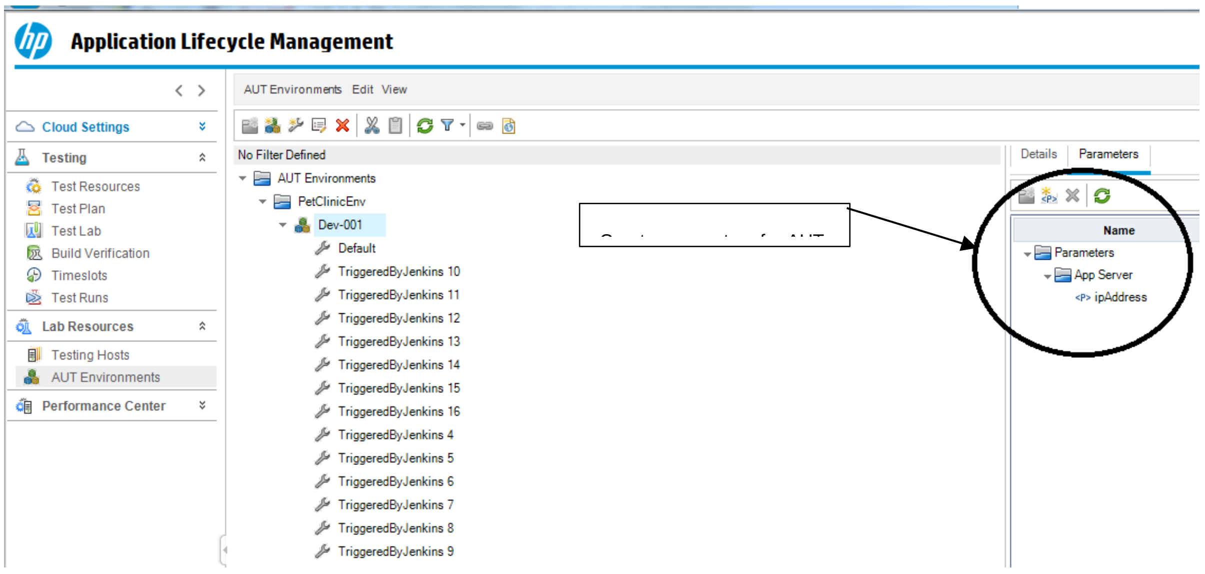
Test Name:	Petclinic Test case	Type:	VAPI-XP-TEST
Creation Date:	2/18/2015	Designer:	Tester
Status:	Design	Test ID:	1

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 15: Added the App Server parameter

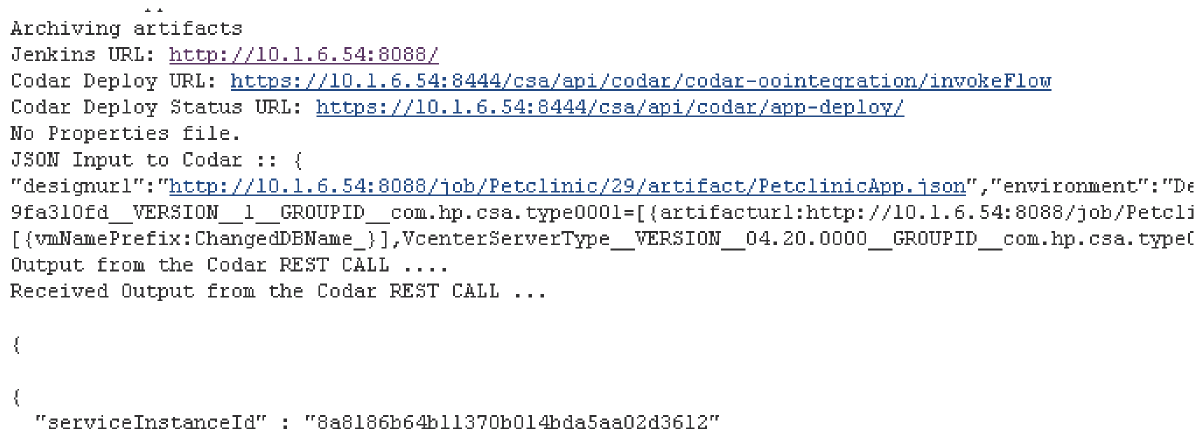


End-to-end flow

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.

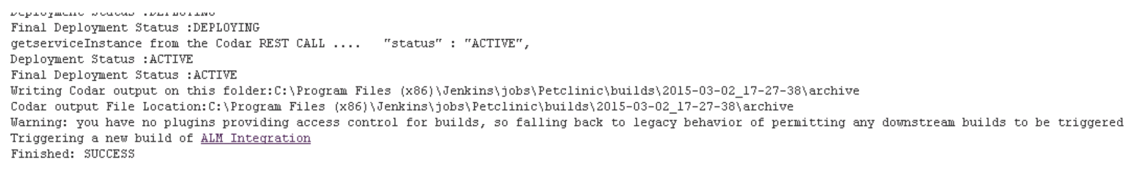
Figure 16: Triggering the post build action



```
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 17: Deployment complete

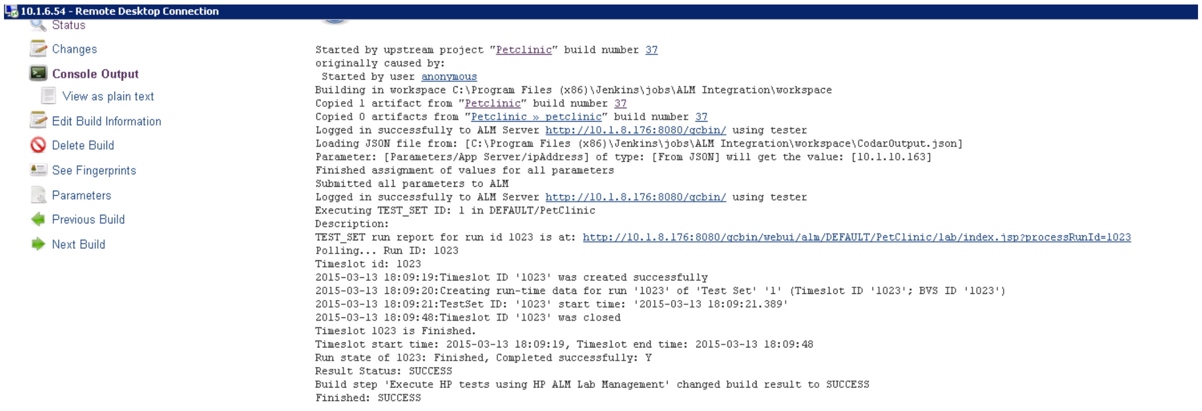


```
Final Deployment Status :DEPLOYING
getserviceInstance from the Codar REST CALL ....   "status" : "ACTIVE",
Deployment Status :ACTIVE
Final Deployment Status :ACTIVE
Writing Codar output on this folder:C:\Program Files (x86)\Jenkins\jobs\Petclinic\builds\2015-03-02_17-27-38\archive
Codar output File 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.

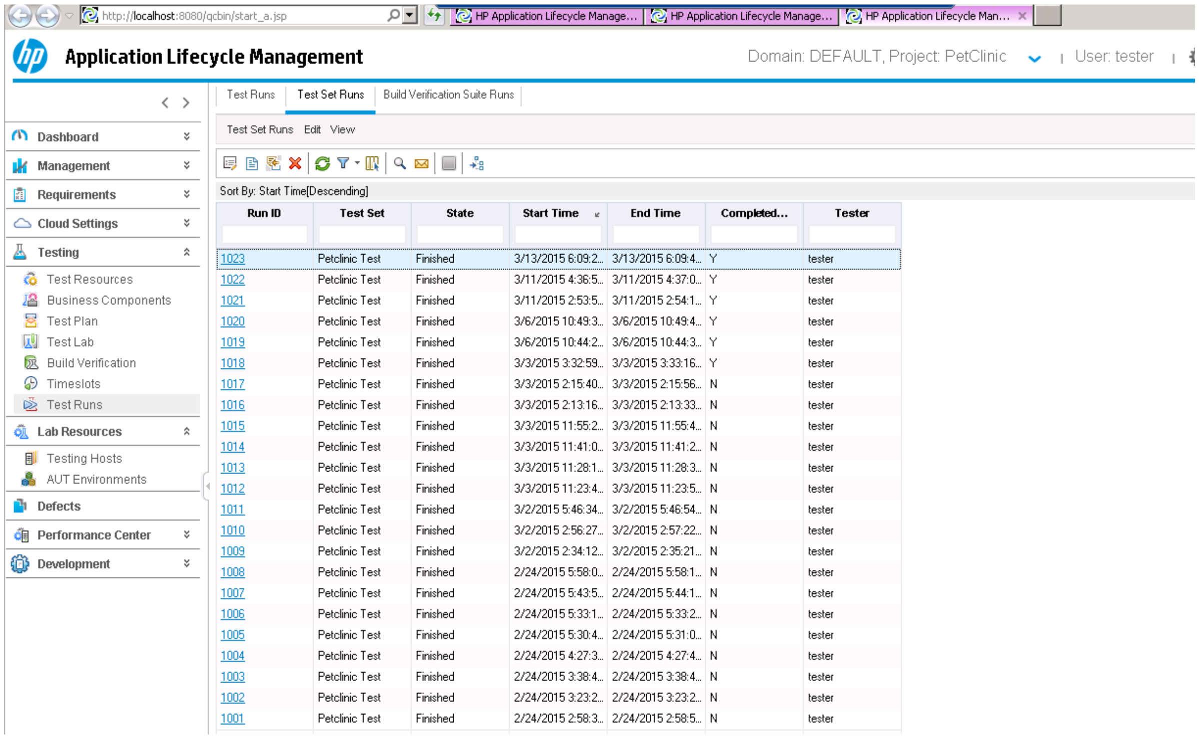
Figure 18: Successful job status



```
Started by upstream project "Petclinic" build number 37
originally caused by:
 Started by user anonymous
Building in workspace C:\Program Files (x86)\Jenkins\jobs\ALM Integration\workspace
Copied 1 artifact from "Petclinic" build number 37
Copied 0 artifacts from "Petclinic » petclinic" build number 37
Logged in successfully to ALM Server http://10.1.8.176:8080/qcbin/ using tester
Loading JSON file from: [C:\Program Files (x86)\Jenkins\jobs\ALM Integration\workspace\CodarOutput.json]
Parameter: [Parameters/App Server/ipAddress] of type: [From JSON] will get the value: [10.1.10.163]
Finished assignment of values for all parameters
Submitted all parameters to ALM
Logged in successfully to ALM Server http://10.1.8.176:8080/qcbin/ using tester
Executing TEST_SET ID: 1 in DEFAULT/PetClinic
Description:
TEST_SET run report for run id 1023 is at: http://10.1.8.176:8080/qcbin/webui/alm/DEFAULT/PetClinic/lab/index.jsp?processRunId=1023
Polling... Run ID: 1023
Timeslot id: 1023
2015-03-13 18:09:19:Timeslot ID '1023' was created successfully
2015-03-13 18:09:20:Creating run-time data for run '1023' of 'Test Set' '1' (Timeslot ID '1023'; BVS ID '1023')
2015-03-13 18:09:21:TestSet ID: '1023' start time: '2015-03-13 18:09:21.389'
2015-03-13 18:09:48:Timeslot ID '1023' was closed
Timeslot 1023 is Finished.
Timeslot start time: 2015-03-13 18:09:19, Timeslot end time: 2015-03-13 18:09:48
Run state of 1023: Finished, Completed successfully: Y
Result Status: SUCCESS
Build step 'Execute HP tests using HP ALM Lab Management' changed build result to SUCCESS
Finished: SUCCESS
```

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 19: Updated test results



Sort By: Start Time[Descending]

Run ID	Test Set	State	Start Time	End Time	Completed...	Tester
1023	Petclinic Test	Finished	3/13/2015 6:09:2...	3/13/2015 6:09:4...	Y	tester
1022	Petclinic Test	Finished	3/11/2015 4:36:5...	3/11/2015 4:37:0...	Y	tester
1021	Petclinic Test	Finished	3/11/2015 2:53:5...	3/11/2015 2:54:1...	Y	tester
1020	Petclinic Test	Finished	3/6/2015 10:49:3...	3/6/2015 10:49:4...	Y	tester
1019	Petclinic Test	Finished	3/6/2015 10:44:2...	3/6/2015 10:44:3...	Y	tester
1018	Petclinic Test	Finished	3/3/2015 3:32:59...	3/3/2015 3:33:16...	Y	tester
1017	Petclinic Test	Finished	3/3/2015 2:15:40...	3/3/2015 2:15:56...	N	tester
1016	Petclinic Test	Finished	3/3/2015 2:13:16...	3/3/2015 2:13:33...	N	tester
1015	Petclinic Test	Finished	3/3/2015 11:55:2...	3/3/2015 11:55:4...	N	tester
1014	Petclinic Test	Finished	3/3/2015 11:41:0...	3/3/2015 11:41:2...	N	tester
1013	Petclinic Test	Finished	3/3/2015 11:28:1...	3/3/2015 11:28:3...	N	tester
1012	Petclinic Test	Finished	3/3/2015 11:23:4...	3/3/2015 11:23:5...	N	tester
1011	Petclinic Test	Finished	3/2/2015 5:46:34...	3/2/2015 5:46:54...	N	tester
1010	Petclinic Test	Finished	3/2/2015 2:56:27...	3/2/2015 2:57:22...	N	tester
1009	Petclinic Test	Finished	3/2/2015 2:34:12...	3/2/2015 2:35:21...	N	tester
1008	Petclinic Test	Finished	2/24/2015 5:58:0...	2/24/2015 5:58:1...	N	tester
1007	Petclinic Test	Finished	2/24/2015 5:43:5...	2/24/2015 5:44:1...	N	tester
1006	Petclinic Test	Finished	2/24/2015 5:33:1...	2/24/2015 5:33:2...	N	tester
1005	Petclinic Test	Finished	2/24/2015 5:30:4...	2/24/2015 5:31:0...	N	tester
1004	Petclinic Test	Finished	2/24/2015 4:27:3...	2/24/2015 4:27:4...	N	tester
1003	Petclinic Test	Finished	2/24/2015 3:38:4...	2/24/2015 3:38:4...	N	tester
1002	Petclinic Test	Finished	2/24/2015 3:23:2...	2/24/2015 3:23:2...	N	tester
1001	Petclinic Test	Finished	2/24/2015 2:58:3...	2/24/2015 2:58:5...	N	tester

Troubleshooting Codar Integration with Jenkins and ALM

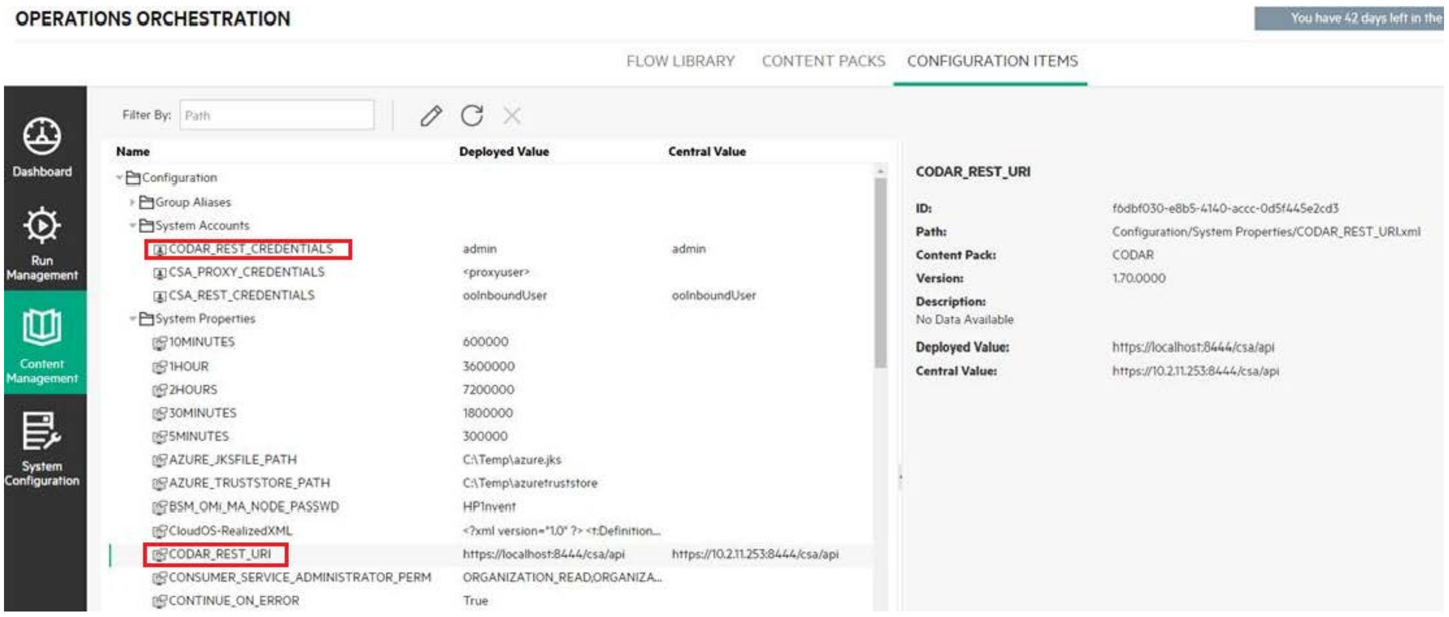
This section contains some of the issues that you may encounter when integrating Codar with Jenkins and ALM, and workarounds to troubleshoot these issues.

Plug-in execution begins but fails

Problem: Codar plug-in execution begins but fails with some error

Solution/Workaround: In the Operations Orchestration console>>Content Management>>Configuration Items>>Systems account>>System properties, overwrite the following properties with the appropriate credentials:

- CODAR_REST_CREDENTIALS
- CODAR_REST_URI



Name	Deployed Value	Central Value
Configuration		
Group Aliases		
System Accounts		
CODAR_REST_CREDENTIALS	admin	admin
CSA_PROXY_CREDENTIALS	<proxyuser>	
CSA_REST_CREDENTIALS	oolnboundUser	oolnboundUser
System Properties		
10MINUTES	600000	
1HOUR	3600000	
2HOURS	7200000	
30MINUTES	1800000	
5MINUTES	300000	
AZURE_JKSFILE_PATH	C:\Temp\azure.jks	
AZURE_TRUSTSTORE_PATH	C:\Temp\azuretruststore	
BSM_OMI_MA_NODE_PASSWD	HPInvent	
CloudOS-RealizedXML	<?xml version="1.0" ?> <t:Definition...	
CODAR_REST_URI	https://localhost:8444/csa/api	https://10.2.11.253:8444/csa/api
CONSUMER_SERVICE_ADMINISTRATOR_PERM	ORGANIZATION_READ,ORGANIZA...	
CONTINUE_ON_ERROR	True	

CODAR_REST_URI

ID:	f6dbf030-e8b5-4140-accc-0d5f445e2cd3
Path:	Configuration/System Properties/CODAR_REST_URI.xml
Content Pack:	CODAR
Version:	1.70.0000
Description: No Data Available	
Deployed Value:	https://localhost:8444/csa/api
Central Value:	https://10.2.11.253:8444/csa/api

Trouble integrating Codar with Jenkins and ALM

Problem: Exception/error while integrating Codar with Jenkins and ALM.

Solution/Workaround: Check the following logs for detailed information:

- Jenkins-related issues: jenkins.err.log (This file exists in the Jenkins installation directory).
- Codar-related-issues: csa.log (This file exists in the following path: `<CODAR_HOME>`\jboss-as\standalone\log)
- OO-related issues: Operations Orchestrations console.

Send documentation feedback

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

Legal notices

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.7