



Codar

Software Version: 1.90

For Linux operating system

Get started

Document Release Date: September 2017

Software Release Date: September 2017



Hewlett Packard
Enterprise

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

© 2017 Hewlett Packard Enterprise Development LP

Trademark Notices

Adobe™ is a trademark of Adobe Systems Incorporated.

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

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

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

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

This product includes an interface of the 'zlib' general purpose compression library, which is Copyright © 1995-2002 Jean-loup Gailly and Mark Adler.

Documentation Updates

To check for recent updates or to verify that you are using the most recent edition of a document, go to: <https://softwaresupport.hpe.com/>.

This site requires that you register for an HP Passport and to sign in. To register for an HP Passport ID, click **Register** on the HPE Software Support site or click **Create an Account** on the HP Passport login page.

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

Support

Visit the HPE Software Support site at: <https://softwaresupport.hpe.com/>.

Most of the support areas require that you register as an HP Passport user and to sign in. Many also require a support contract. To register for an HP Passport ID, click **Register** on the HPE Support site or click **Create an Account** on the HP Passport login page.

To find more information about access levels, go to: <https://softwaresupport.hpe.com/web/softwaresupport/access-levels>.

HPE Software Solutions Now accesses the HPSW Solution and Integration Portal website. This site enables you to explore HPE Product Solutions to meet your business needs, includes a full list of Integrations between HPE Products, as well as a listing of ITIL Processes. The URL for this website is <https://softwaresupport.hpe.com/km/KM01702731>.

Contents

Get started	5
Key concepts	6
What is DevOps?	7
Codar overview	8
Components	9
Service Designs	10
Sequenced design	10
Topology design	10
Declarative-based modeling	11
Topology composition	12
Microservices	14
Application pipeline management	16
Managing packages	18
Package operations	20
Deploy and redeploy	21
Scale out	22
Roles and user access	24
Lifecycle stages and actions	27
Package states	28
Grouping service designs by lifecycle stage	29
Release gate actions	30
Pipeline statistics	32
Environments	33
Schedule releases	34
External integrations	36
Jenkins integration	37
ALM integration	38
Infrastructure as code (IaC)	39

Deploy containers on Docker Cluster	40
SAML support for Codar	42
Use cases	43
Continuous integration, deployment, and delivery	45
Application modeling	46
Continuous integration and deployment	47
Importing an application design	48
Deploying on an environment	49
Publishing a design	50
Customizable release pipeline	53
Deploy and redeploy packages	55
Deployment and scale out	57
Quickstart	58
Prerequisites to using the PetClinic application	59
Downloading files	60
Accessing the application design	62
Customizing components	64
Customizing the existing server	64
Customizing the second existing server	65
Customizing MySQL	65
Customizing Tomcat	66
Customizing the PetClinic DB Conf component	67
Customizing the PetClinic Application component	68
Deploying the application design	70
Troubleshooting	72
Send documentation feedback	73

Get started

You can get started with Application Release Automation (Codar) using the following information.

- [Key concepts](#)
- [Use cases](#)
- [Quickstart](#)

Key concepts

This section discusses briefly the key concepts used in Codar.

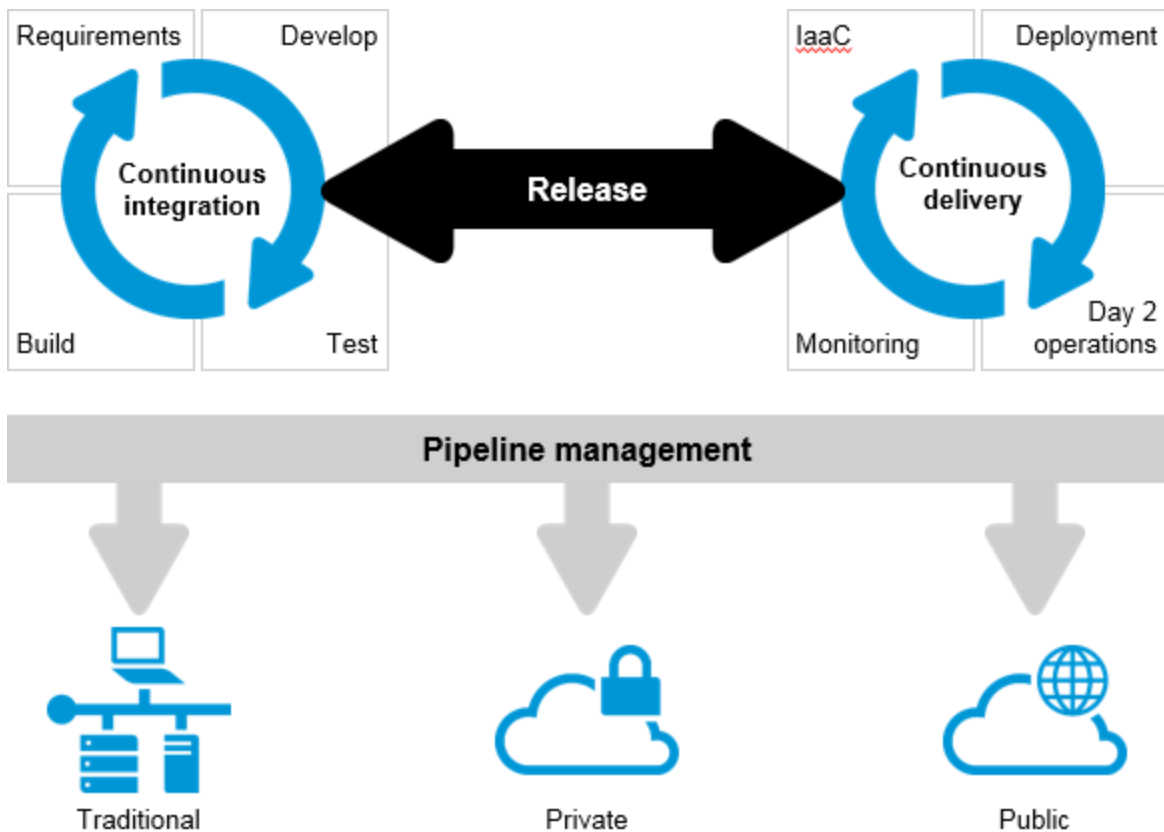
- [What is DevOps?](#)
- [Overview](#)

What is DevOps?

Organizations are facing new challenges when extending continuous integration into continuous delivery. Challenges include consistently deploying applications through development to production environments while considering the differences in those environments.

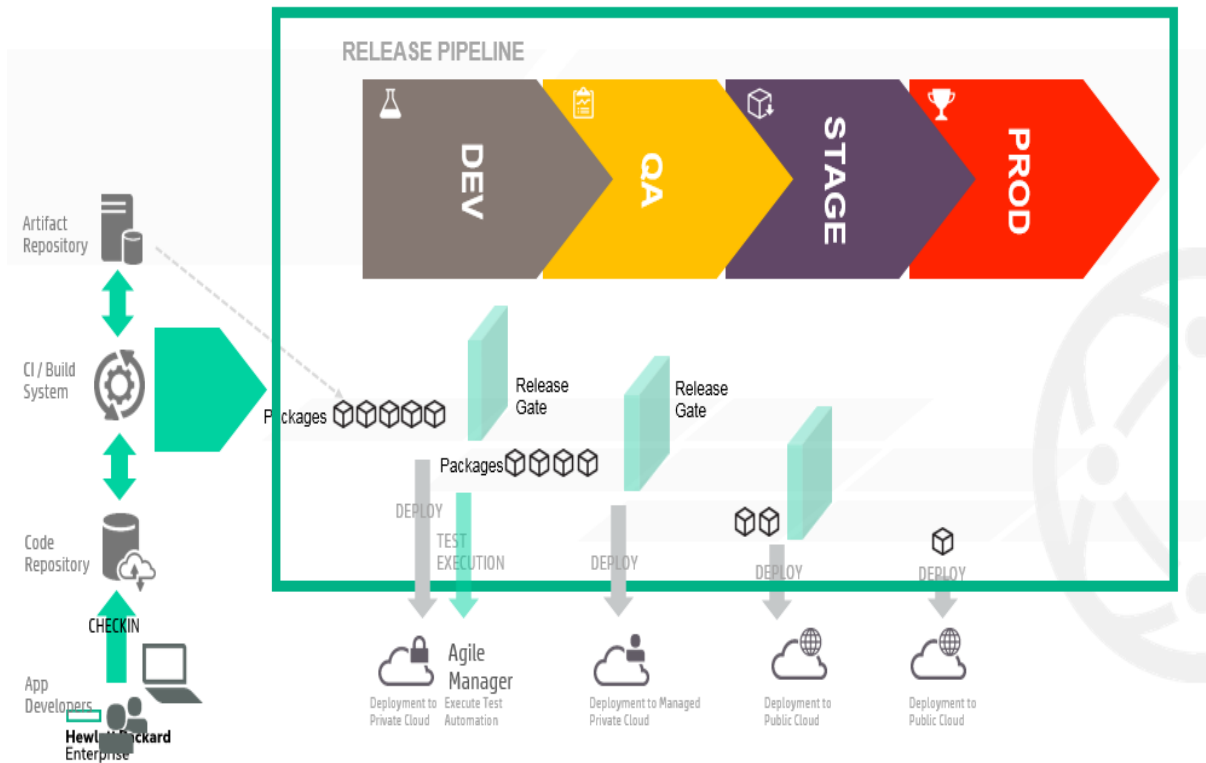
DevOps provides a framework to bridge the gaps between the development (Dev) and operations (Ops) environments by using a set of principles, methods, and practices around collaboration, automation, and governance. The goal is to extend continuous build or assembly integration to repeatable and consistent application deployment across heterogeneous environments.

The following diagram illustrates the continuous integration and continuous delivery cycle in a DevOps environment.



Codar overview

Hewlett Packard Enterprise Codar facilitates continuous delivery in which every change to the system is releasable and every code change can be deployed in production. It enables automation of continuous delivery where every code change triggers a build, which is deployed, automated unit tests are executed, and the application is automatically deployed to an environment based on policies that are defined in a runbook automation flow. Continuous delivery aims to deliver frequently and get quick feedback from users.



Components

Components are elements of service design. Only topology components are displayed in the Components tab. Sequenced components are not associated with providers or provider types. From the Components tab, you can view the topology components associated with a specific provider instance and manage the topological components.

Service Designs

Create, configure, and modify service designs to provide on-demand, automated service delivery. Service designs are the recipes for automating the cloud, and include reusable service components. Service components and their relationships in a service design define the framework for creating the service.

Service designs also provide a structure for options that consumers can select when ordering a service. You can re-use designs for multiple service offerings, with each service offering customized to meet the needs of different consumer organizations and groups. You can also leverage service designs shipped with Codar as well as exporting and importing designs between Codar systems.

You can create sequenced and topology designs.

Sequenced design

Sequenced designs specify directed execution of the service component lifecycle and provide mechanisms for controlling resource selection as each component is deployed. When creating sequenced designs, associate one or more resource offerings on a service component to constrain provider selection. This association or link ensures that the resource offering will be provisioned when the service component is deployed. You can also associate resource offerings with component templates.

Use sequenced designs for complex services and services that rely on runbook automation, such as integrations with legacy data center systems. Create a sequenced design as a directed component hierarchy to define lifecycle execution. Sequenced designs use components to group multiple automation providers within a single entity, and they permit explicit specification of lifecycle actions.

Topology design

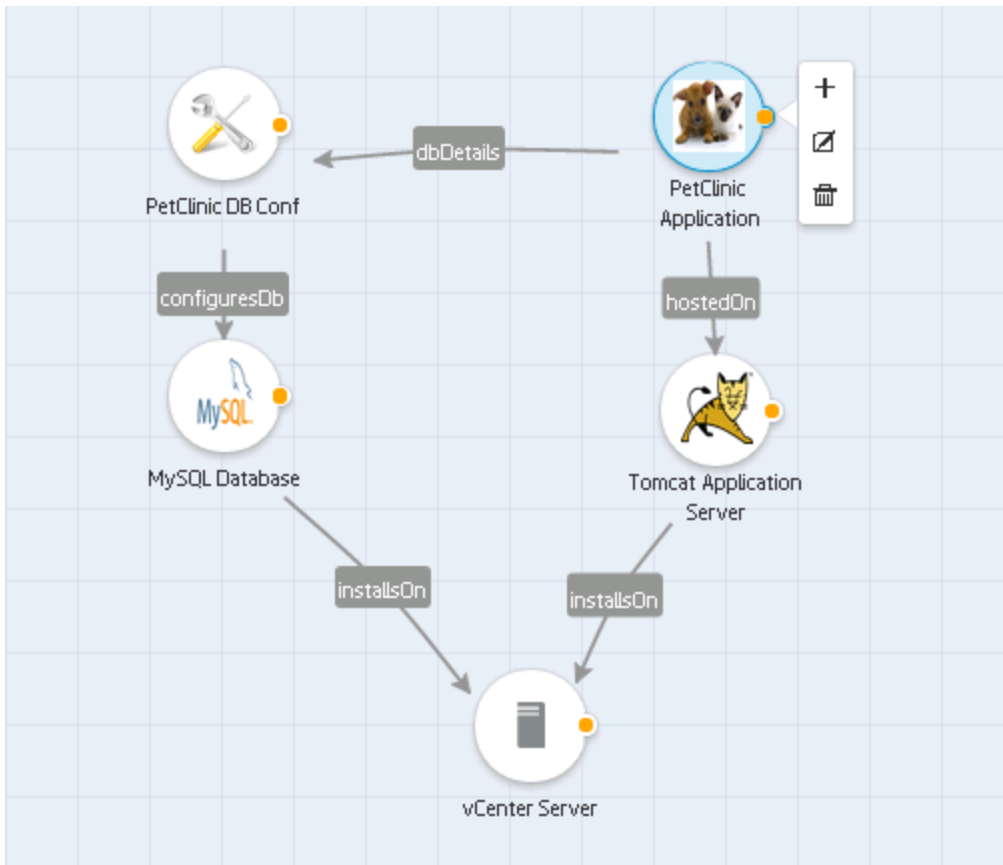
Topology designs specify components, relationships, and properties. In contrast to sequenced designs, which more explicitly define the provisioning order and the sequence of actions that will run, topology designs are declarative in nature and do not include explicit actions or sequencing. The provisioning sequence is inferred by the relationships that exist between components in a topology design.

Use topology designs for Infrastructure as a Service (IaaS), Platform as a Service (PaaS), and Software as a Service (SaaS) deployments that are enabled using Chef, Puppet, Server Automation, and Operations Orchestration flow-based components.

Declarative-based modeling

Automating the deployment of applications using declarative based modeling allows the user to declare the end state of the application deployment (the application components and the dependencies between them) while the process to get to that state is triggered in the background. This allows the user to focus on what is deployed rather than how it gets deployed, which results in a shorter time to automate the deployment of multi-tier applications and greater simplicity in managing them over time.

Codar supports declarative-based model development that involves creation, integration, and maintenance of complex designs through a user interface. A model consists of a topology design and its properties. Codar provides flexibility for the user to modify the properties during the time of realization (similar to late binding).



Topology composition

An application design, also called a topology design, specifies components and their relationships to define the application lifecycle. An application design delegates lifecycle sequencing to cloud providers.

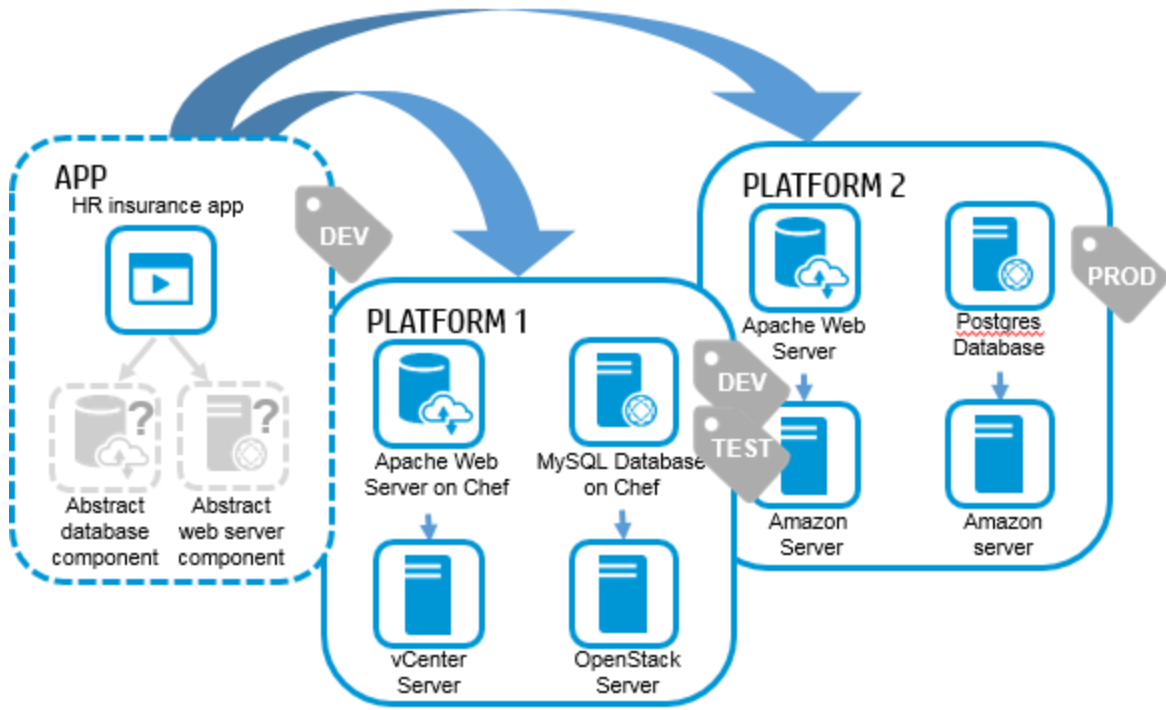
An application design can be of two types:

- Complete design: all components in this design exist for fulfillment
- Partial design: this design requires another design for fulfillment

Topology composition is used to compose the application design with the infrastructure design at run time. During application deployment, the infrastructure need varies for each deployment; topology composition helps in defining these variable infrastructure needs in the application design and allows to compose with different infrastructure designs at deploy time.

The capabilities and characteristics are used to describe the components. The application design will define the requirements using the capability components and characteristics in the design. The application design cannot be provisioned on its own and requires the selection of a compatible service design. The service design components are matched for their capability and characteristics to check the compatibility and the matching designs are chosen as compatible service design during the deployment.

The following illustration shows the topology composition for an HR insurance app. The app requires a database component and web server component, which are defined in the application design APP. This is fulfilled by PLATFORM1 as it has the Apache Web Server which has the web server capabilities and its characteristics and MySQL database, which has the database capability and its characteristics. Similarly the PLATFORM2 also matches the APP requirements.



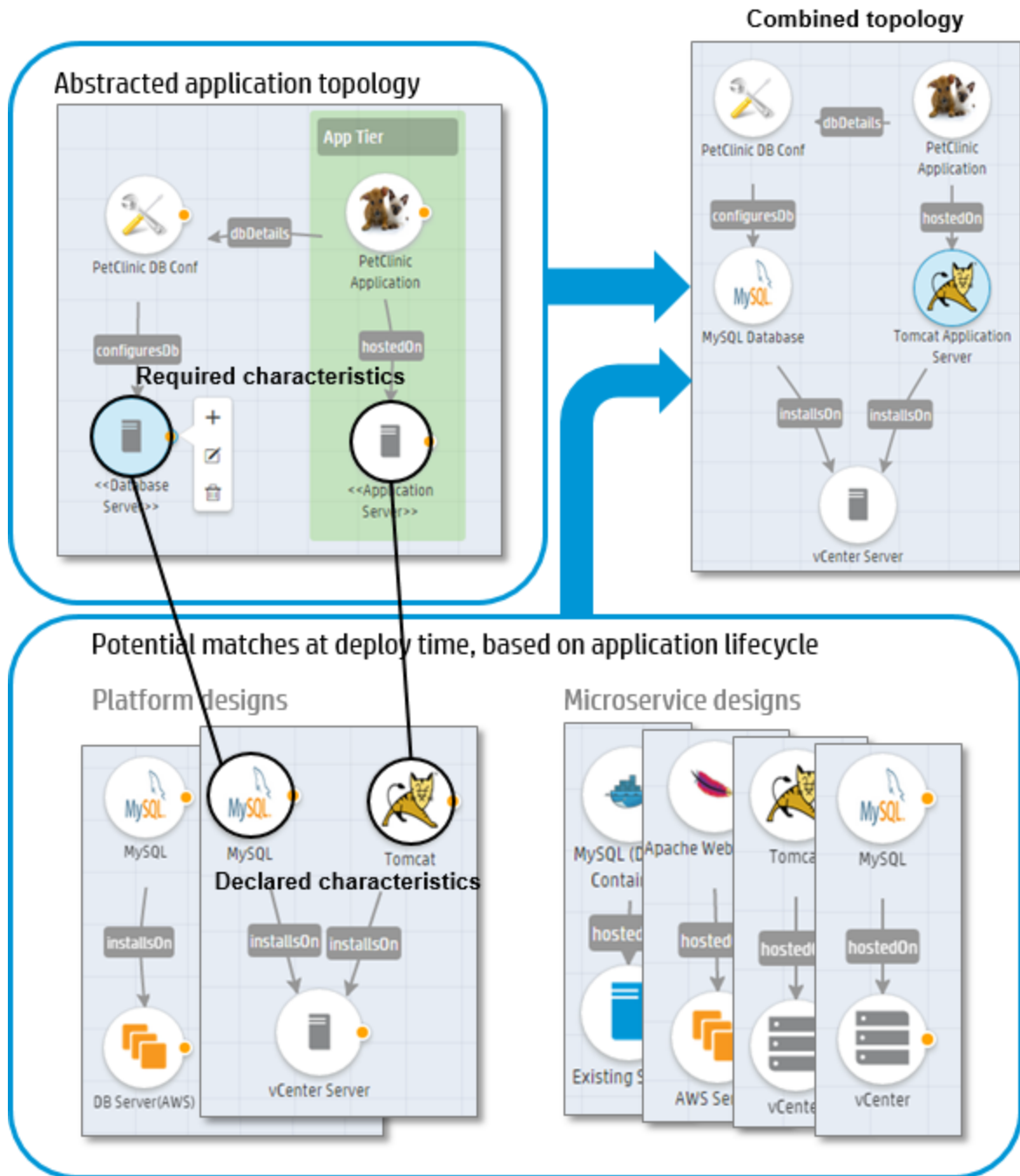
Microservices

A partial application design can be deployed using multiple infrastructure designs that provide the platform or infrastructure services rather than a single service. A partial design with multiple open requirements (capabilities) can compose with multiple infrastructure designs to satisfy all its open requirements and build a complete application design. You can choose either a single design that matches all capabilities, or you can choose components from different designs.

A partial design with multiple open requirements (capabilities) could compose with multiple (micro) service designs to satisfy all its open requirements and build a complete application design.

For example, if the application requires database and application services, it should be possible to select a design which has database and application in a single design or you can choose a database from one service design and an application from another service design.

A combined topology is created at run time based on microservice selection. The microservice can be associated with lifecycle stages.



Application pipeline management

Automating the deployment of applications is a complicated and lengthy process and requires significant investment. Applications are deployed differently in development and in production, causing many errors. Application pipeline management allows you to deploy applications across different environments using the same topology model. You can choose different microservices in different stages; however, the application design remains the same. This means that the same design is deployed and tested across different lifecycle stages.

You can also customize your release pipeline and have each application team use a separate lifecycle stage. This enables a fully automated and continuous deployment. Codar increases the agility of application release cycles while increasing the quality and reducing the cost of application deployments by eliminating manual steps.

Pipeline management in Codar includes:

- Creating your own roles thus enabling you to create your own user access structure
- Creating your own lifecycle stages in addition to the out-of-the box stages
- Selecting resource environments that already exist and associating them with only certain lifecycle stages thus creating a lifecycle stage superset comprising a subset of pre-defined lifecycle stages
- Viewing pipeline statistics and getting a visual representation of your deployments
- Filtering your view based on packages, actions, and environments

APPLICATION : Pet Clinic

 Application Architect **Creates Application**

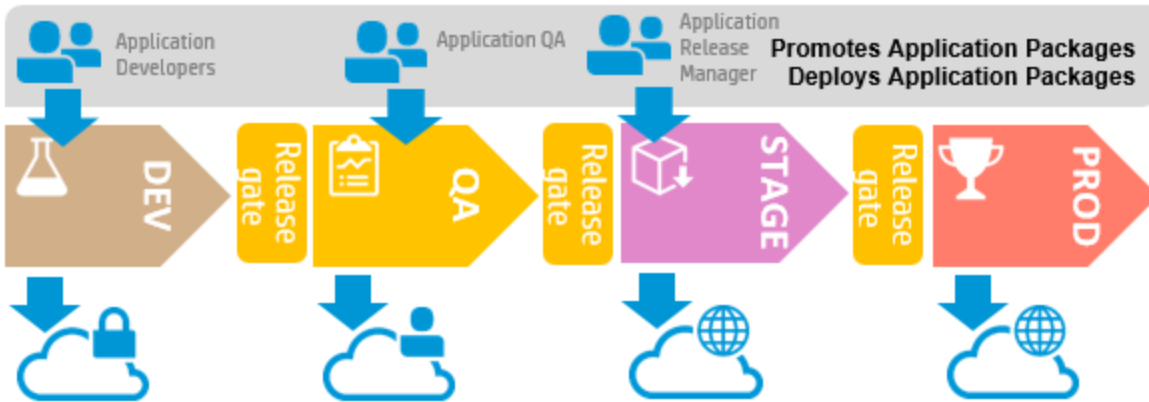


APPLICATION VERSION : 1.0.0

 Application Architect **Creates Application Topology**



PACKAGE (Build) : 333



Managing packages

Packages represent a snapshot of an application design and allow properties to be parameterized within the design. We can also say that the package represents a particular build of an application.

A package is the smallest unit that can be deployed for an application. It represents both the implementation artifacts (the manner in which an application should be deployed) and deployment artifacts (the location of libraries like war, ear, etc., that should be deployed).

Packages are associated with a lifecycle stage. A package can belong to Development, Testing, Staging, or Production stages.

Packages are associated with pipeline management. They can be managed across lifecycle stages, such as promotion or rejection in a given stage. For example, a user with the QA role can reject a package. See "[Lifecycle stages and actions](#)" on page 27.

Tasks

- **Create a package from a specific application version.** An application version can consist of multiple packages. See "[Create a Package](#)" on page 1.
- **Deploy or redeploy a package.** In this case the corresponding state of an application design along with the properties of the design specified in the package will be fulfilled. See "[Deploy a Package](#)" on page 1.
- **Delete a package.** Go to the Release Pipeline tab, press and hold down Ctrl to select multiple packages, and click **Delete**.

To delete a package

- a. Click **Release Automation >> Release Pipeline**.
- b. From the drop-down list on the left side of the screen, select the type of application design from which you need to delete a package (**Topology Designs** is selected by default).
- c. Click the application design version, which contains the package you need to delete. All packages pertaining to their respective lifecycle stage for the selected application design version are displayed.
- d. Click the package you need to delete.
- e. Click the gear icon pertaining to the package you want to delete and select **Delete**.
- f. In the **Confirmation Required** dialog, click **Yes** to confirm package deletion.

Note: You cannot delete a package if:

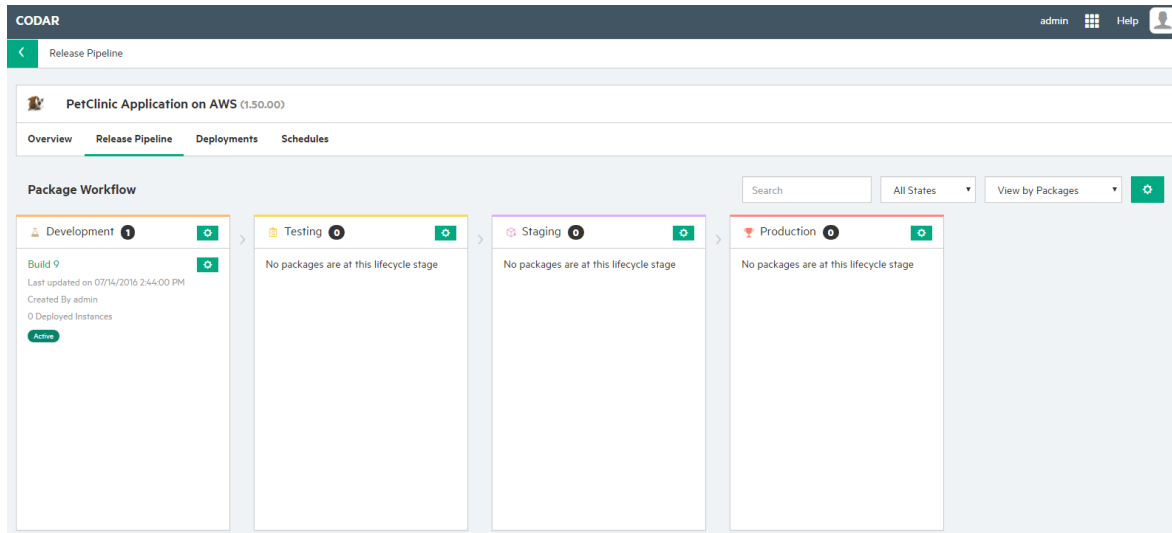
- it has an action instance associated with it.
- it is in the last stage of the lifecycle. Such a package must first be rejected before it can be deleted.

Package operations

Codar is a centralized structure for implementing a DevOps environment. Different roles can deploy, redeploy, promote, or reject the packages. Packages are promoted from one stage to another in a consistent and repeatable manner. This ensures visibility to team members when their applications are pushed into production.

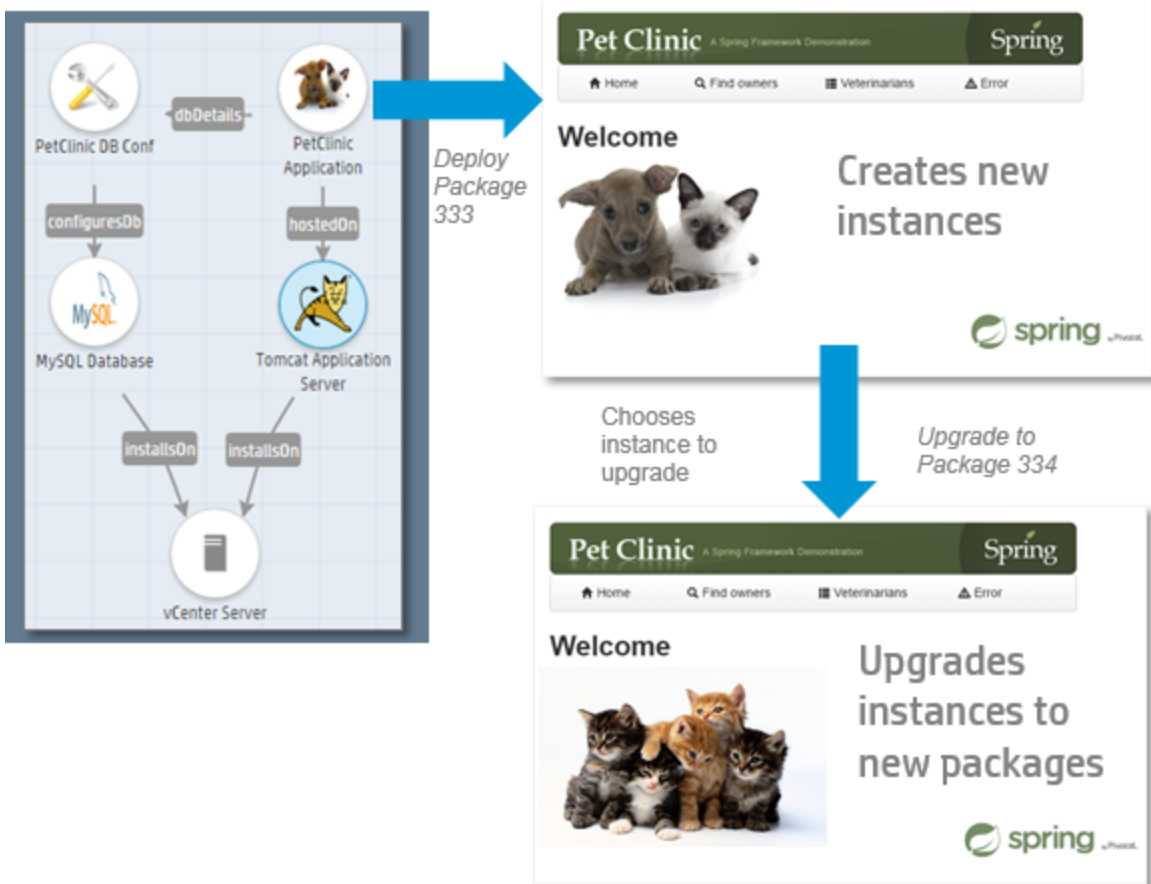
When packages are created and deployed, new virtual machines will be created and packages will be deployed. You can execute tests on a deployed instance, and the package can be either promoted or rejected.

Codar facilitates application pipeline management capabilities, as shown in the following image.



Deploy and redeploy

A package can be redeployed on an instance that has an older package. You can view instance details and pick an existing instance. Redeploy can also be used to upgrade or patch a component. Because redeploy invokes a modify action for all components, all components in a design can be upgraded to new versions.

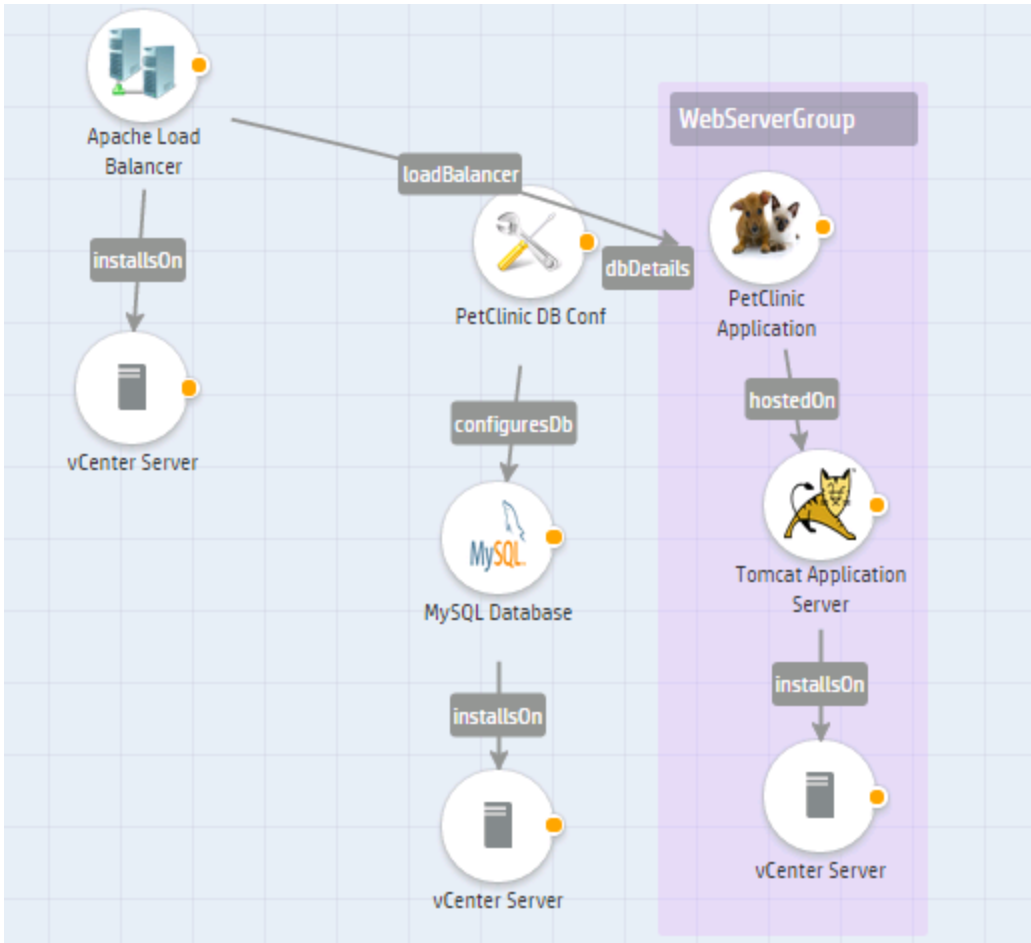


Scale out

During Topology design creation you can create a scaling group. A scaling group represents a scalable stack. There can be multiple scalable groups in an application design.

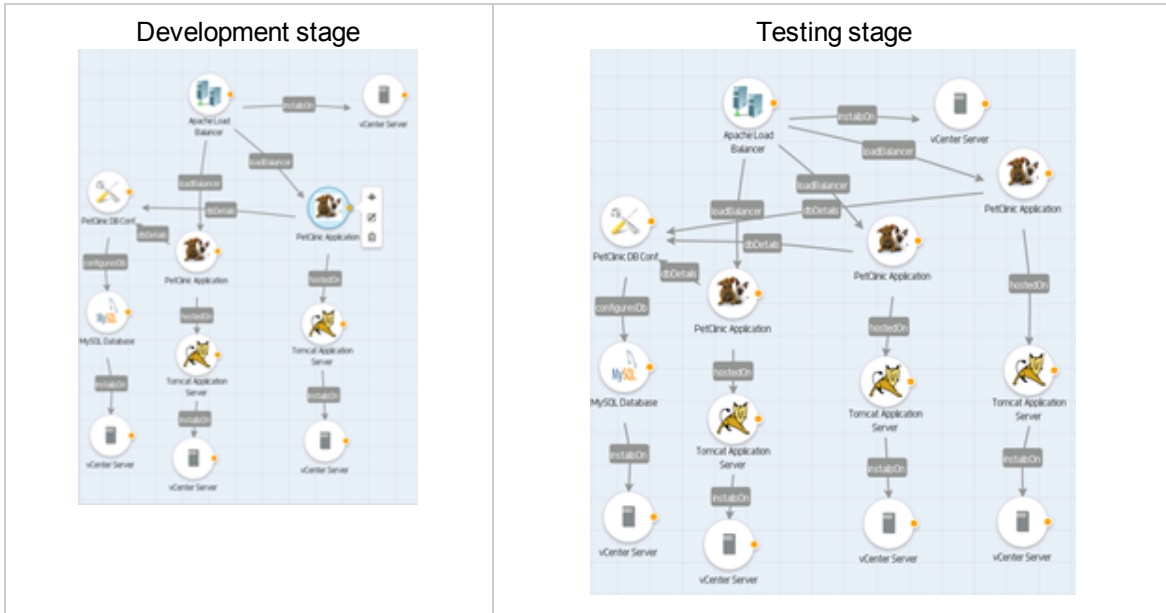
You can scale out after the deployment is complete. When you scale out, the full stack is replicated.

For example, the image below shows the web tier as a logical group named webServerGroup.



This group was scaled in the Development stage to one group and in Testing it is scaled to two groups in the image below.

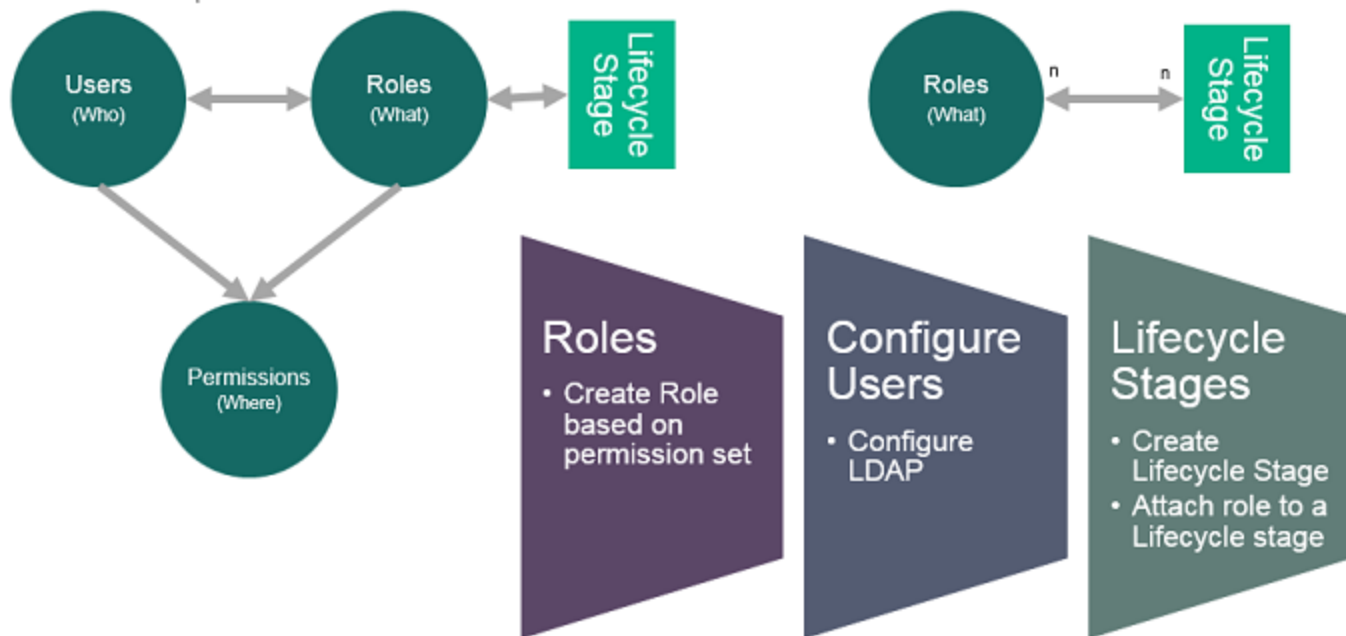




Roles and user access

User access can be configured for topology designs. Both users and LDAP groups can be added to the designs. An application architect can create a design and either make the design public or restrict it to certain users.

In Codar, user access comprises roles and permissions. Every user in Codar is assigned one or multiple roles. Every role is assigned one or more permissions. Therefore, users belonging to a particular role have all the permissions defined for that role. Codar contains some out-of-the-box roles; however, users can also create their own roles and then assign permissions to the roles they create. For information about how to create custom roles, see the *Codar Online Help*.



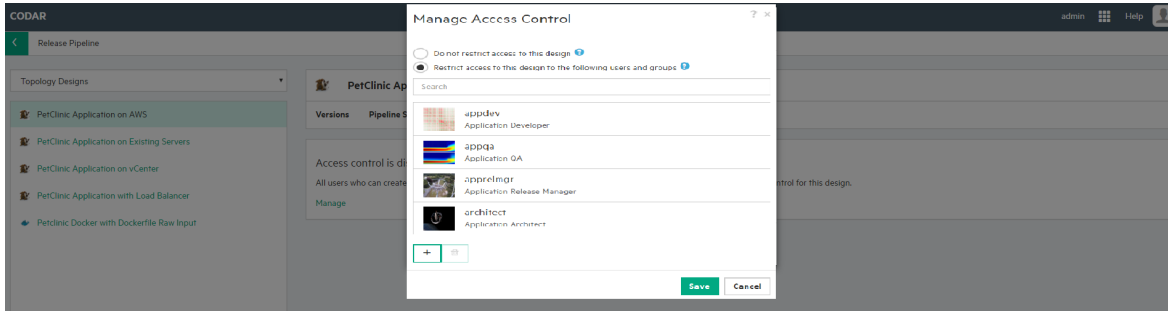
The administrator and the application architect can configure users and groups:

- Users for each role are defined at the application level for a granular level of control.
- Groups should represent application teams to automatically assign roles for the application.

The following image shows a design configured for various users including an application architect, developers, QA, and release manager.

Get started

Roles and user access



Lifecycle stages and actions

Lifecycle actions contribute to the initial deployment of a service, communicating with the service provider through a process engine such as HP Operations Orchestration. Lifecycle actions also provide other important functions, such as actions required to modify the service upon request or actions required to remove the service from deployment.

Every lifecycle comprises a stage and every stage has roles associated with it. It means that all users who belong to the roles in a particular lifecycle stage can perform operations defined in the role during that stage. For example, if the Development lifecycle stage has the Application Architect role associated with it, then users belonging to the Application Architect role can perform tasks associated with the role in the Development lifecycle stage.

The following are the out-of-the-box lifecycle stages available in Codar:

- **Development:** This is usually the first stage in which the code is developed and application artifacts are created.
- **Testing:** Stage in which test cases are executed against the code developed in the Development stage.
- **Staging:** Pre-production stage that replicates the production environment; used to test the code and artifacts.
- **Production:** This is usually the final stage in which the application is deployed in a live environment.

Apart from the out-of-the-box lifecycle stages, there are custom stages that you can create. For information about creating, editing, and deleting custom stages, see the Codar Online Help.

The following table lists the actions pertaining to a package that take place in each lifecycle stage:

Stage	Promote	Deploy, Redeploy	Edit	Delete	Reject
First stage (usually the Development stage)	Yes	Yes	Yes	Yes	Yes
Intermediate	Yes	Yes	Yes	Yes	Yes
Final stage (usually the Production stage)	No	Yes	Yes	Yes	Yes

You can access lifecycle stages by using the **Release Automation > Pipeline Configurations** sidebar menu item.

Use the following actions to deploy or move a package through the stages. These actions describe the flow when release gate actions are not defined.

- **Promote:** Moves the package to the next lifecycle stage. The package state remains Active.
- **Deploy, Redeploy:** Deploys the package. See ["Deploy a Package" on page 1](#).
- **Edit:** Changes the properties of a package. See ["Edit a Package" on page 1](#).
- **Reject:** Stops the package from advancing to another stage. The package will remain in its current stage, its state will be set to Rejected, and the action buttons will no longer be available.
- **Delete:** Deletes a package. The package will be removed permanently from the system.

Note: A package can only be deleted if all associated deployed instances are canceled and deleted.

- **Refresh:** Retrieves current package status.

Package states

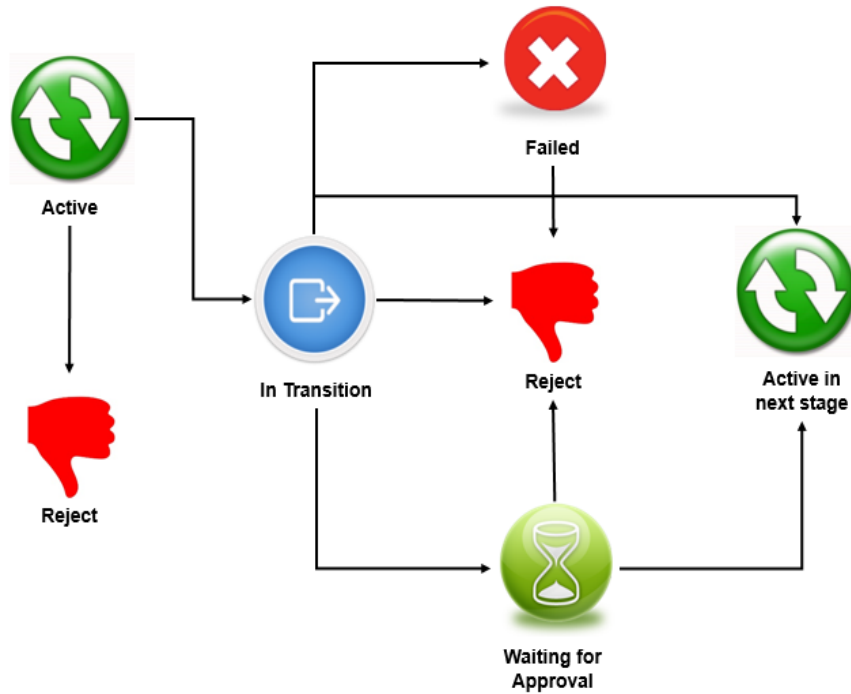
Packages have the following states:

- **Active:** the package is active in the current lifecycle stage
- **Rejected:** the package has been rejected and will not move to the next lifecycle stage
- **Transition:** the package is in transition to the next lifecycle stage
- **Failed:** the promotion of the package has failed

If you reject a package, then it remains in its current stage, its state is set to Rejected, and no further actions can be applied; however, it can be deleted and is removed from the system.

When a package is promoted, it moves to the next stage and remains in the active state. Packages are always created in the first lifecycle stage. If the Codar Jenkins plug-in is configured, then after a successful build the Jenkins plug-in talks to Codar and creates a package.

Package States



Grouping service designs by lifecycle stage

A partial design with an active package requires you to select a service design to provision in the deploy package wizard. These service designs can be grouped for different lifecycle stages. This grouping enables package deployment in a lifecycle stage to list only those grouped service designs from that lifecycle stage.

To group the service designs for a lifecycle stage, create a tag with the name of the lifecycle stage in each topology design. When a lifecycle stage is created, a tag with the lifecycle stage is automatically created. Hence, only the designs need to be associated with the right lifecycle stage tag. For example, you could create a Development tag and associate it with all required designs in the Development lifecycle state.

Note: The test run wizard in the Test tab lists all designs and does not group by tag.

Release gate actions

Release gate actions are actions that are user-defined and act as a promotion request check between two lifecycle stages. Only if a package passes through each enabled action and the status of all the actions in a lifecycle stage is successful, is the package promoted to the next stage.

Release gate actions are of six types:

- Deploy Application

This action deploys a partial or complete application design based on pre-defined Operations Orchestration (OO) content packs. The deploy action can be configured such that an email message is sent to users who initiated the promotion request notifying them of the promotion success or failure. Users can even choose to reject the package and clean up the deployment if the deploy action fails to execute and the package has not been promoted.

- Run OO Flow

This action executes the specified flow in Operations Orchestration. Typically, this action can be used to execute specific tests, with or without a deployment instance.

Custom actions can be configured such that an email message is sent to users who initiated the promotion request notifying them of the promotion success or failure. Users can choose to either reject the package or proceed with the package promotion if the custom action fails to execute.

- Add Approver

This action promotes a package only if designated approvers manually approve or reject a package promotion. An approval action can also be configured to automatically approve or reject a package promotion.

- Execute Test Set

This action can be used for executing specific tests with or without a deployment instance.

- Run Script

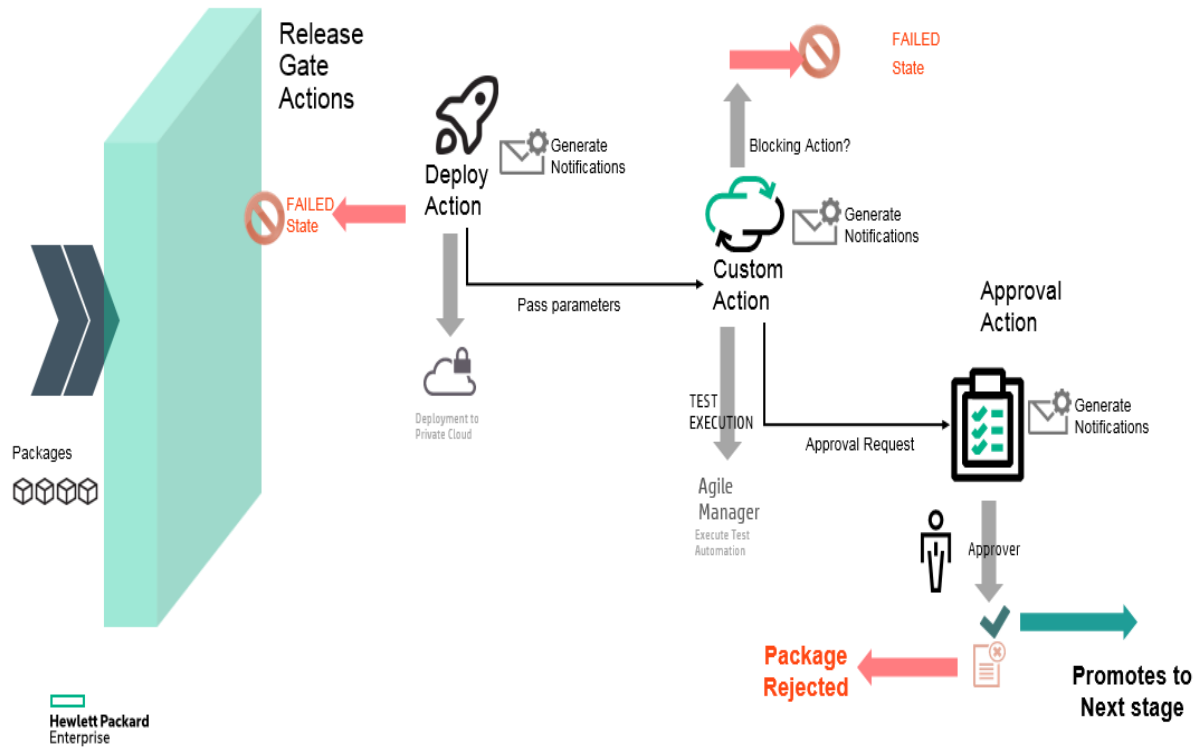
This action can be used to execute a script on a remote machine, with or without a deployment instance.

- Consume CMP Service

This action can be used to execute cloud offerings from CSA. After an offering is realized, the same can be used to deploy applications.

For detailed information about creating, editing, and deleting these release gate actions, see the *Codar Online Help*.

The following figure is a representation of the way in which release gate actions work:



Pipeline statistics

The Pipeline Statistics tab displays detailed information about packages and includes graphical representations of package summary, trends, states, deployment status and so on. It provides a holistic view of all packages and deployments and enables you to make informed decisions with respect to package deployment.

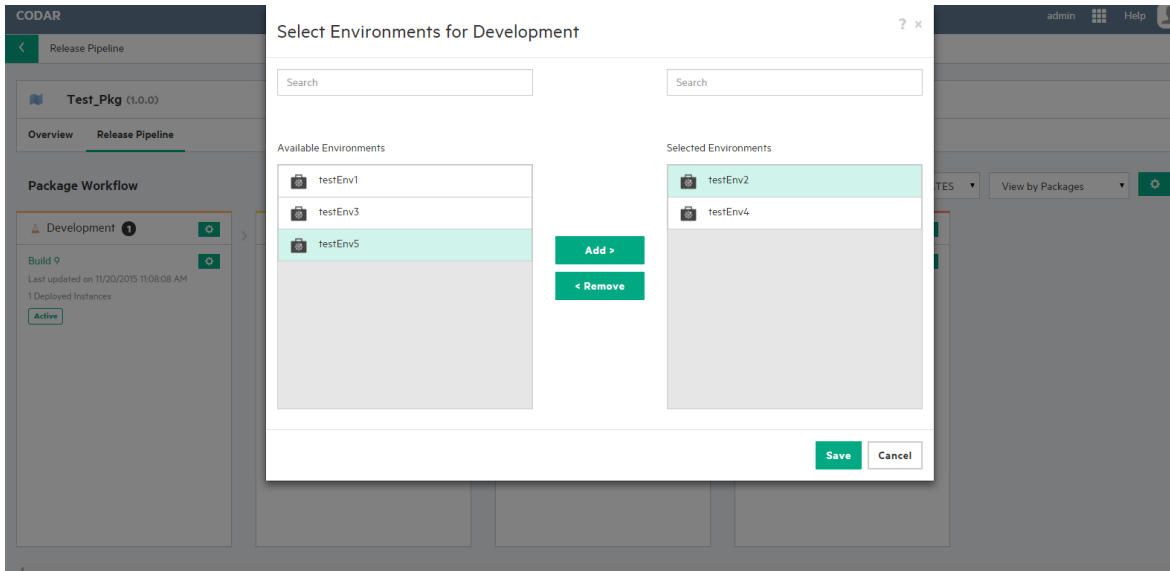
It displays information about the number of packages created on any date, the number of successful transitions, the number of deployments and so on.

For detailed information about the pipeline statistics, see the *Codar Online Help*.



Environments

You can select different environments for each lifecycle stage at the application level. For example, you can configure vCenter for deployment but a public cloud environment for staging.





Schedule releases

You can schedule and govern the transition of packages across release pipeline. You can set up the following schedules:

- Automatic promotion of packages from one lifecycle stage to another
- Automatic purging of packages and deployments periodically


You can view the schedules in a calendar view to get visibility of scheduled package promotions across lifecycle stages.

 Package1

Overview Configure  Deployments **Promote Scheduler** Release Gates

Promote Scheduler

Promotion Stage	Schedule Name	Schedule Date	Scheduled By	Description
Development	Development To Testing	06/06/2016 12:30:00 PM	admin	Promote the package from Development to Testing
Testing	Testing to Staging	06/07/2016 10:00:00 AM	admin	Promote the package from Testing to Staging on th
Staging	Staging To Production	06/08/2016 10:30:00 AM	admin	Promote the package from Staging to Production o



Overview Release Pipeline Deployments **Schedules**

Month Day July 2016 Today

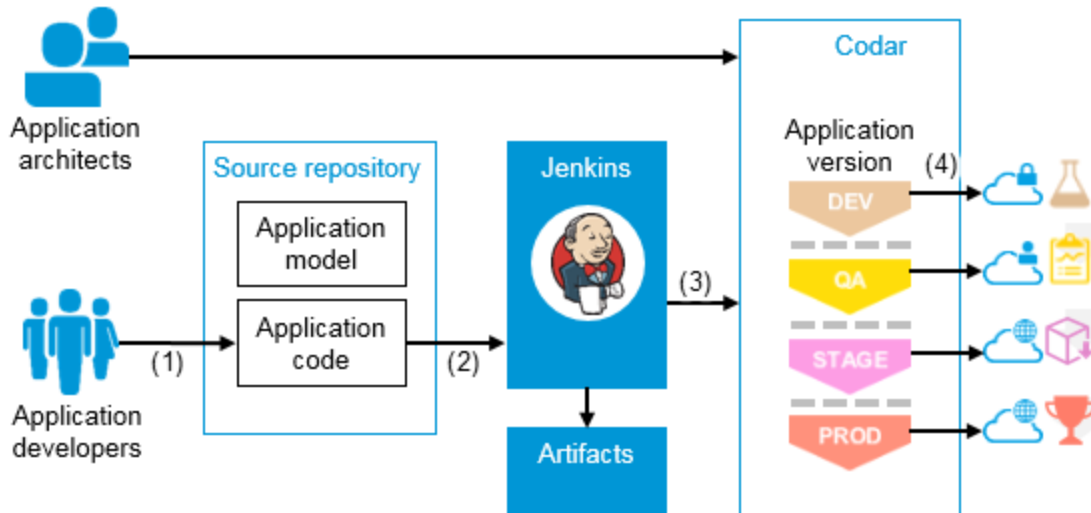
Sun	Mon	Tue	Wed	Thu	Fri	Sat
26	27	28	29	30	1	
3	4	5	6	7	8	
10	11	12	13	14	15	
17	18	19	20	21	22	
24	25	26	27	28	29	
31	1	2	3	4	5	

External integrations

Codar is open and extensible, and can be integrated with different build systems such as Jenkins, Hudson, etc. A comprehensive set of REST APIs can be used with other external tools to achieve continuous integration, deployment, and delivery. The Codar architecture also provides options for you to hook into customized flows for DevTest and DevOps.

Jenkins integration

Codar includes a Jenkins plug-in for continuous deployment. The following illustration shows how it works.

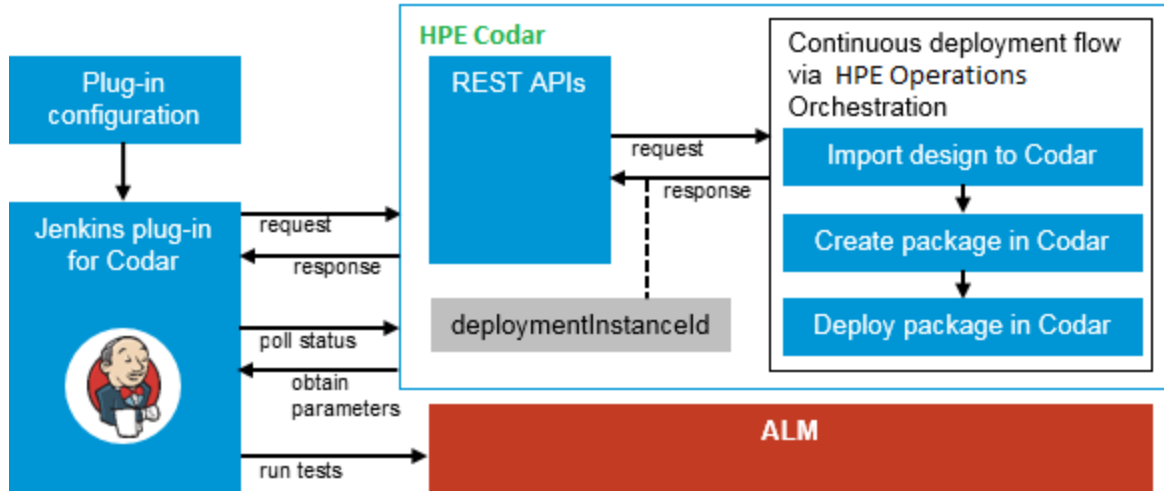


1. Developers check in changes.
2. Continuous integration triggers build.
3. Jenkins plug-in creates and deploys package.
4. Application is deployed to different environments depending on lifecycle stage.
5. In case of continuous promote, packages are moved to the final lifecycle stage if all the release gate actions are executed successfully.

Note: Similar kind of OOTB integration is also available with Bamboo and TFS.

ALM integration

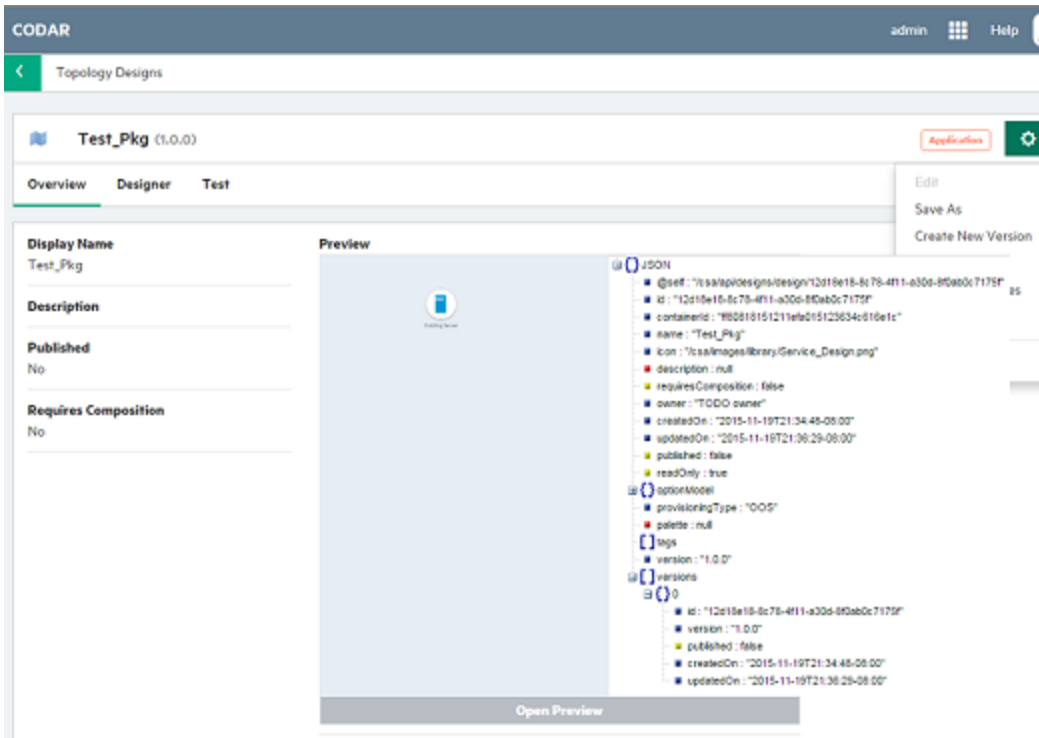
You can integrate Application Lifecycle Management (ALM) with Codar to run tests after successful deployment. The following illustration shows how Jenkins acts as an orchestrator.



Infrastructure as code (IaC)

Managing infrastructure as code (IaC) allows IT teams to leverage the best practices for developing code, such as code reviews and unit testing for how infrastructure and applications get provisioned.

Codar can manage infrastructure as code. Topology designs that can contain server configurations, networks, volumes, relationships, and application-specific details like the application version and package information can be exported in JSON format and managed with the application in the source control system. Developers can make changes to the model using a text editor and use it for automation. The modified model can also be imported back into Codar.



Deploy containers on Docker Cluster

Docker Universal Control Plane provides an on-premises, or virtual private cloud (VPC) container management solution for Docker app.

You can do the following using Docker UCP.

- Configure Docker UCP as a Resource Provider within Codar.
- Create designs based on the Docker content.
- Deploy containers on Docker UCP and perform Pipeline management.

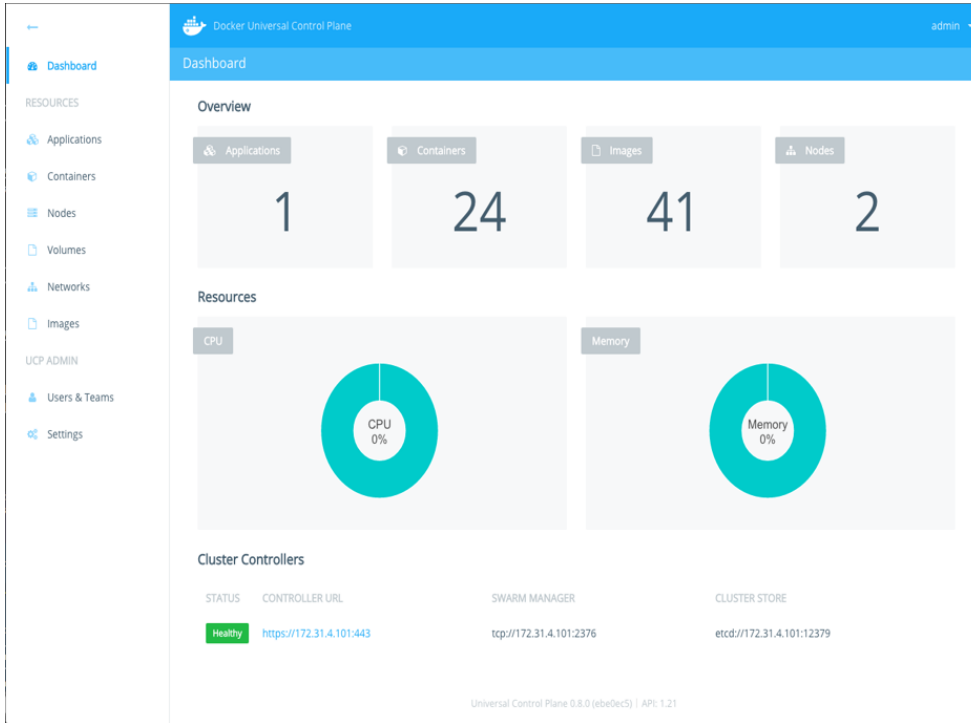
Docker Universal Control Plane



Docker UCP Dashboard

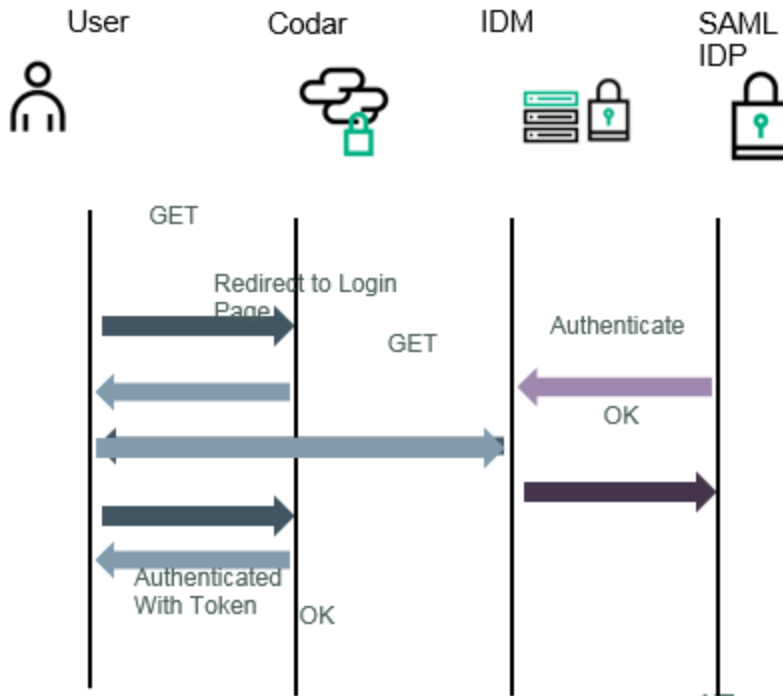
Get started

Deploy containers on Docker Cluster



SAML support for Codar

Codar provides SSO with SAML. Authenticate only once (using your Identity Provider with SAML support) and subsequently use Cloud Service Automation without having to type in a password again.



Use cases

This section contains information related to the following use cases:

- ["Continuous integration, deployment, and delivery" on page 45](#)
- ["Customizable release pipeline" on page 53](#)
- ["Deploy and redeploy packages" on page 55](#)
- ["Deployment and scale out" on page 57](#)

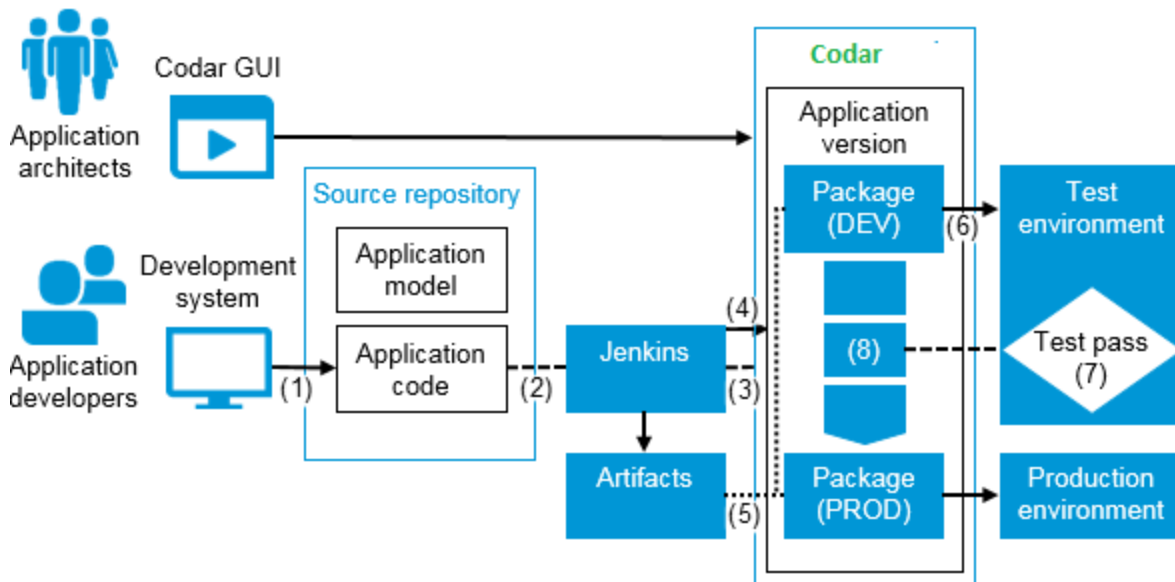
Click the respective links for more information.

Get started
Use cases

Continuous integration, deployment, and delivery

The goal is for an application to be enabled for continuous integration (CI) and continuous deployment. An application developer codes the application and an application architect models the application in the Codar interface and then exports the application model as code (IaaS). When the application developer checks in the code, a Jenkins build is triggered and the application is deployed using the application model on a specific environment. After the application is deployed, the continuous deployment process is extended to continuous delivery whereas application-specific tests can be automatically run on the deployed instance, with the application possibly being deployed to a different environment dependent on the outcome of the tests.

The following section describes how Codar achieves this scenario.



Application modeling

Application architects model applications graphically by including the necessary components of the design in the designer interface and connecting them via relationships. Codar contains a palette of standard components, and components can be imported (embraced) from various deployment engines such as HPE Operations Orchestration and Chef. Such designs, called application models, are representations of the methods in which applications are to be deployed. An application model can be exported in JSON format and managed in an external source repository, achieving infrastructure as code (IaaS).

Continuous integration and deployment

In continuous integration, the code for the sample application and the model for the deployment of the application (in JSON format) is available in a source repository.

When an application developer makes a code change to the application and checks it into the source repository (1), Jenkins triggers a build (2).

Codar provides a Jenkins plug-in which has details such as the IP address, user name, and password for Codar. It establishes a connection and invokes an API as part of a post-build step (3). The API then invokes a workflow that executes various actions for achieving continuous deployment and continuous delivery.

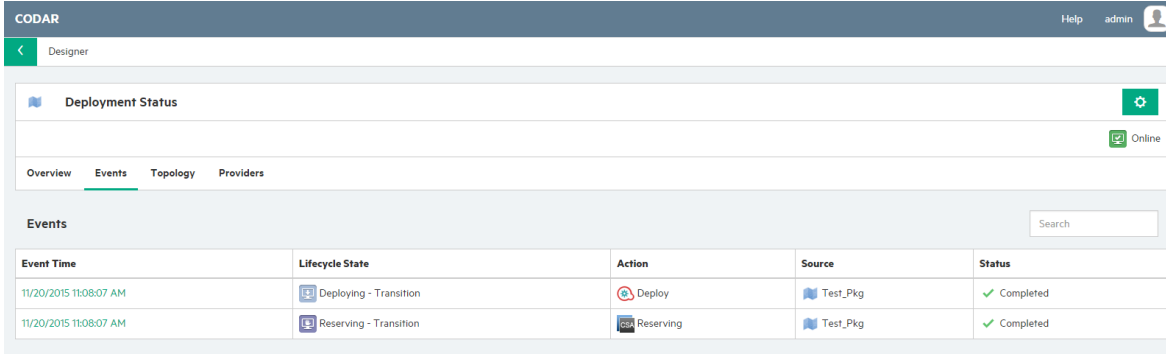
Importing an application design

If the application model has not already been imported into Codar or if it has changed, the continuous deployment workflow imports it, in JSON format (IaaS), into Codar (4) as a new version of the application design. This allows changes that have been made by application developers and architects to be taken into consideration at the time of deployment.

It is important to note that if the application model has already been imported or if there is no change in the application design, then this import operation does not take place and the application version within Codar remains the same. You can view the application model in the Topology/Sequenced sidebar menu item in the designer.

Deploying on an environment

After the package is created, the continuous deployment workflow fulfills the application design based on the environment (6). You can view deployments for the package on the Deployments tab:



The screenshot shows the CODAR Designer interface. At the top, there is a header with 'CODAR' on the left and 'Help admin' on the right. Below the header is a navigation bar with a back arrow and 'Designer'. The main content area is titled 'Deployment Status' and includes a settings gear icon and an 'Online' status indicator. Below this, there are tabs for 'Overview', 'Events', 'Topology', and 'Providers'. The 'Events' tab is active, showing a table of deployment events. A search box is located to the right of the table.

Event Time	Lifecycle State	Action	Source	Status
11/20/2015 11:08:07 AM	Deploying - Transition	Deploy	Test_Pkg	Completed
11/20/2015 11:08:07 AM	Reserving - Transition	Reserving	Test_Pkg	Completed

A runbook automation engine creates an execution plan based on the design that fulfills the infrastructure layer, platform layer, and application layer. You can monitor the status of the deployment of a particular package and view a graphical representation of the deployed application, which includes component-level properties and actions.

Publishing a design

Publishing a design makes it available as an offering to service consumers. You must have a CSA license installed before you can publish a design.

A complete design with an active package in the Production stage contains package-specific properties as part of the design and can be published.

A partial design with an active package in the Production stage contains package-specific properties as part of the design, but it cannot be published until a final composed design is created by deploying the production package.

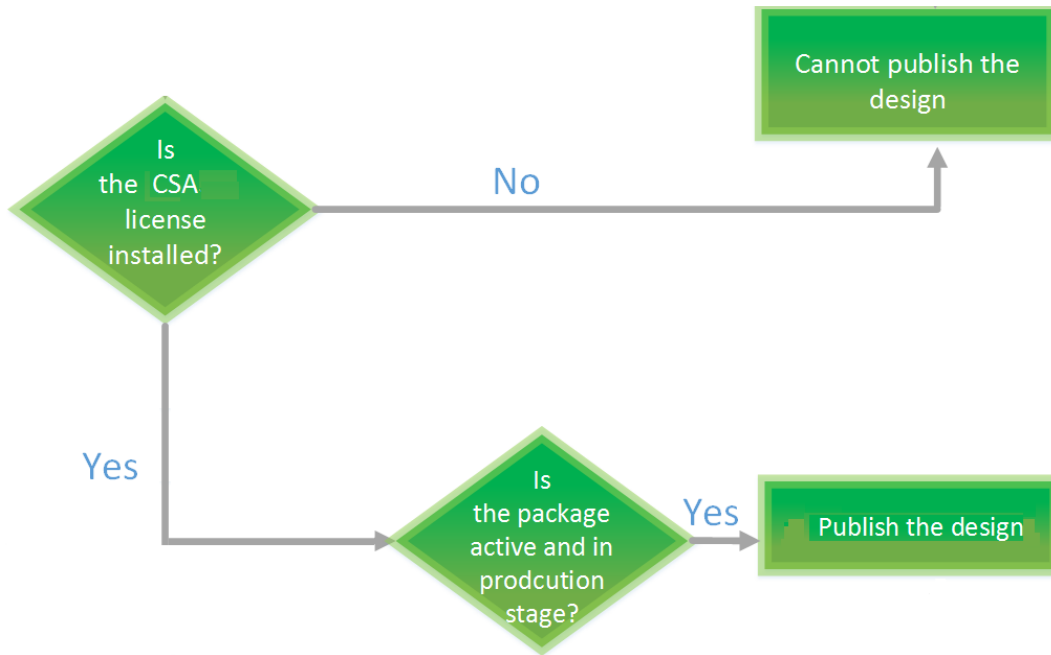
Publishing a partial design is different depending on which licenses you have installed:

- A Codar application design that has been advanced to the Production stage is deployed on a production infrastructure, and then the composed production design is made visible on successful production deployment. The design can then be published to service consumers.
- A design that is not a Codar application design must be saved as a composed design from the Test tab. The design can then be published to service consumers.

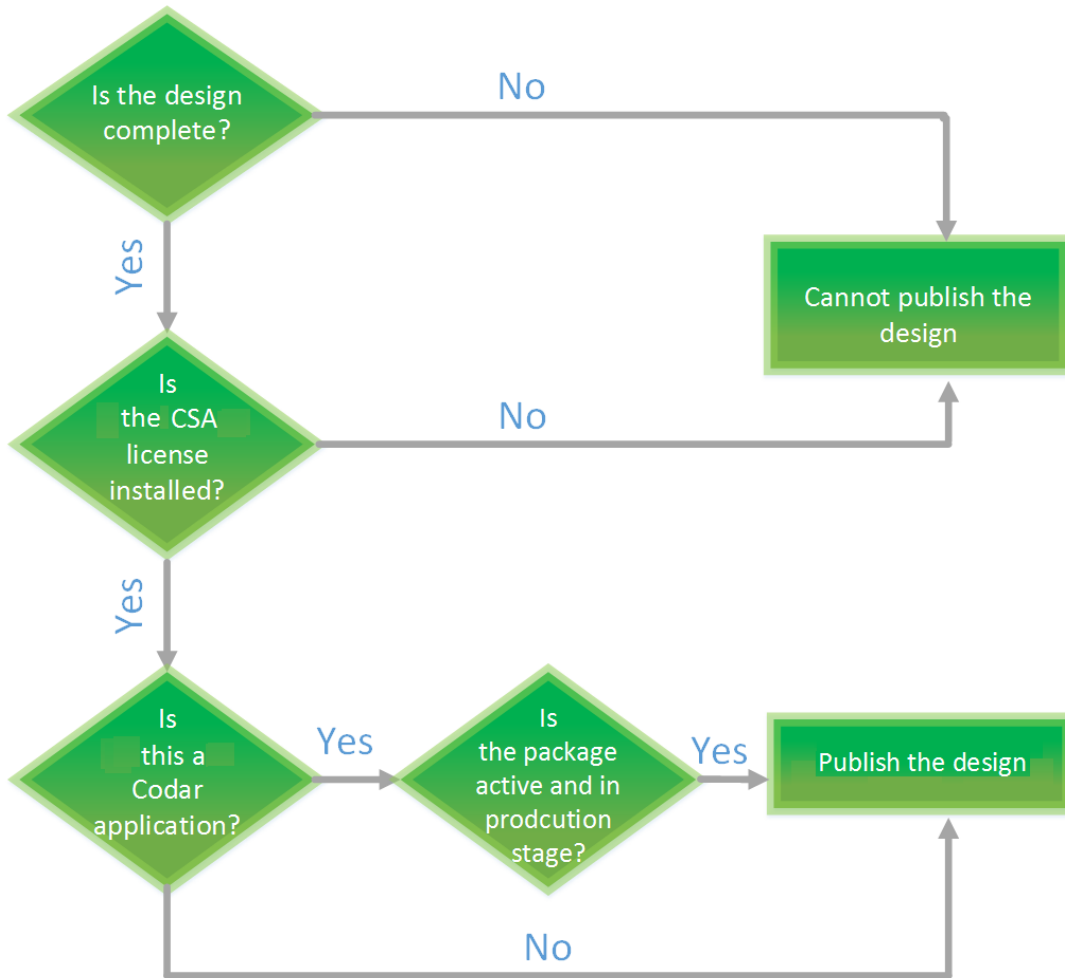
Note: A sequenced design can be published only if it has no packages or if the package is in the last stage of Pipeline Management, with an active deployment instance.

If a topology design has been tagged with an application, it cannot be published unless it is untagged.

The following figure illustrates when a sequenced design can be published based on the license used:



The following figure illustrates when a topology design can be published based on the license used:



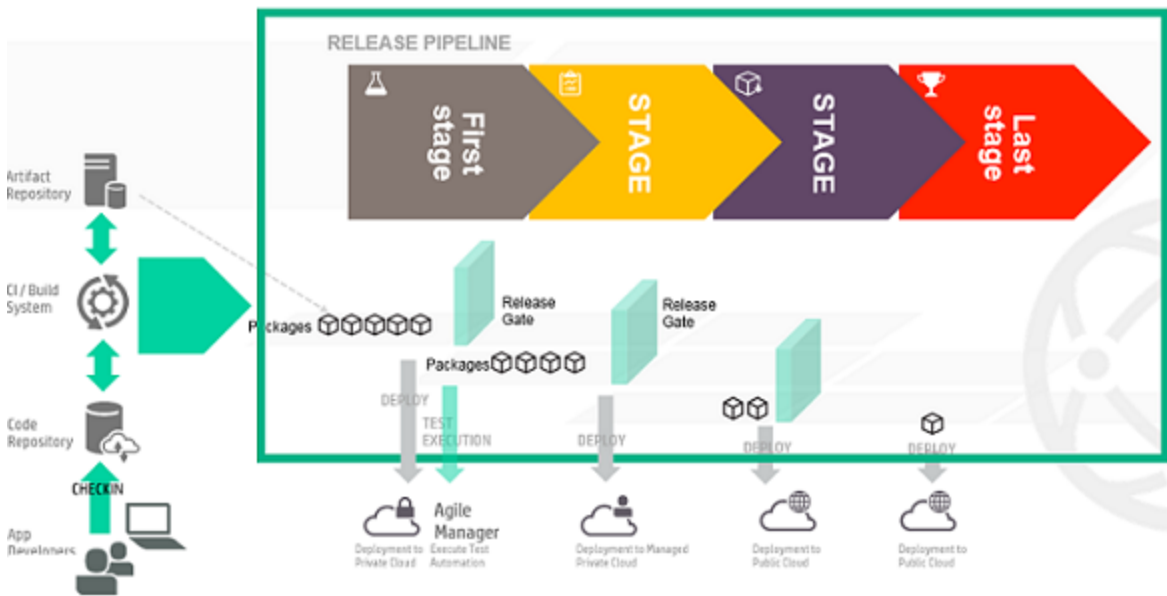
Customizable release pipeline

The goal is to allow users to build their own custom release pipeline and have each application team define and use separate lifecycle stages. The following are the high-level steps for users to define their own release pipeline. For information about how to perform these steps, see the *Codar Online Help*.

1. Create roles and add permissions
2. Create lifecycle stages and associate roles with each lifecycle stage
3. Add lifecycle stages to the application design
4. Add release gate actions to each lifecycle stage
5. Create packages using the continuous promote API

The package is moved to the last lifecycle stage if all the release gates execute successfully.

The following illustration depicts the customizable release pipeline.



Get started
Customizable release pipeline

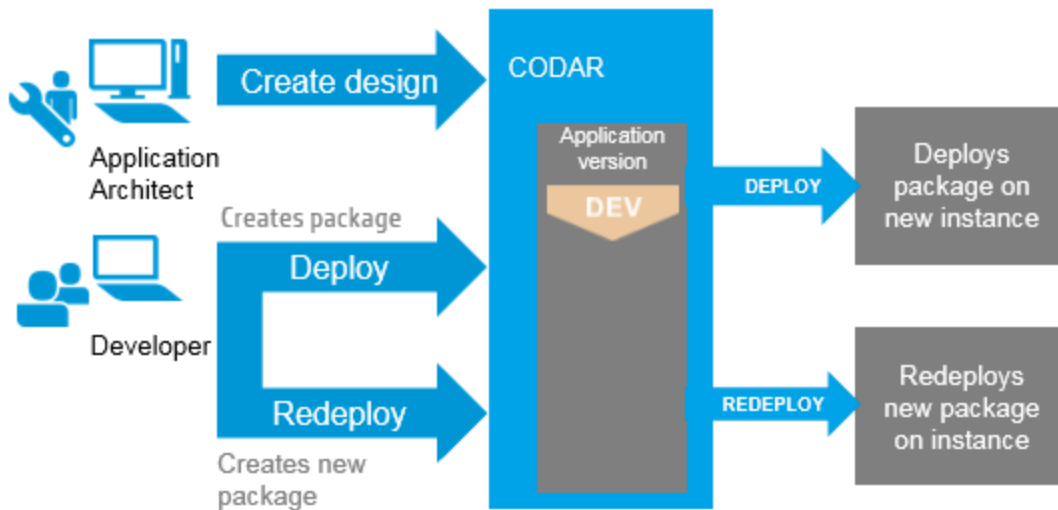
Deploy and redeploy packages

The goal is to deploy a package and then use the same instance to redeploy a newer version of the package on an instance that has older versions.

An application architect models the application and marks it for pipeline management. The developer will then create a package and deploy it. A new instance is created when the package is deployed. The deployment is based on the topology design.

After deployment, the same instance can be used to redeploy a newer version of the package. The instance can be upgraded or patched to newer packages or builds.

The following diagram shows the process:



Get started

Deploy and redeploy packages

Deployment and scale out

The goal is to create a scalable stack and scale out the stack on demand after deployment. An application architect models the application and marks the application for pipeline management. The architect identifies the components that should be scaled out in different life cycle stages. The scalable stack can contain a single component or group of components. During development, the stack can be scaled in by one, and then during testing the stack can be scaled to two, and so on.

Quickstart

This section provides steps that will help you deploy a sample application called "PetClinic" using Codar.

- [Prerequisites to using the PetClinic application](#)
- ["Downloading files" on page 60](#)
- [Accessing the application design](#)
- [Customizing components](#)
- [Deploying the application design](#)
- [Troubleshooting](#)

Prerequisites to using the PetClinic application

The following are the prerequisites required to use the PetClinic application:

- In order to use this guide, Codar 1.90 must already be installed.
- You must have Ubuntu operating system version 12.04 or later for the PetClinic application to be deployed successfully. Ensure that your Ubuntu systems are clean and do not have Tomcat or MySQL installed.
- The application design to deploy the sample PetClinic application requires two Ubuntu systems.
- You can use the “admin” user role in Codar to perform all the steps in deploying the PetClinic application.

Downloading files

Perform the following steps to download the files and software required to deploy the PetClinic application:

1. Download the onboarding kit. The onboarding kit contains the necessary software and scripts required to install the PetClinic application.
2. To download the onboarding kit, click the following link: <https://hpln.hpe.com/product/hpe-codar/resources/file-repository>.
3. Click **HPE Codar Downloads>> OnboardingKit>> OnboardingKit** (zip file).
4. Extract the downloaded `OnboardingKit.zip` file in to a folder using WinZip or WinRAR software. This zip file contains the scripts to install Tomcat and MySQL software, to configure MySQL, and deploy the application on the Tomcat server. The URL to download Tomcat and MySQL server is available in this guide and these two software can be placed under the `software` directory inside the `OnboardingKitextracted` folder.
5. Download the Apache HTTP Server. The Apache HTTP Server is used to host the extracted artifact files. The Apache HTTP Server link which will refer to these artifacts is used to configure the parameters inside the Codar application design components. If you have any HTTP Server based repository, skip steps 4 to 8. Ensure that you place the downloaded files under the HTTP directory.
6. Extract the downloaded `httpd-2.4.12-win64-VC11.zip` file to a folder using WinZip or WinRAR software.
7. Open the `httpd-2.4.12-win64-VC11\Apache24\conf\httpd.conf` file.
8. Modify the `httpd.conf` file with the following steps to host the content of the folder you downloaded in step 1.
 - a. Replace the value of `ServerRoot "c:/Apache24"` with the extracted Apache HTTP Server location. For example, `ServerRoot "C:\httpd-2.4.12-win64-VC11\Apache24"`.
 - b. Replace the value of `DocumentRoot "c:/Apache24/htdocs"` with the location of the sample folder that you downloaded in step 1. For example, `DocumentRoot "C:/OnboardingKit"`.
 - c. Replace the value of `Directory "c:/Apache24/htdocs"` with the location of the onboarding kit folder that you downloaded in step 1. For example, `Directory "C:/OnboardingKit"`.
 - d. In the same `Directory` tag, change `AllowOverride None` to `AllowOverride All` and `Require all denied` to `Require all granted`.
9. Start the Apache HTTP Server by running the following commands:

- a. `<installation_drive>\httpd-2.4.12-win64-VC11\Apache24\bin\httpd.exe -k install`
- b. `<installation_drive>\httpd-2.4.12-win64-VC11\Apache24\bin\httpd.exe -k start`
where `<installation_drive>` is the drive in which the Apache the HTTP Server must be installed. For example, C:, E:

10. Launch the `http://localhost:80/` URL and the following folders and files are displayed:



11. Download MySQL and Tomcat from the following locations and copy them to the Software folder inside the OnboardingKit folder.

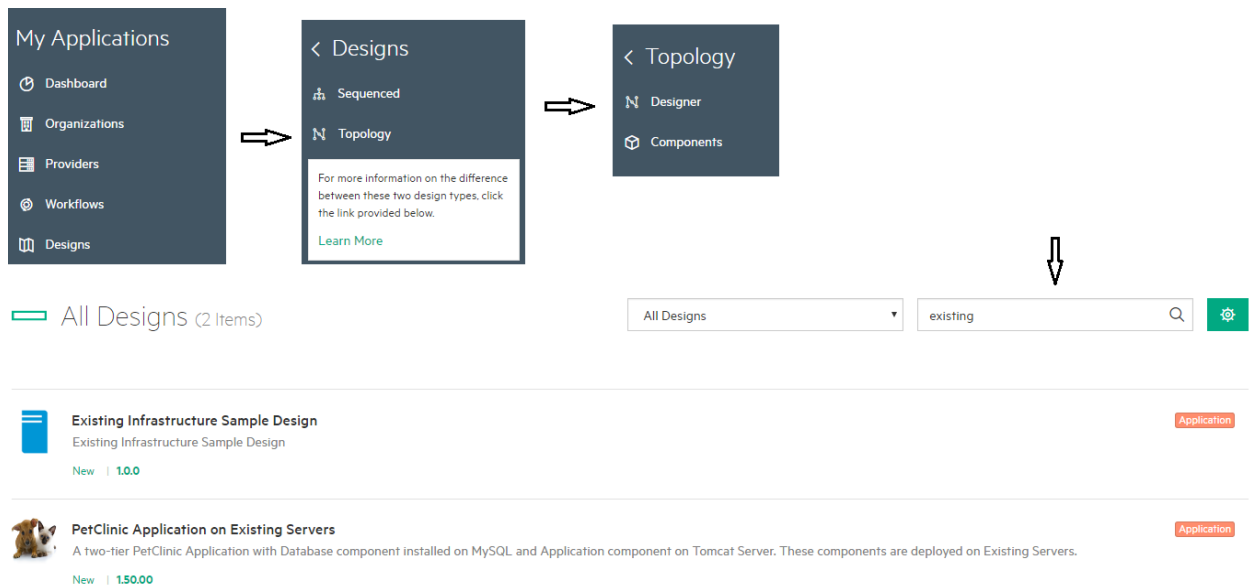
Software	Location
MySQL	<p>http://ftp.kaist.ac.kr/mysql/Downloads/MySQL-5.6/</p> <p>Download <code>mysql-server_5.6.22-1ubuntu12.04_amd64.deb-bundle.tar</code> or the latest version</p> <p>If you want to install the latest MySQL server, modify the <code>OnboardingKit/scripts/install_mysql.sh</code> script so that it contains the changed name and version of the MySQL server package.</p>
Tomcat	<p>http://tomcat.apache.org/download-70.cgi</p> <p>Download <code>apache-tomcat-7.0.56.tar.gz</code> or the latest version</p> <p>If you want to install the latest Tomcat server, modify the <code>OnboardingKit/scripts/install_tomcat.sh</code> script so that it contains the changed version and name of the Tomcat server package.</p>

12. Ensure that you have two Ubuntu systems with the following packages installed in them:
- o JDK 1.7
 - o `libaio1`
 - o `unzip`
 - o `zip`

Accessing the application design

In Codar 1.90, the application design is available out-of-the-box. You do not need to import it. This topic explains how to use the out-of-the-box Codar application design. The application design helps in deploying the PetClinic sample application.

1. Login to the Codar application portal at <http://<codarmc.mydomain.com>:8444/csa>
2. Navigate to **Designs -> Topology -> Designer**.



3. Click the application version, for example **1.50.00**, to configure the components.



4. Select each of the components and customize by specifying the required values for each of the components.

The application design has six components:

- Existing server 1 that can be used as the database server
- Existing server 2 that can be used as the application server
- MySQL database
- Tomcat
- PetClinic DB Conf - this component creates the database and configures it for the PetClinic application to run
- PetClinic Application

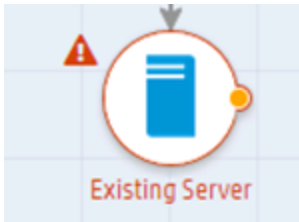
A database connection relationship is created between PetClinic DB Conf and the PetClinic Application components. The application design deploys the PetClinic application by performing the following steps:

1. Accesses the IP address of the existing servers
2. Deploys MySQL and Tomcat on the existing servers
3. Configures the PetClinic database on MySQL
4. Deploys the PetClinic application on the Tomcat application server

Customizing components

This topic explains how to customize the application design components in "[Accessing the application design](#)". It is assumed that you are *not* working in a virtual environment.

Customizing the existing server



Perform the following steps to customize the existing server component:

1. Click the existing server component.
2. On the **Properties** tab, enter the following information:

Field	Description
Name	Name of the server. This is typically the name of one of the two Ubuntu systems specified in Prerequisites .
HOST-FQDN	Complete FQDN name of the server
Modifiable during service creation	Select this check box if you want to update the FQDN during deployment. For more information about this, see the Codar online help.
IP ADDRESS	IP address of the Ubuntu system
PASSWORD	Password of the Ubuntu system
SSH PORT	Port number for the application server and database to connect to this Ubuntu system. The default port is 22.
PRIVATE KEY	Private key for secure SSH access
USER NAME	User name for whom SSH access must be granted to this Ubuntu system. For example, root

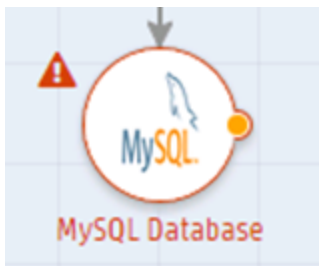
Customizing the second existing server

Perform the following steps to customize the application server component:

1. Click the existing server component.
2. On the **Properties** tab, enter the following information:

Field	Description
Name	Name of the server. This is typically the name of the second Ubuntu systems specified in Prerequisites .
HOST-FQDN	Complete FQDN name of the server
Modifiable during service creation	Select this check box if you want to update the FQDN during deployment. For more information about this, see the Codar online help.
IP ADDRESS	IP address of the second Ubuntu system
PASSWORD	Password of the second Ubuntu system
SSH PORT	Port number for the application server and database to connect to this Ubuntu system. The default port is 22.
PRIVATE KEY	Private key for secure SSH access
USER NAME	User name for whom SSH access must be granted to this Ubuntu system. For example, root

Customizing MySQL

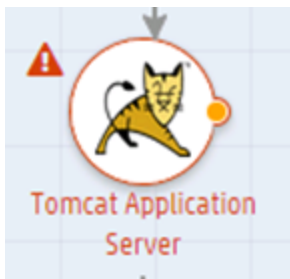


Perform the following steps to customize the MySQL component:

1. Click the MySQL component.
2. On the **Properties** tab, enter the following information:

Field	Description
Name	Name of the database.
artifactPassword	Enter the password for accessing the HTTP location. This is an optional field.
Modifiable during service creation	Select this check box if you want to update the password during deployment. For more information about this, see the Codar online help.
artifactURL	HTTP location where the MySQL database is located. That is, the location where <code>mysql-server_5.6.22-1ubuntu12.04_amd64.deb-bundle.tar</code> is located
artifactUsername	Name of the user who accesses the HTTP location
configurationUrl	HTTP location in which <code>install_mysql.sh</code> is located. For example, <code>http://localhost:80/scripts/install_mysql.sh</code>
installPath	Path in which MySQL or Tomcat is installed. This is an optional field.
privatekeyPath	Key to connect to MySQL. This is an optional field.
remoteFilePath	Remote file path in which the files must be copied to the Ubuntu server.
serviceCommand	<code>sh install_mysql.sh</code>
sshPort	Enter 22 as the SSH port number.

Customizing Tomcat

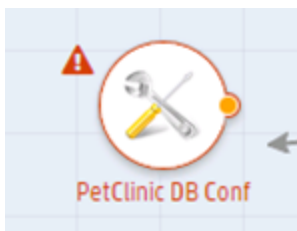


Perform the following steps to customize the Tomcat component:

1. Click the Tomcat component.
2. On the **Properties** tab, enter the following information:

Field	Description
Name	Name of the Tomcat server.
artifactPassword	Enter the password for accessing the HTTP location. This is an optional field.
Modifiable during service creation	Select this check box if you want to update the password during deployment. For more information about this, see the Codar online help.
artifactURL	HTTP location where the Tomcat server is located.
artifactUsername	Name of the user who accesses the HTTP location
configurationUrl	HTTP location in which <code>install_tomcat.sh</code> is located. For example, <code>http://localhost:80/scripts/install_tomcat.sh</code>
installPath	Path in which MySQL or Tomcat is installed. This is an optional field.
privatekeyPath	Key to connect to Tomcat. This is an optional field.
remoteFilePath	Remote file path in which the files must be copied to the Ubuntu server.
serviceCommand	<code>sh /tmp/install_tomcat.sh</code>
sshPort	Enter 22 as the SSH port number.

Customizing the PetClinic DB Conf component



Perform the following steps to customize the PetClinic DB Conf component:

1. Click the PetClinic DB Conf component.
2. On the **Properties** tab, enter the following information:

Field	Description
Name	Name of the component. That is, PetClinic DB Conf
artifactPassword	Enter the password for accessing the HTTP location. This is an optional field.
Modifiable during service creation	Select this check box if you want to update the password during deployment. For more information about this, see the Codar online help.
artifactURL	HTTP location where the PetClinic DB Conf is located.
artifactUsername	Name of the user who accesses the HTTP location
configurationUrl	HTTP location in which <code>install_tomcat.sh</code> is located. For example, <code>http://localhost:80/scripts/mysqladb_conf.sh</code>
installPath	Path in which MySQL or Tomcat is installed. This is an optional field.
privatekeyPath	Key to connect to Tomcat. This is an optional field.
remoteFilePath	Remote file path in which the files must be copied to the Ubuntu server.
serviceCommand	<code>sh /tmp/mysqladb_conf.sh</code>
sshPort	Enter 22 as the SSH port number.

Customizing the PetClinic Application component



1. Click the PetClinic Application component.
2. On the **Properties** tab, enter the following information:

Field	Description
Name	Name of the component. That is, PetClinic Application
artifactPassword	Enter the password for accessing the HTTP location. This is an optional field.

Field	Description
Modifiable during service creation	Select this check box if you want to update the password during deployment. For more information about this, see the Codar online help.
artifactURL	HTTP location in which <code>petclinic.war</code> is located
artifactUsername	Name of the user who accesses the HTTP location
configurationUrl	HTTP location in which <code>petclinic_jdbc_conf.sh</code> is located <code>http://localhost:80/scripts/petclinic_jdbc_conf.sh</code>
localfilepath	Path in which <code>petclinic.war</code> is located. This is an optional field.
port	Enter 22 as the port number.
privatekeyPath	Key to connect to <code>petclinic.war</code> . This is an optional field.
remoteFilePath	<code>/tmp</code>
serviceCommand	<code>sh /tmp/petclinic_jdbc_conf.sh</code>

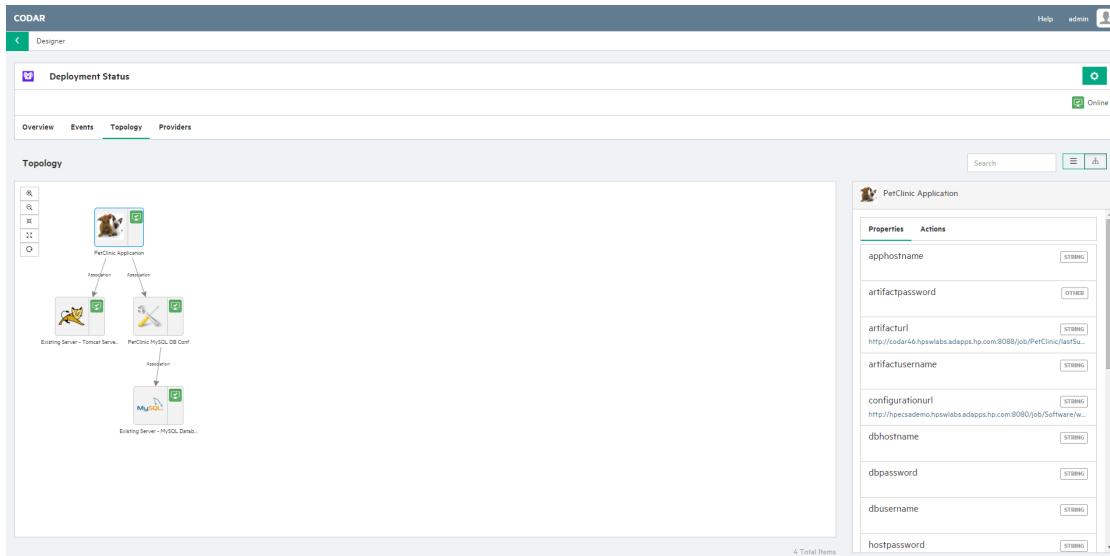
Deploying the application design

After customizing the components in the application design, you can deploy it and start working with the PetClinic application. The application design is saved automatically after every change. Perform the following steps to deploy the application design:

1. Click the **Test** tab and click **Test Run** to install the software and deploy the application on the Tomcat server. Ensure that Tomcat and MySQL are not already installed in Ubuntu systems 1 and 2.
2. In the Test Run dialog box, in the **Name** field, enter a name for the application. You can enter any name, for example, Test Run <application_name>.
3. Click **Finish**.
4. Click the **Test** tab to see the status of the deployment. The status is first displayed as **Deploying** and then **Online**. An **Online** status indicates a successful deployment.



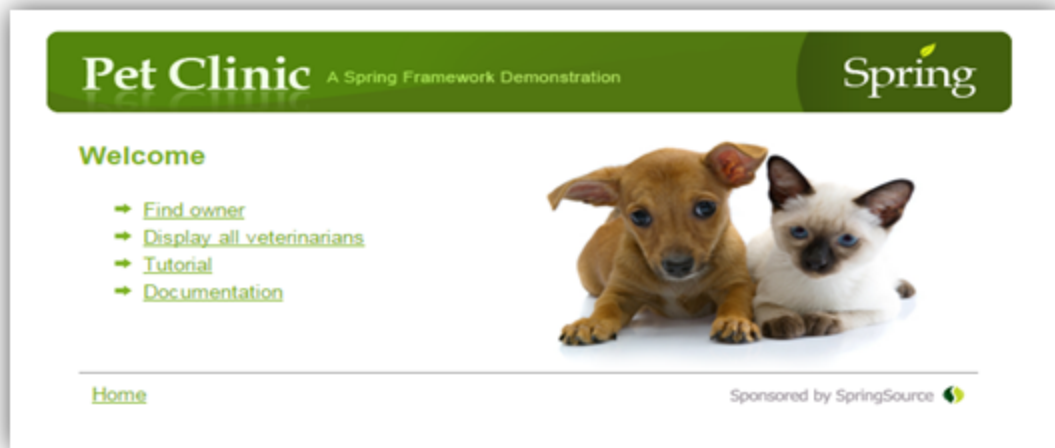
5. To see the summary of the application, click the <application_name> link and then click the **Topology** tab. The screen displays the structure of the application and also displays which components were successfully executed and which components have failed.



Get started

Deploying the application design

6. Launch the application URL to check for the successful deployment of the application. The application URL is in the form `http://<IP_address of the Tomcat server>/petclinic`. For example, `http://10.1.155.58:8080/petclinic`



Troubleshooting

This topic explains some troubleshooting steps in case you encounter errors or problems while launching the PetClinic application.

If you are not able to launch the application, perform the following checks:

- Ensure that you have used the right version of Ubuntu (version 12.04 or later).
- Ensure that the dependent libraries mentioned at [step 9](#) in the "[Downloading files](#)" topic are installed.
- Verify that port numbers 8080 and 1521 are open and available for Tomcat and MySQL respectively.

Send documentation feedback

If you have comments about this document, you can [contact the documentation team](#) by email. If an email client is configured on this system, click the link above and an email window opens with the following information in the subject line:

Feedback on Get started (Codar 1.90)

Just add your feedback to the email and click send.

If no email client is available, copy the information above to a new message in a web mail client, and send your feedback to clouddocs@hpe.com.

We appreciate your feedback!