Hewlett Packard
Enterprise

HPE Codar

Software Version: 1.60

Concepts Guide

Document Release Date: January 2016
Software Release Date: January 2016

Legal Notices

Warranty

The only warranties for Hewlett Packard Enterprise Development LP 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. HPE 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 HPE required for possession, use or copying. Consistent with FAR 12.211 and 12.212, Commercial Computer Software, Computer Software Documentation, and Technical Data for Commercial Items are licensed to the U.S. Government under vendor's standard commercial license.

Copyright Notice

Trademark Notices

Adobe™ is a trademark of Adobe Systems Incorporated.

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

UNIX® is a registered trademark of The Open Group.

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

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: **https://softwaresupport.hp.com**

This site requires that you register for an HPE Passport and sign in. To register for an HPE Passport ID, go to: **https://hpp12.passport.hp.com/hppcf/createuser.do**

Or click the **the Register** link at the top of the HPE Software Support 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 Online web site at: **https://softwaresupport.hp.com**

This web site provides contact information and details about the products, services, and support offered by HPE Software.

HPE Software online support provides customer self-solve capabilities. It provides a fast and efficient way to access interactive technical support tools needed to manage your business. As a valued support customer, you can benefit by using the support web site to:

- Search for knowledge documents of interest
- Submit and track support cases and enhancement requests
- Download software patches
- Manage support contracts
- Look up support contacts
- Review information about available services
- Enter into discussions with other software customers
- Research and register for software training

Most of the support areas require that you register as a Passport user and sign in. Many also require a support contract. To register for a Passport ID, go to:

https://hpp12.passport.hp.com/hppcf/createuser.do

To find more information about access levels, go to:

https://softwaresupport.hp.com/web/softwaresupport/access-levels

HPE Software Solutions Now accesses the HPSW Solution and Integration Portal web site. 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 web site is **http://h20230.www2.hp.com/sc/solutions/index.jsp**

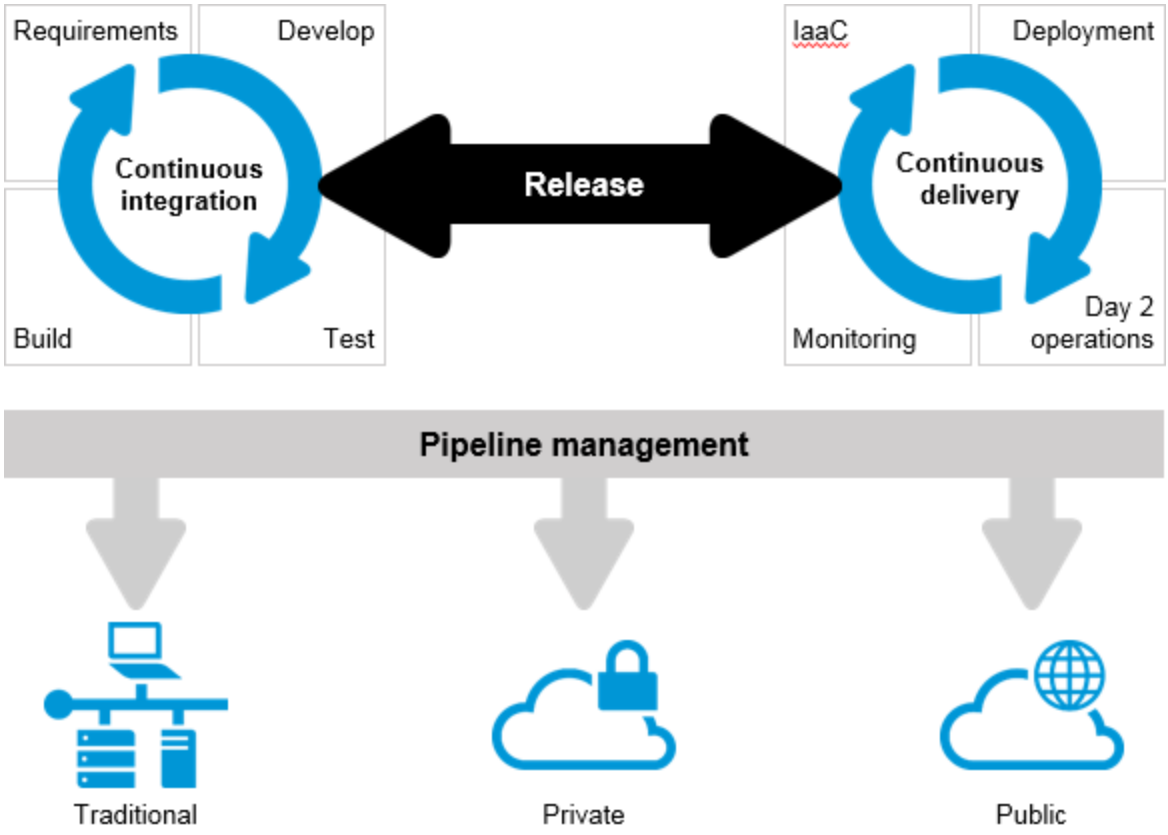
Contents

Codar

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.

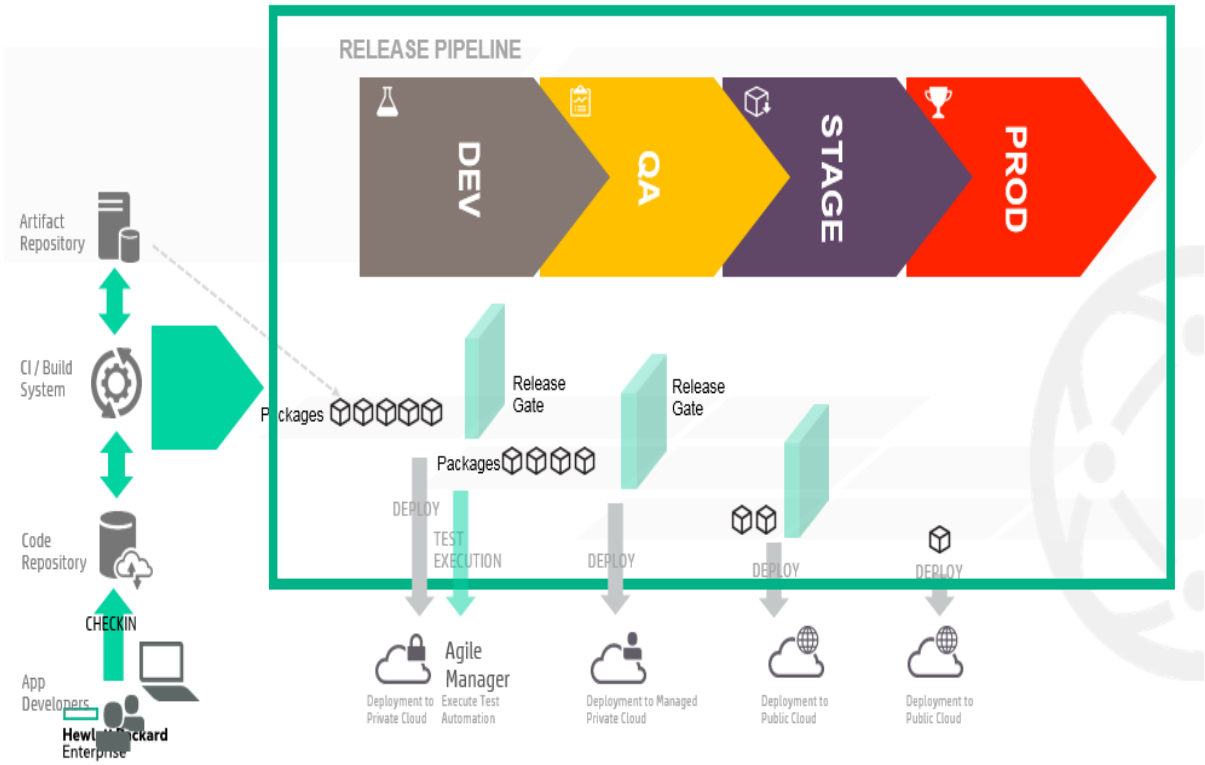
DevOps provides a framework to bridge the gaps between the development (Dev) and operations (Ops) environments by using a set of principles, methods, and practices around collaboration, automation, and governance. The goal is to extend continuous build or assembly integration to repeatable and consistent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

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



Codar overview

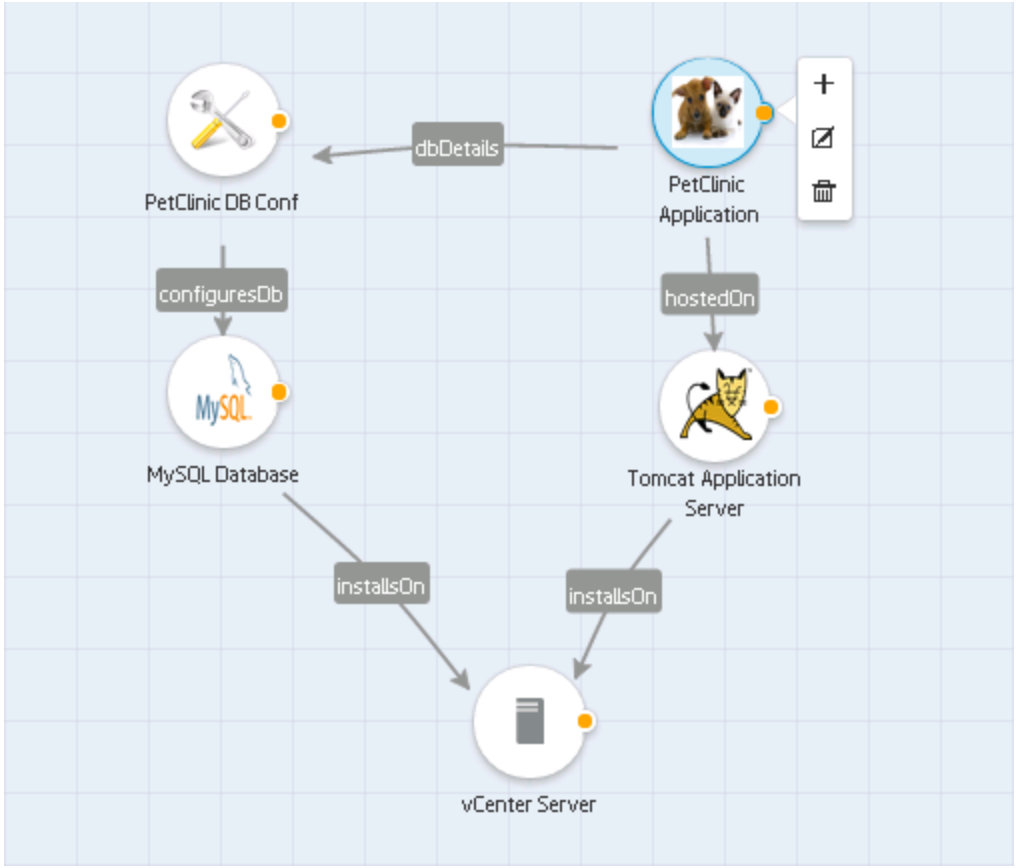
Hewlett Packard Enterprise Codar facilitates continuous delivery in which every change to the system is releasable and that every code change can be deployed in production. It enables automation of continuous delivery where every code change triggers a build, which is deployed, automated unit tests are 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 fast feedback from users.



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.

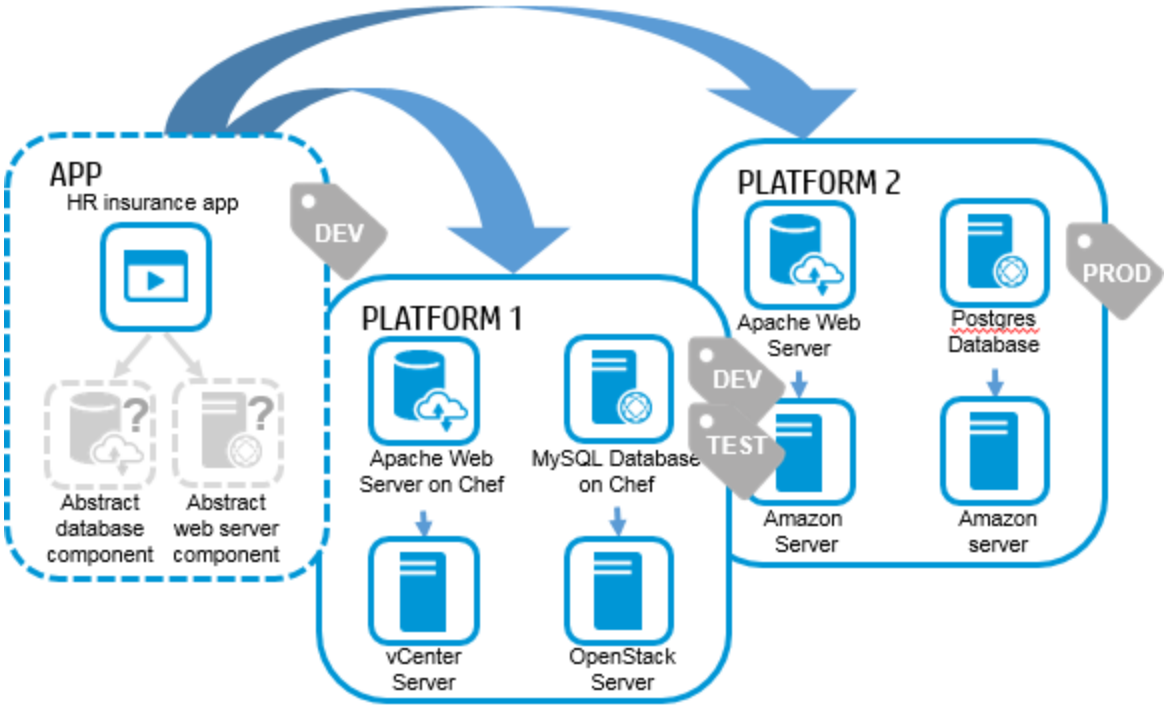
An application design can be of two types:

- 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.



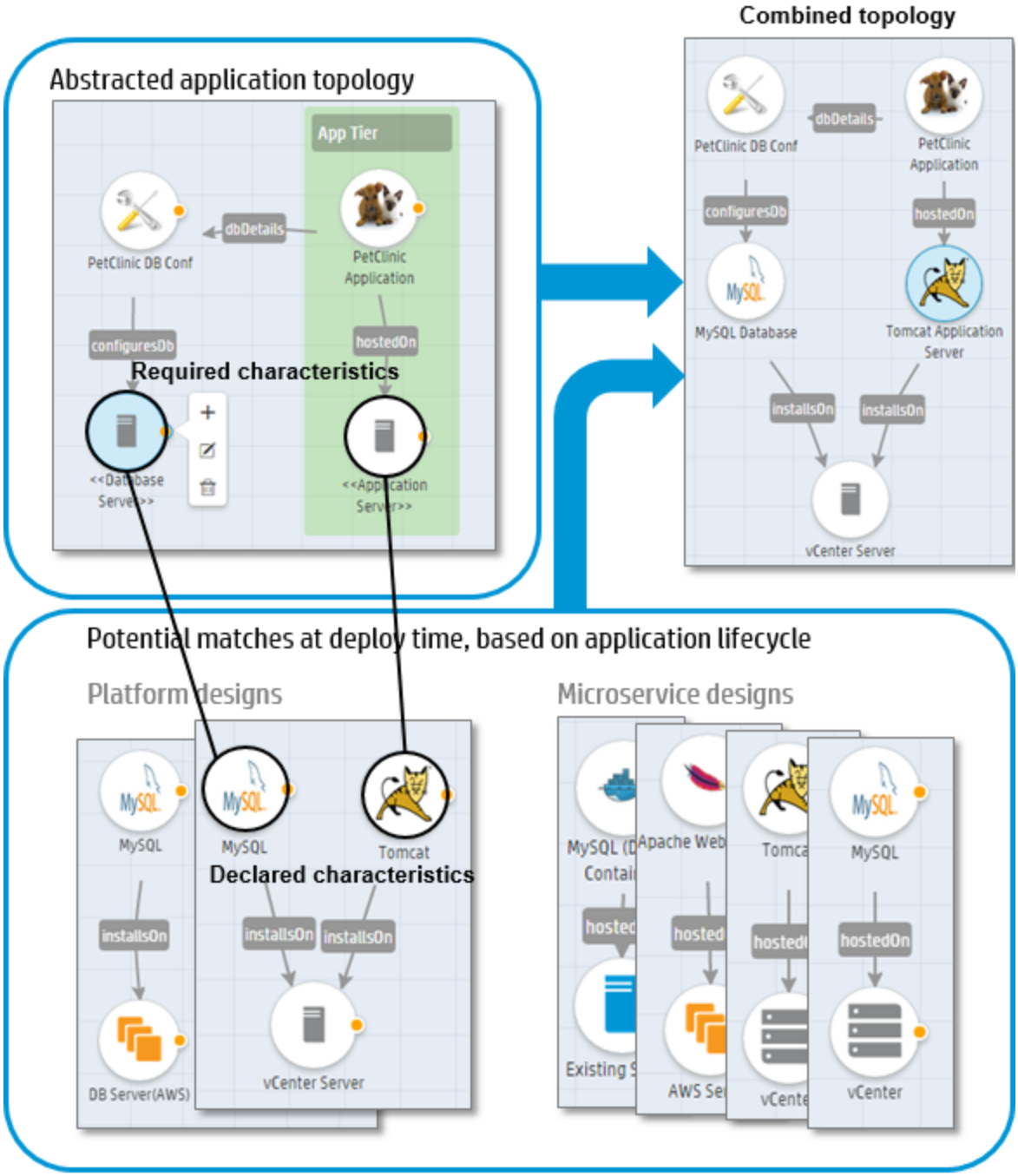
Microservices

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

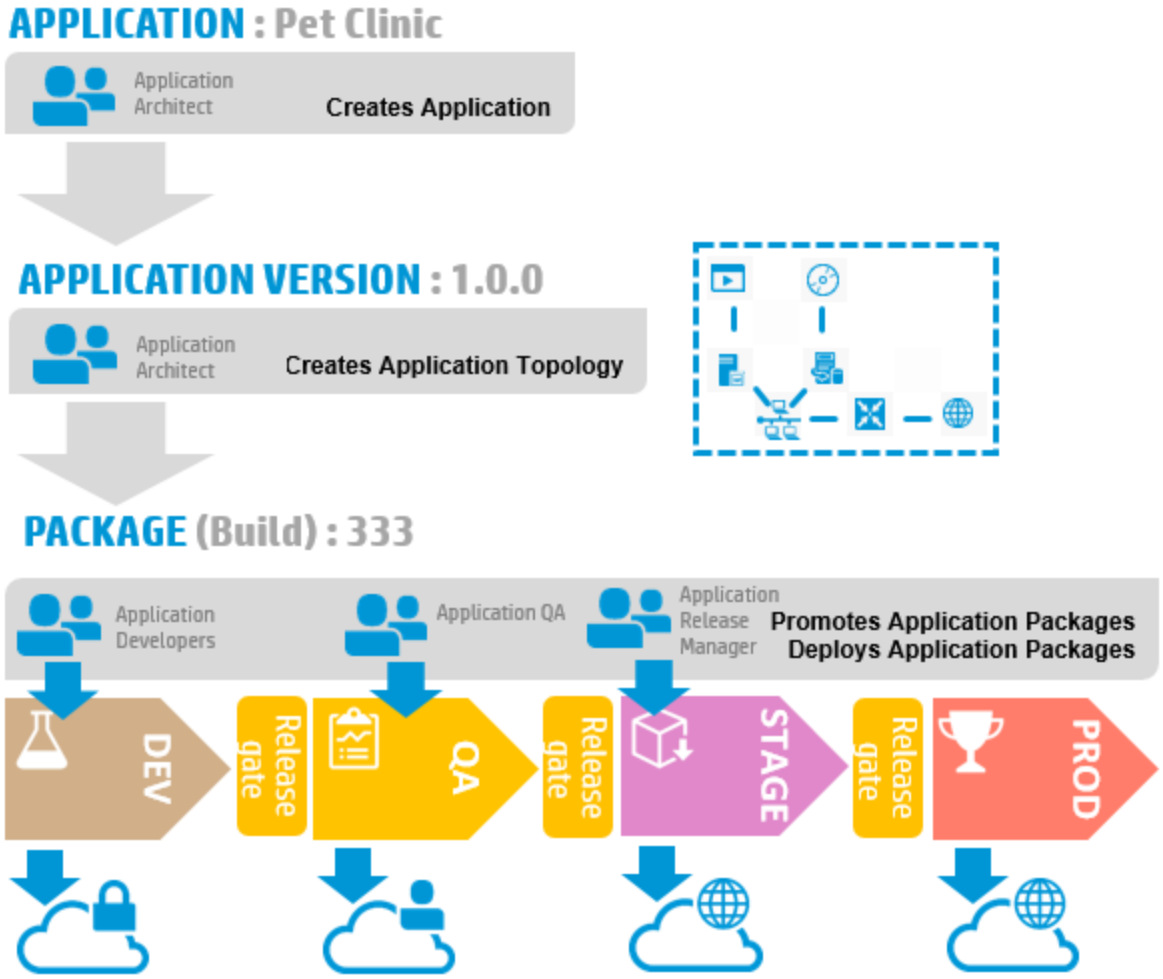
Automating the deployment of applications is a complicated and lengthy process and requires significant investment. Applications are deployed differently in development and in production, causing many errors. Application pipeline management allows you to deploy applications across different environments 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



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.

Tasks

- **Create a package from a specific application version.** An application version can consist of multiple packages.
- **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.
- **Delete a package.** Go to the Release Pipeline tab, hold Ctrl to select multiple packages, and click **Delete**.

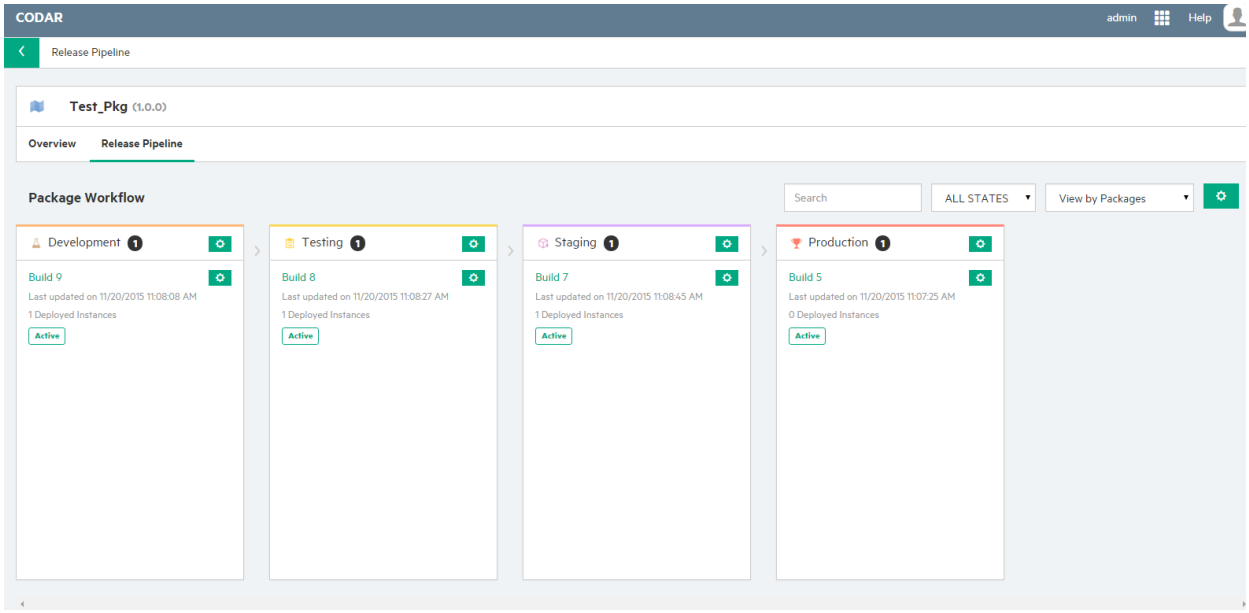
  You cannot delete a package that has an instance associated with it.

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.

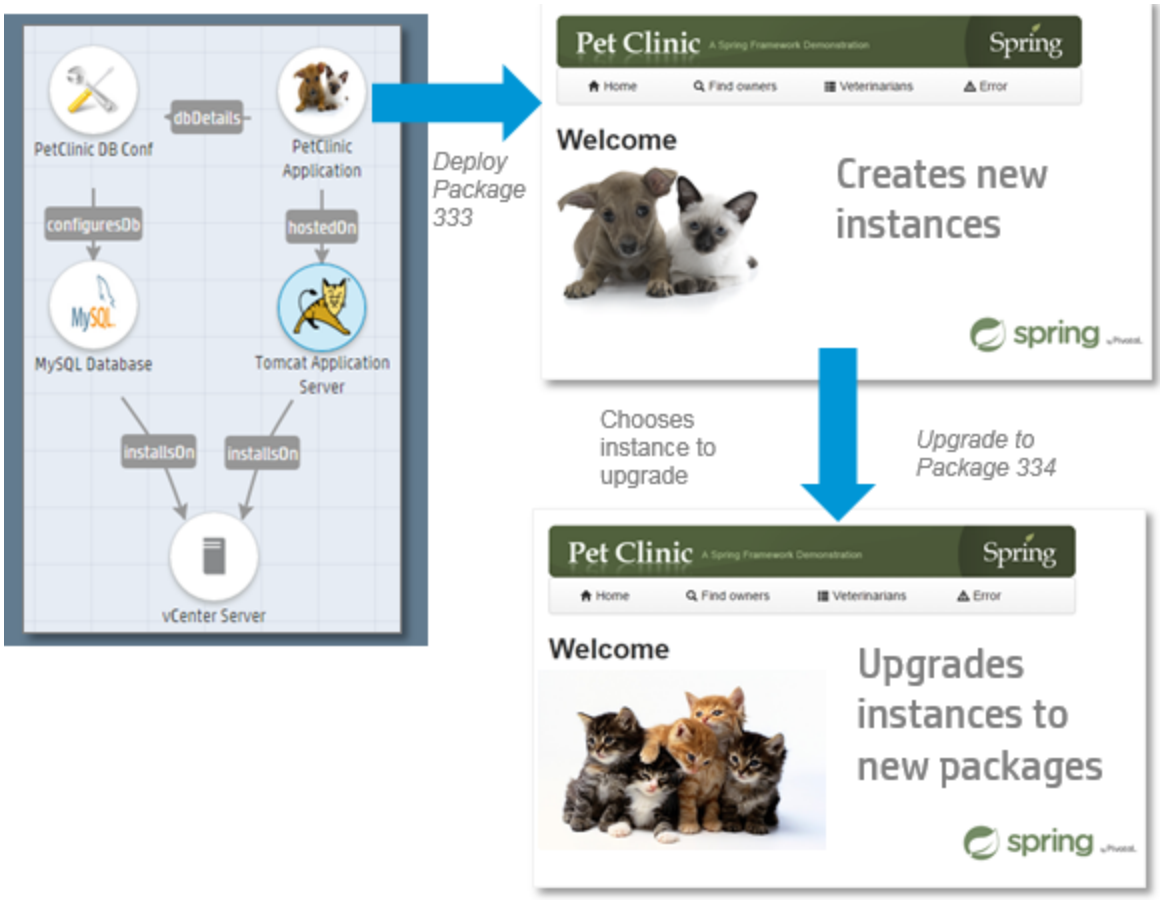
When packages are created and deployed, new virtual machines will be created and packages will be deployed. You can execute tests on a deployed instance, and the package can be either promoted or rejected

Codar facilitates application pipeline management capabilities, as show in the following screenshot.



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 insistence. 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.

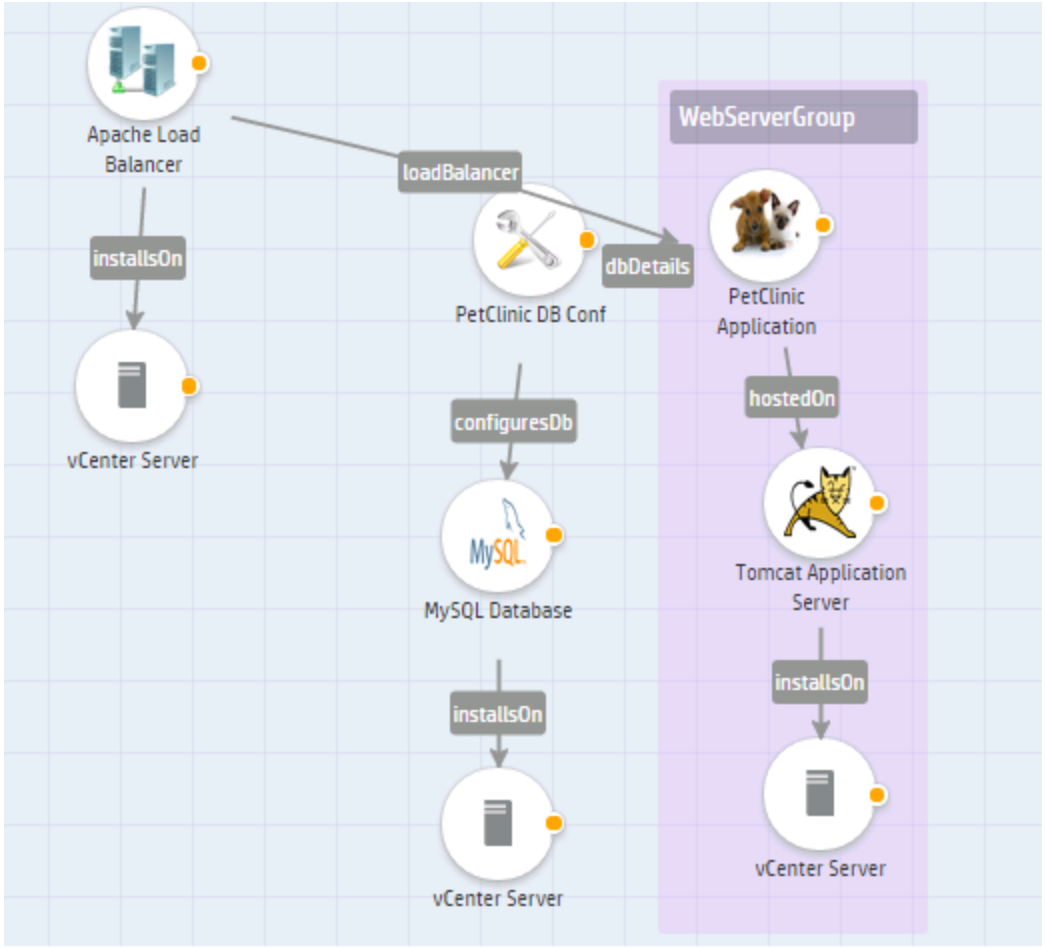


Scale out

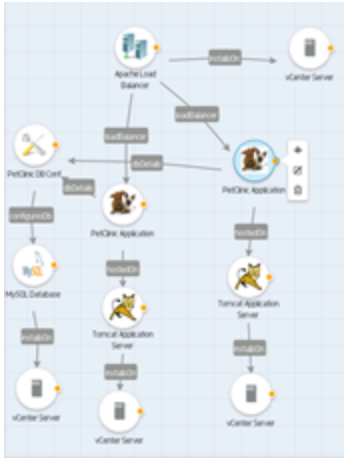
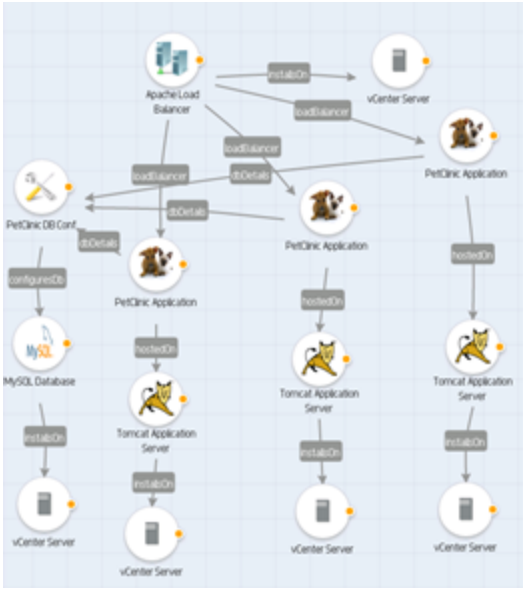
During Topology design creation you can create a scaling group. A scaling group represents a scalable stack. There can be multiple scalable groups in an application design.

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.



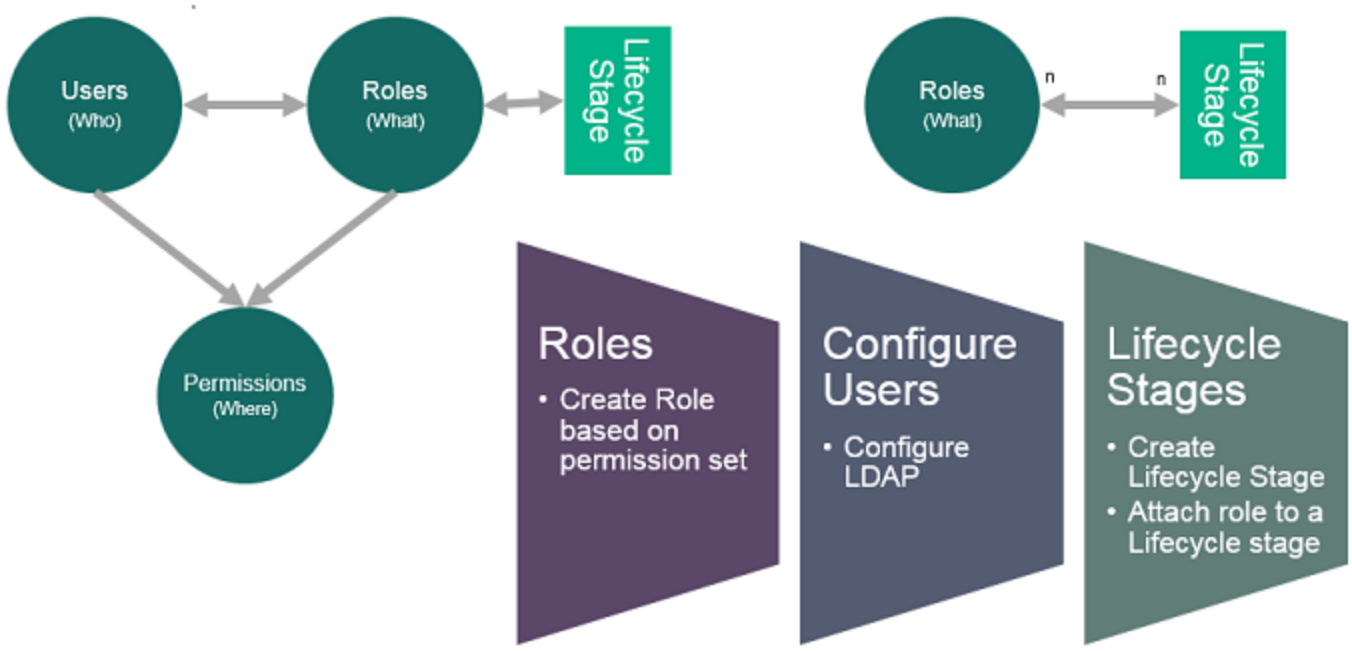
This group was scaled in the Development stage to one group and in Testing it is scaled to two groups in the image below.

Development stage	Testing stage
	

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.

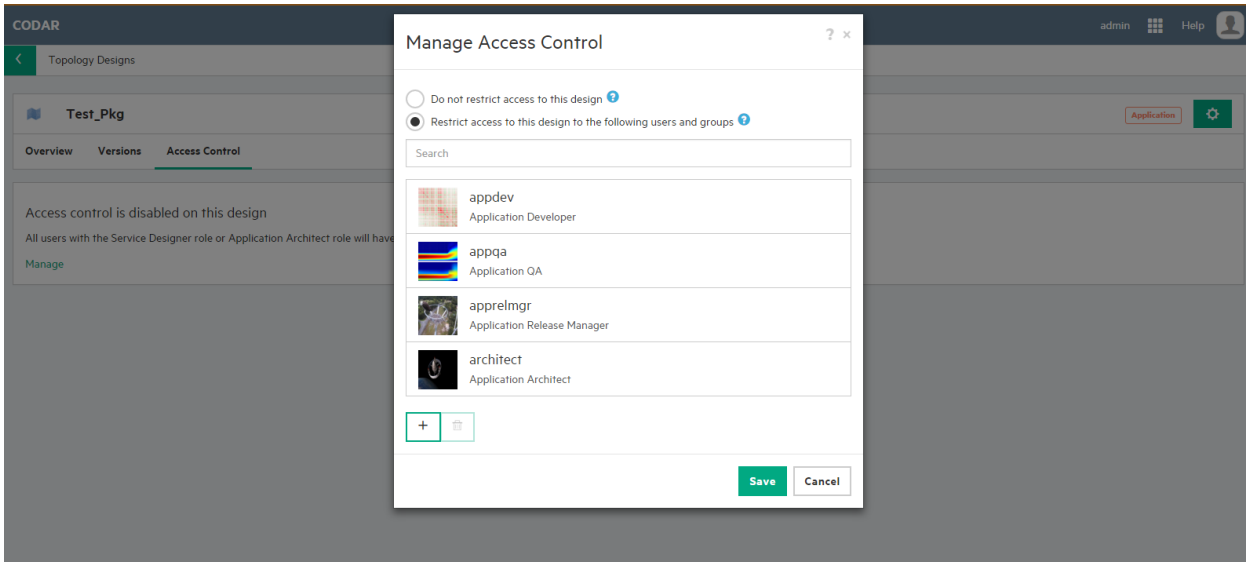
In Codar, user access comprises roles and permissions. Every user in Codar is assigned one or multiple roles. Every role is assigned one or more permissions. Therefore, users belonging to a particular role have all the permissions defined for that role. Codar contains some out-of-the-box roles; however, 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.


CODAR
admin
Help
Topology Designs
Test_Pkg
Application
Overview
Versions
Access Control
Access control is disabled on this design
All users with the Service Designer role or Application Architect role will have
Manage
Manage Access Control
Do not restrict access to this design
Restrict access to this design to the following users and groups
Search
appdev
Application Developer
appqa
Application QA
apprelmgr
Application Release Manager
architect
Application Architect
Save
Cancel

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** tile.

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.
- **Edit**: Change the properties of a package.
- **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**: Delete a package. The package will be removed permanently from the system. A package can only be deleted if all associated deployed instances are canceled and deleted.
- **Refresh**: Retrieves current package status.

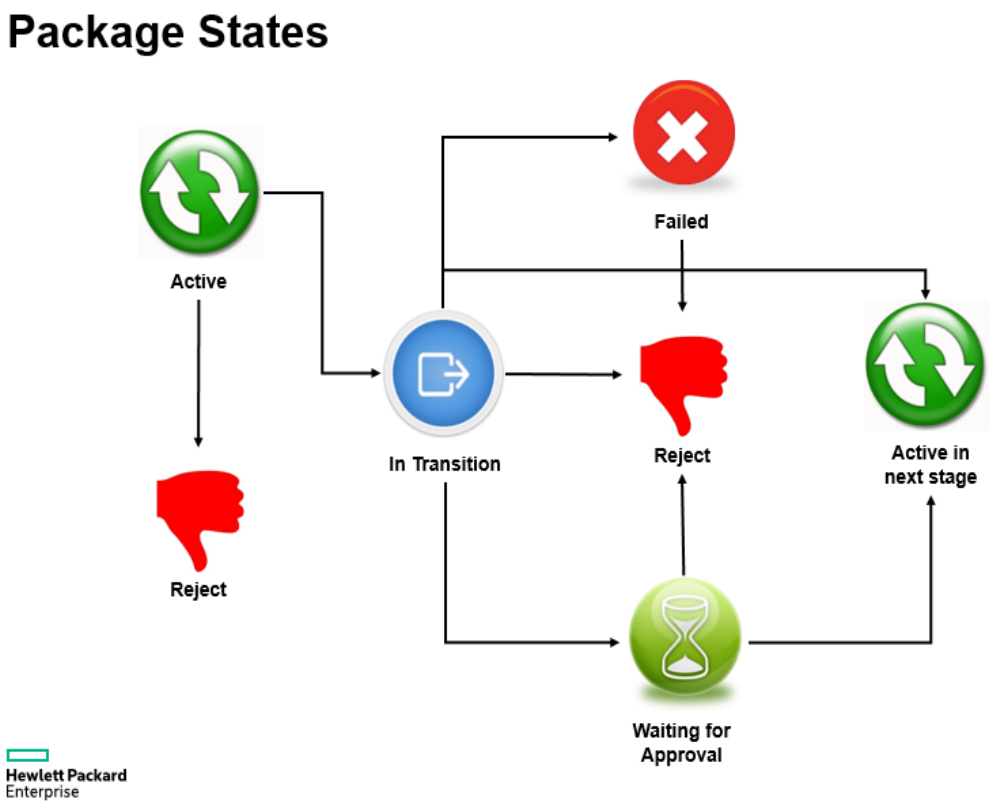
Package states

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.



Grouping infrastructure designs by lifecycle stage

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

To group the infrastructure 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 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 three types:

- Deploy action

  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.

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

- Custom action

  This action requires a package deployment that is created using a deploy action. It executes some OO flows on the previously deployed instance. Deploy actions serve as input to custom actions; therefore, a deploy action must necessarily exist or be created before creating a custom action.

  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.

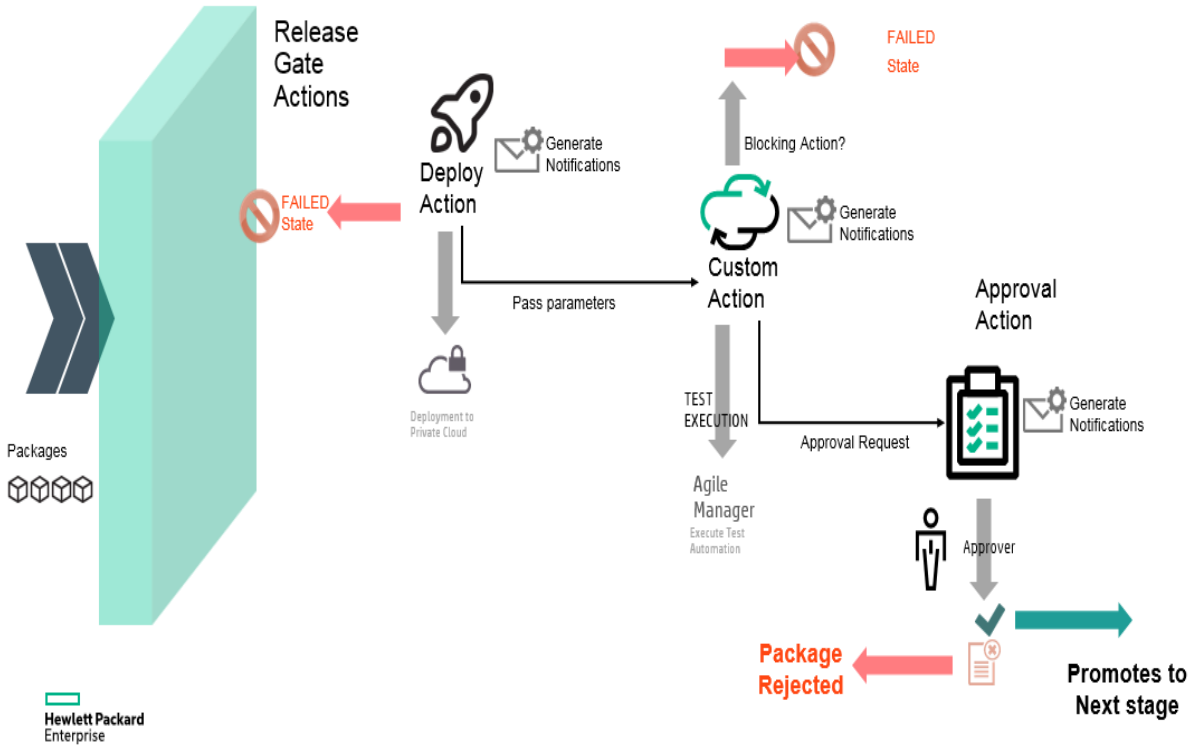
  For detailed information about creating, editing, and deleting custom actions, see the *Codar Online Help*.

- Approval action

  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.

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

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



Pipeline statistics

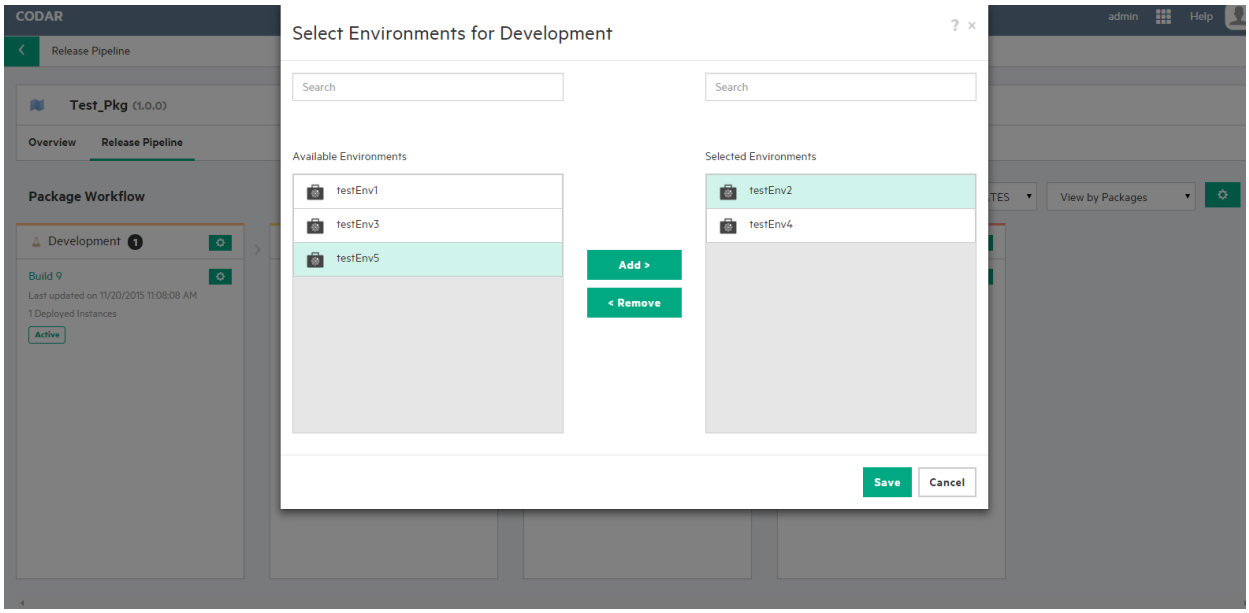
The Pipeline Statistics tab displays detailed information about packages and includes graphical representations of package summary, trends, states, deployment status and so on. It provides a holistic view of all packages and deployments and enables you to make informed decisions with respect to package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

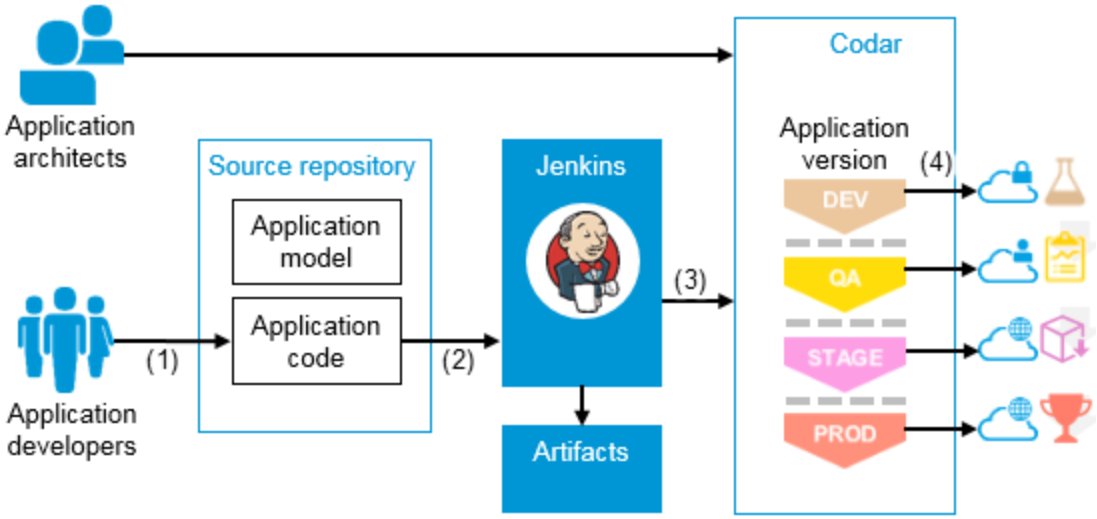


External integrations

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

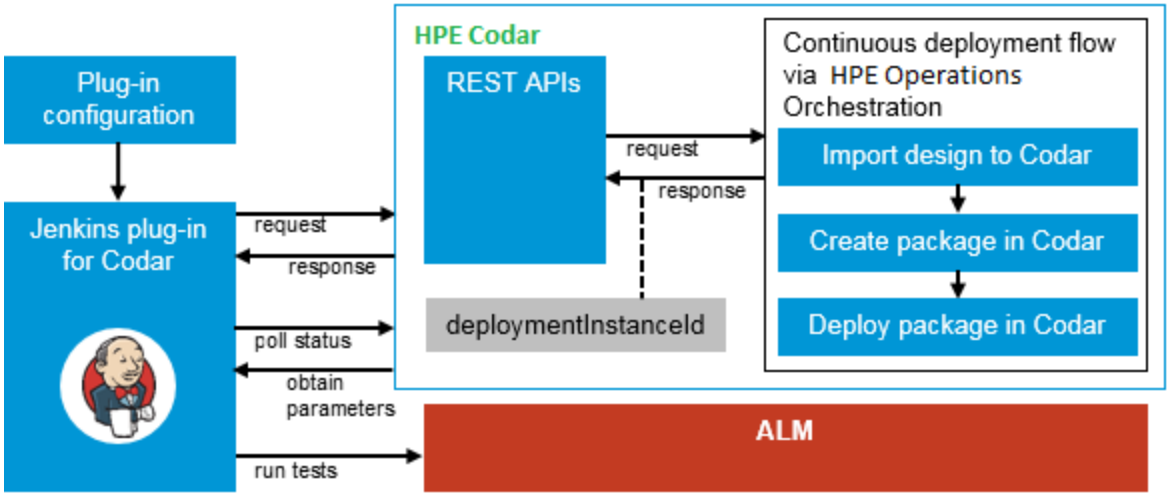
Codar includes a Jenkins plug-in for continuous deployment. The following illustration shows how it works.



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.

ALM integration

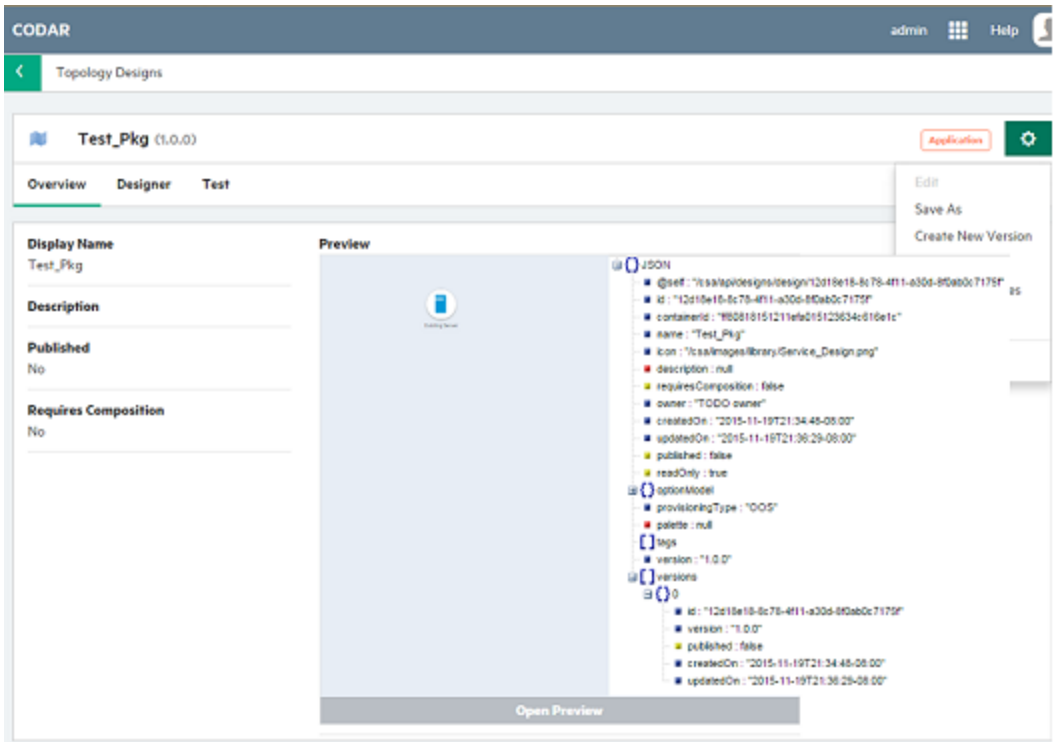
You can integrate Application Lifecycle Management (ALM) with Codar to run tests after successful deployment. The following illustration shows how Jenkins acts as an orchestrator.



Infrastructure as code (IaaC)

Managing infrastructure as code (IaaC) 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.

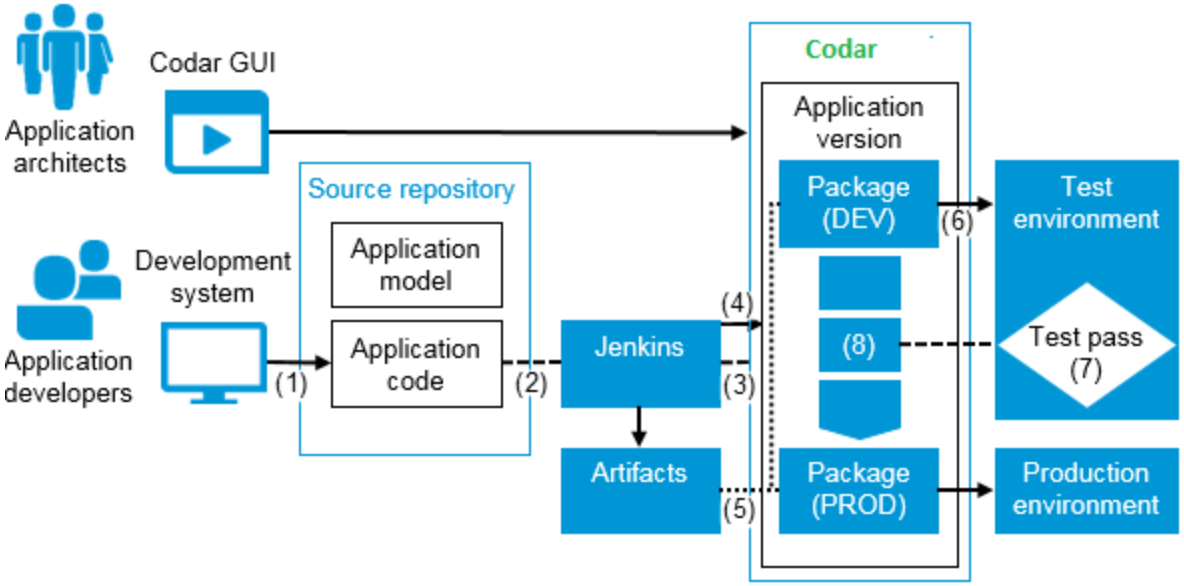
Codar can manage infrastructure as code. Topology designs that can contain server configurations, networks, volumes, relationships, and application-specific details like the application version and package information can be exported in JSON format and managed with the application in the source control 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.



Use case: 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 (IaaC). 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 (IaaC).

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.

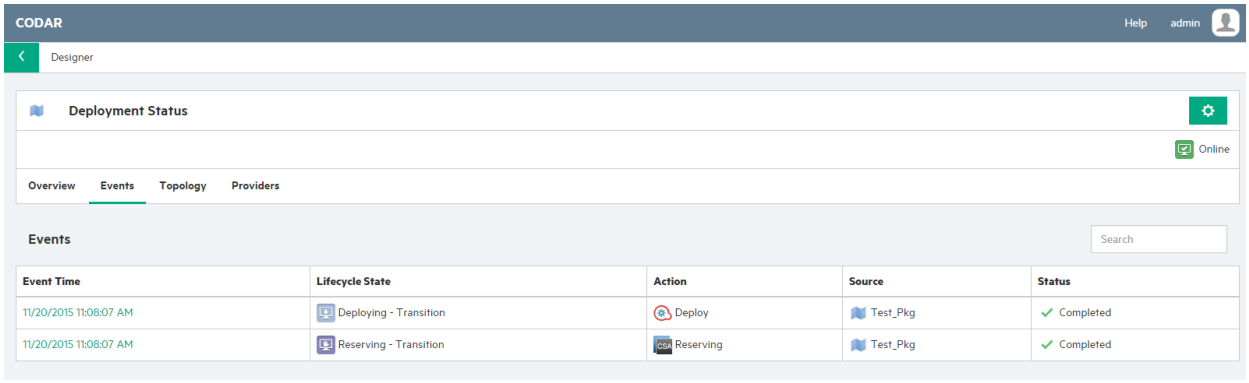
Importing an application design

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 (IaaC), 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 tile 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:



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 an Hewlett Packard Enterprise CSA license installed before you can publish a design.

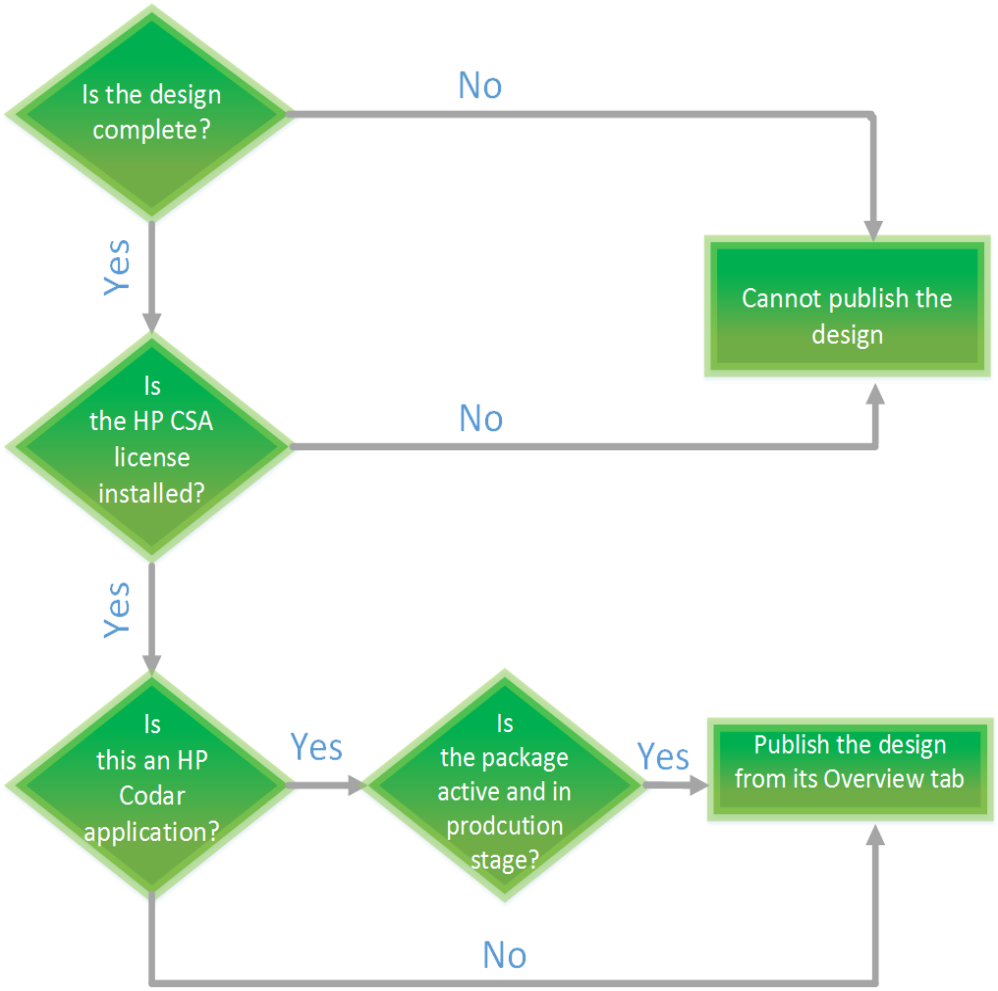
A complete design with an active package in the Production stage contains package-specific properties as part of the design and can be published.

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:

- An 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 an Codar application design must be saved as a composed design from the Test tab. The design can then be published to service consumers.

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



Use case: Customizable release pipeline

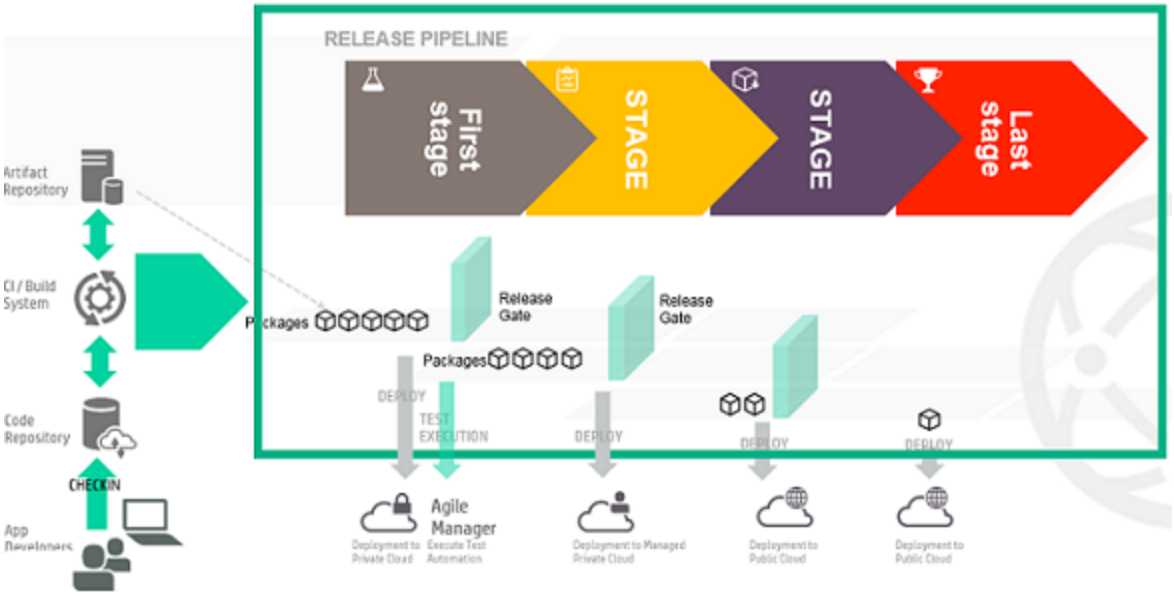
The goal is to allow users to build their own custom release pipeline and have each application team define and use separate lifecycle stages. The following are the high-level steps for users to define their own release pipeline. For information about how to perform these steps, see the *Codar Online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



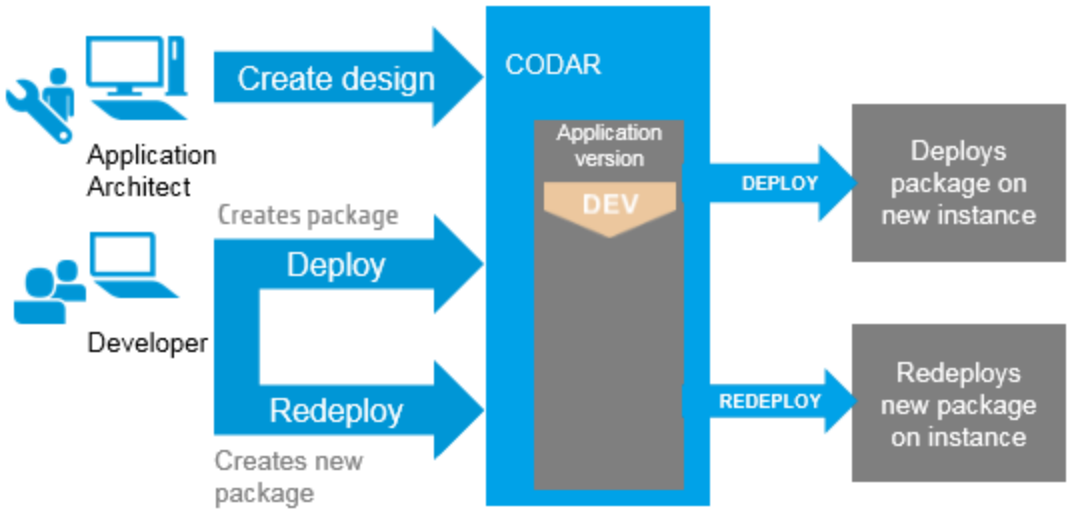
Use case: Deploy and redeploy packages

The goal is to deploy a package and then use the same instance to redeploy a newer version o 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:



Use case: 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.

Next steps

Codar Installation Guide and*CodarConfiguration Guide* explain how to download, install, and configure the software, and *Codar API and CLI Reference* provides a brief introduction to the REST APIs and explains how to obtain detailed information for each API. It also explains the command-line interface. You can access online help from the application for task-oriented assistance.