HP Codar

Software Version 1.50

Secure Application Deployment Using Topology Composition and RBAC

Contents

Document release date: June 2015

Software release date: June 2015

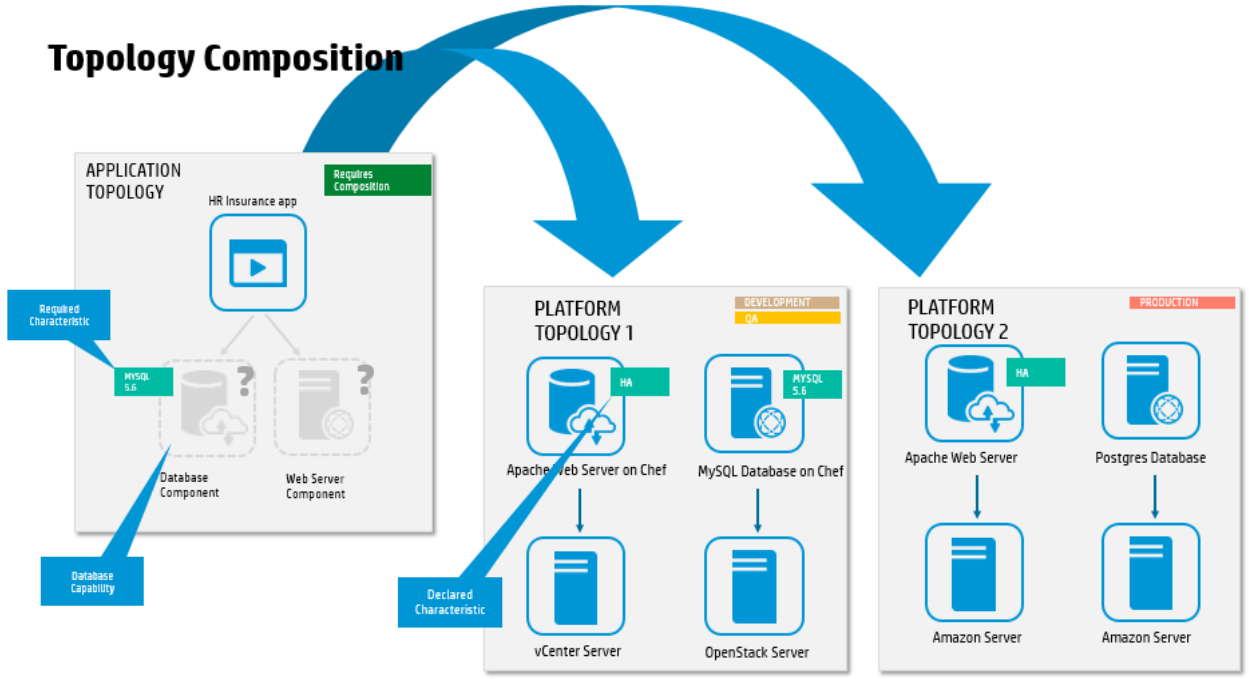
Introduction

The HP Codar resource provider provides integrations with multiple infrastructure resource providers such as VMWare, Amazon Web Services, and OpenStack etc. These providers are used to host middleware software such as application servers, database servers, web servers etc. Each of these infrastructure providers can be used in different lifecycle environments based on the requirement, feasibility, and advantage.

HP Codar provides modelling capability to deploy these software, application on the dedicated infrastructure across different environments. This modeling capability comes from different resources such as HP Operations Orchestration workflow, Opscode Chef Cookbooks, HP Server Automation software policies, Puppet etc. An application design includes many components that are catered to provision the infrastructure and install the software and application. It is most important to create infrastructure based on the lifecycle environment to avoid misuse of virtual machines by different persona who are part of different teams such as development, testing, and release management. There are various **personas** introduced in HP Codar like architect, developer, tester, and application release manager that can operate on these environment or lifecycle stages.

Topology composition, a feature introduced in HP Codar, provides a sophisticated and secure way to deploy applications on elevated and non-elevated environments thus allowing end users to select the infrastructure meant to be used only by them. This proactively secures the elevated environment. This technical documentation will help end users understand this feature based on the use cases described and implement the feature based on the samples provided. For a detailed description of this feature, see the HP Codar online Help.

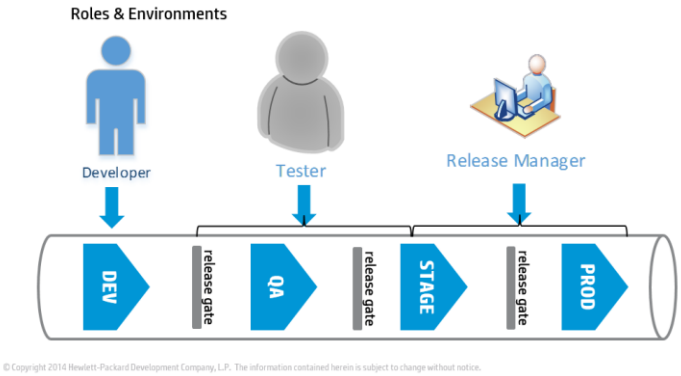
Figure 1: **Topology composition**



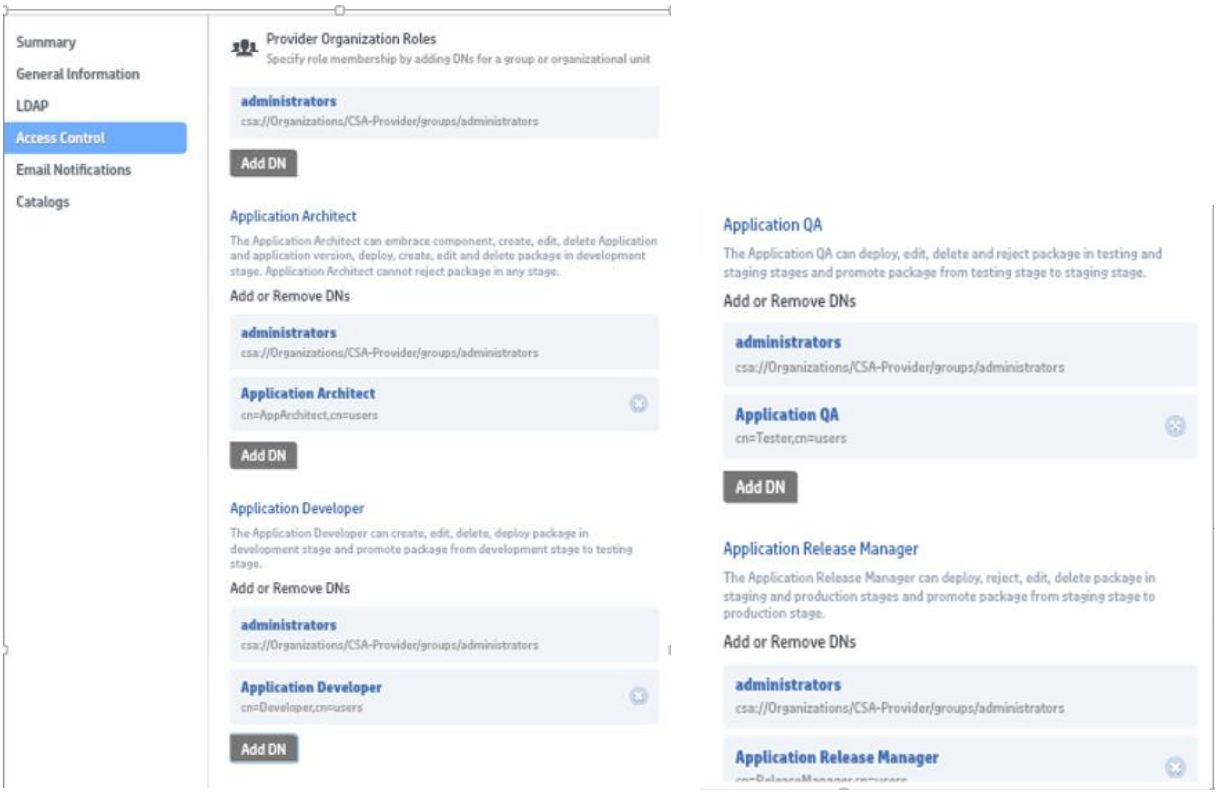
Roles and environments

There are four roles available out-of-the-box in HP Codar: developer, tester, application release manager, and application architect. Three of these roles (developer, tester, and application release manager) can be associated with different lifecycle environments such as development, testing, staging, and production. Each of these personas can deploy the application to verify and validate whether the application functions as expected and the same build version can be promoted to the next stage once it is verified by the users.

Figure 2: Roles and environment association



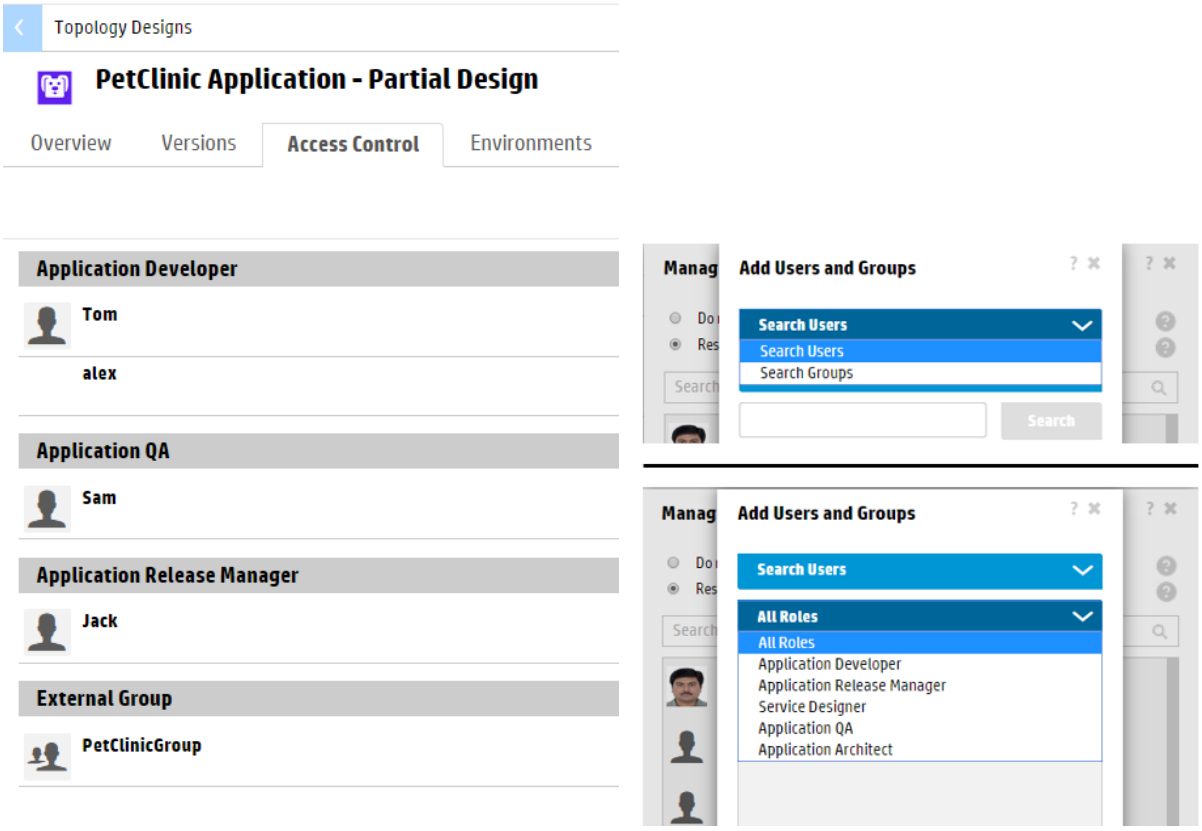
An application architect's role is to create application and infrastructure designs that allow other personas to deploy the application on the infrastructure associated with the respective lifecycle environment. The LDAP/AD user groups that map to various roles or personas are configured in the Provider Organization from which the product uses these groups to validate authentication and authorization.



Access control for application

The **Access control** feature in HP Codar protects an application by preventing users who are not allowed to access an application from accessing or using it. That is, it allows only required users to access the application.

A user can be associated with an application design or an infrastructure design (micro service). Once a user is associated with a design, the user can use those design to deploy an application on a dedicated environment. Users can be searched by user name, group name, or roles.



A group of users who are a part of LDAP or AD can also be associated with the application design or infrastructure design by searching through group names and these groups need not be configured with the Provider in Organization tile.

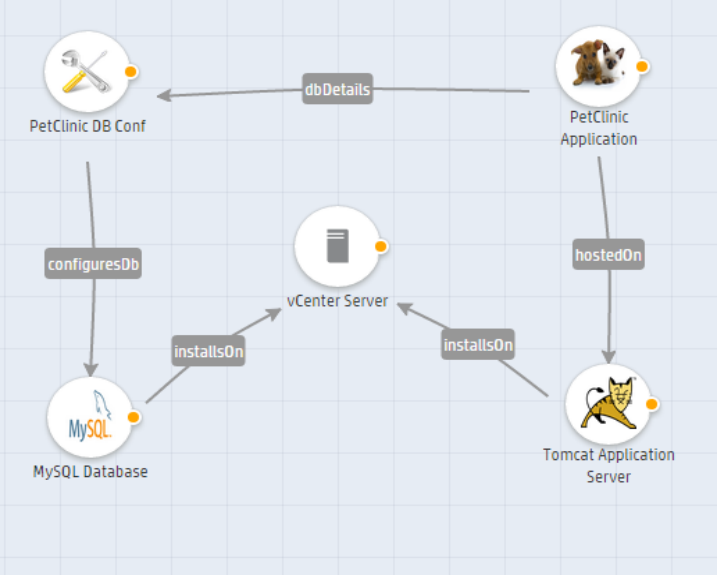
Micro services

A micro service is an individual infrastructure service that caters to an individual application tier. The application tier can either be an application server tier or a database server tier that can be tied to one or more lifecycles. The micro service can be created with a variety of combinations varying from infrastructure to platform software

Illustrative example

Consider a three-tier application that must be tested across different lifecycle environments by different personas and must finally be deployed in production. The following example uses a sample application called PetClinic that must be deployed and tested by various roles across development, test, staging, and production environments. This is a two-tier application that contains a database tier and an application tier. The infrastructure can either be provisioned from VMware or can be existing infrastructure or can be from public cloud vendors such as Amazon Web Services or OpenStack based on the requirement. The middleware layer installs Tomcat and MySQL server. The last layer is the application layer that contains components to configure the MySQL database required for this application and to deploy the petclinic.war application on the Tomcat server.

Figure 3: Sample PetClinic application design



Partial design

The application design show in Figure 3 is a complete design that contains middleware components and the infrastructure component. This design can be used to provision new virtual machines, install middleware software, and deploy the application. The application design comprises two core components: PetClinic DB Conf and PetClinic Application. These two components help the user deploy the PetClinic application on the required infrastructure. These two application components can be designed in a separate application design and set relationship with the required capabilities, which in HP Codar terminology is called ***partial design*** (see figure 6).

Capability components

What are capabilities?

A capability component is used in a partial design and contains only properties or relationships. Capability components can be associated with a concrete component which can be application components. A characteristic of a component is its capability to install an application server, database server, or a web server. A capability can be an application server, database server, web server, Docker container etc.

A partial design can have these capabilities components set as a relationship with the application components to deploy the application on the required infrastructure by selecting the filtered micro service during deployment. This is explained in detail using the following use cases.

Figure 4: Out-of-the-box capabilities components

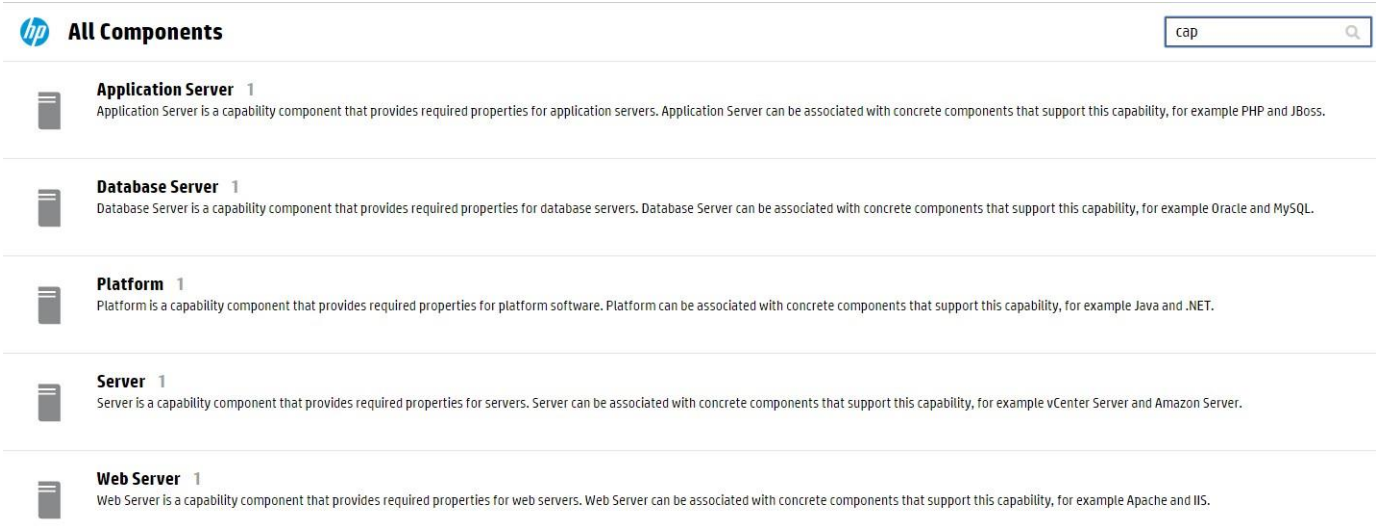
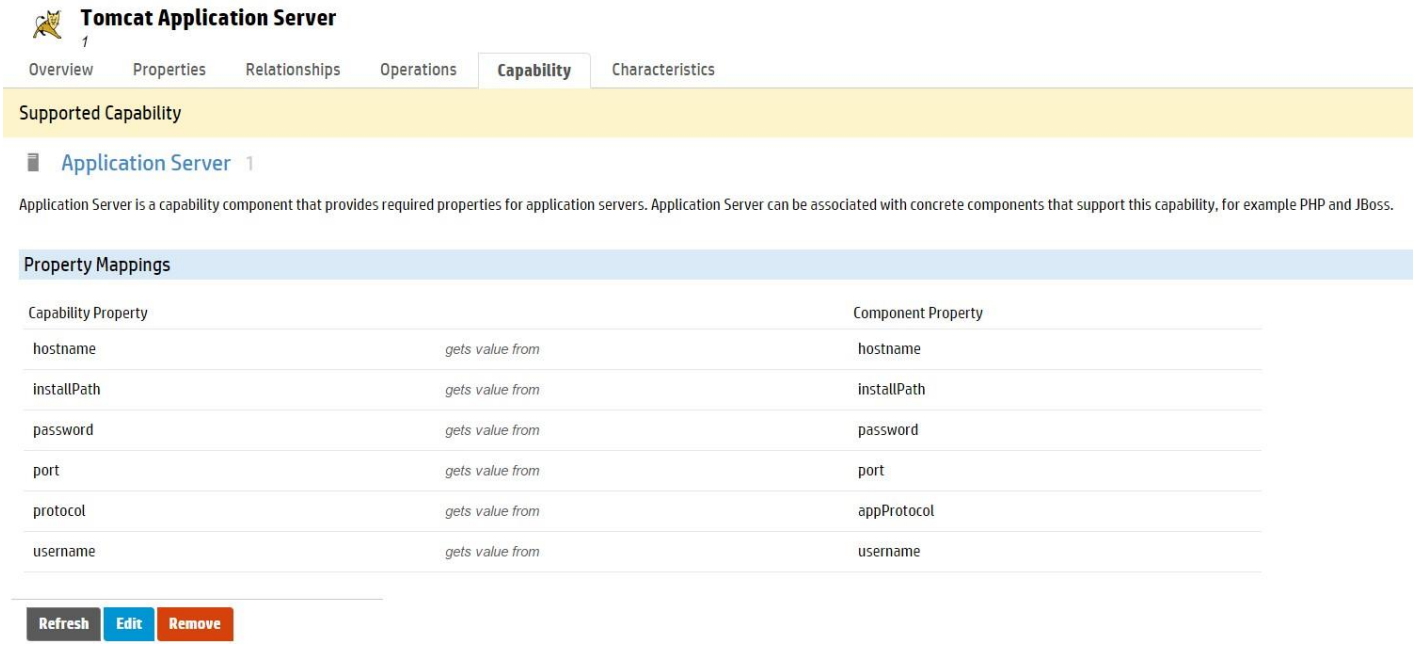


Figure 5: An application server capability is set for the component that can install the Tomcat server



Use case 1

Assume a sample PetClinic application that can be deployed with a combination of multiple application servers such as Jetty, Tomcat, JBoss and databases such as MySQL and PostgreSQL. The QA team may need to certify this application across these servers. To do this, all these components need not be a part of the same application design. The application architect can create a partial design as shown in figure 6 and create micro service (infrastructure) designs as show in figure 7.

Figure 6: A partial PetClinic application design

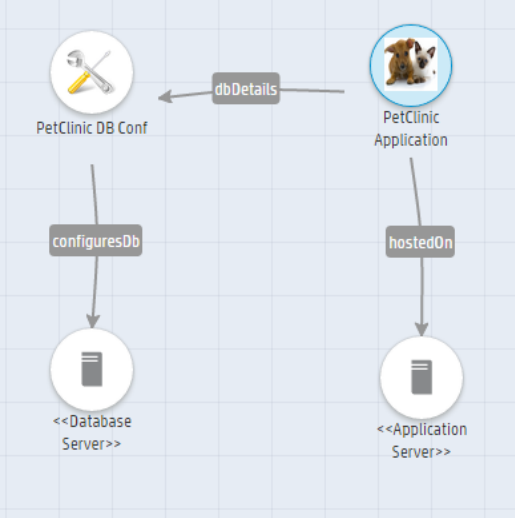


Figure 7 a: Micro service infrastructure designs to certify on multiple servers such as Tomcat, JBoss, PostgreSQL and MySQL

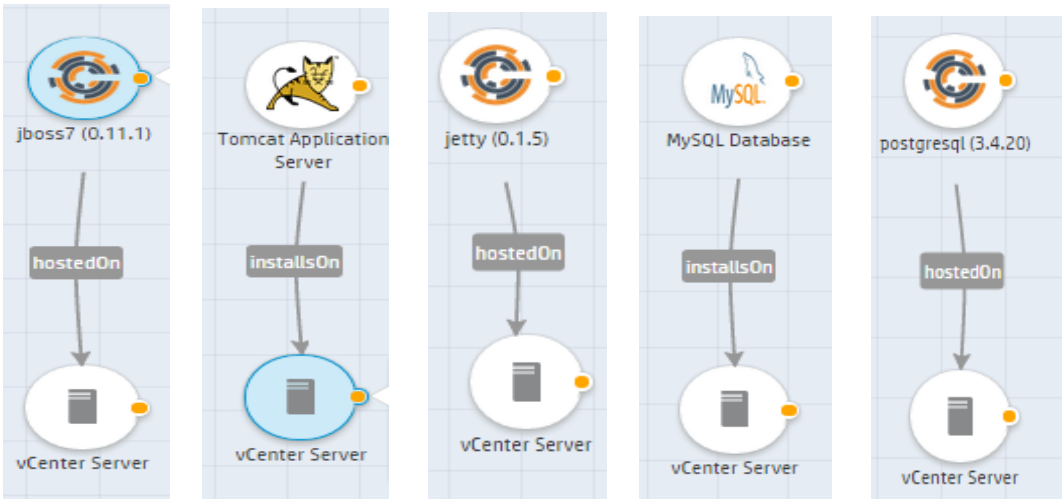


Figure 7 b: Infrastructure designs with both the tiers

Figure 8 a: Application server infrastructure designs filtered to list the matched capabilities based on partial design

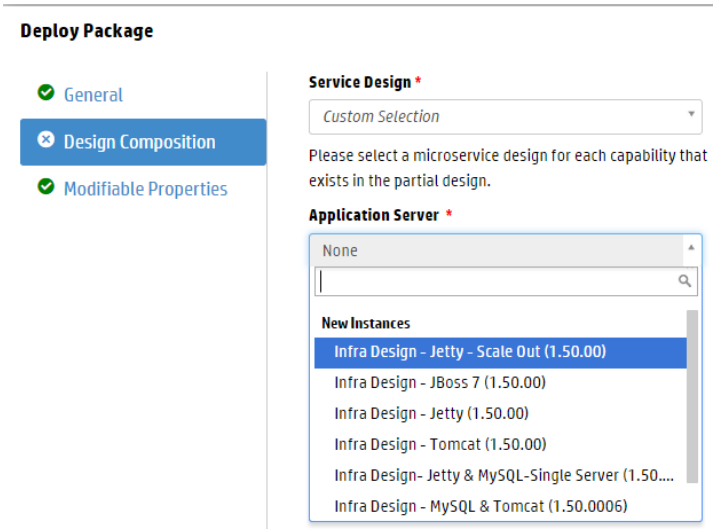
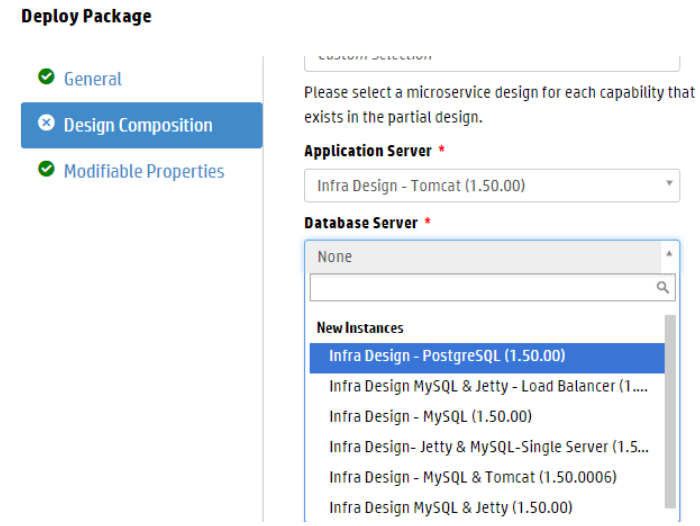


Figure 8 b: Database server infrastructure designs filtered to list the matched capabilities based on partial design



Benefits

When an application QA engineer deploys the application using partial design, the UI displays multiple infrastructure designs as shown in Figure 7 a and b. As per the characteristics of this feature, the UI can display all the infrastructure designs that contain the application and database servers and those that satisfy the requirements or match the partial design capabilities as shown in Figure 8 a and b.

Without this feature, the application QA engineer will be given a list of all available infrastructure designs that may not match the application design and users may need to search for the matching infrastructure on which they want to deploy the application. This feature saves lot of time and avoids lot of complexities.

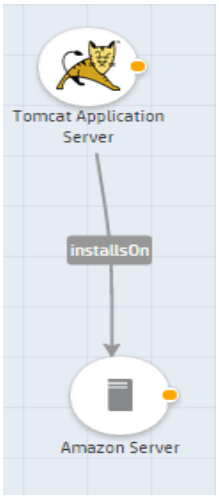
Use case 2

The second advantage of the topology composition feature is that it prevents users from deploying an application on an infrastructure design not meant to be used by a persona other than the one designated to use it. For example, an application developer or the application QA team must not deploy the application on an infrastructure design meant for a production environment. An application release manager will not wrongly choose the infrastructure design designated only for development and testing purposes.

Application developer

Consider a developer who wants to deploy an application and execute some unit tests. The developer's team may not want Tomcat and MySQL installed on two different servers. They may choose to install them in a single server from AWS or OpenStack or in an existing infrastructure or Docker where they can crash and burn or use servers already provisioned. In this case, the application architect creates an infrastructure designs similar to the designs in figure 9 and 10.

Figure 9: Single-tier AWS & OpenStack infrastructure



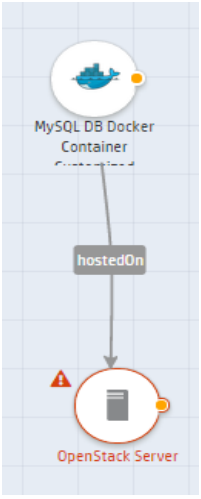
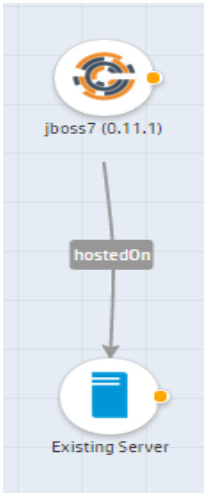
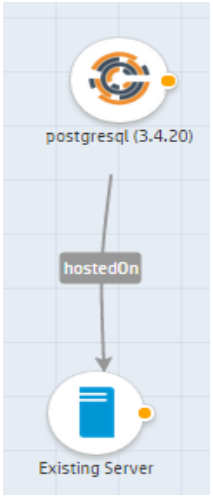


Figure 10: Single-tier existing infrastructure





Application QA

An application QA engineer may deploy the same application to test on multiple infrastructures that may contain Tomcat, MySQL, JBoss, PostgreSQL, and Jetty Server. The application architect can create multiple designs to support these combinations. The QA team can use the infrastructure designs in a combination of designs depicted in Figures 7 a and b, and Figures 9 and 10.

Application release manager

An application release manager may deploy the same application on a complex infrastructure that involves a load balancer configuration. Typically, in the stage and production environments, a load balancer is available to balance the requests and to test the application on a load balancer environment. The application architect may need to create an infrastructure design that contains an additional component to configure a software load balancer on the server as shown in Figure 11.

Figure 11: Partial design that contains application and database components with the Apache load balancer

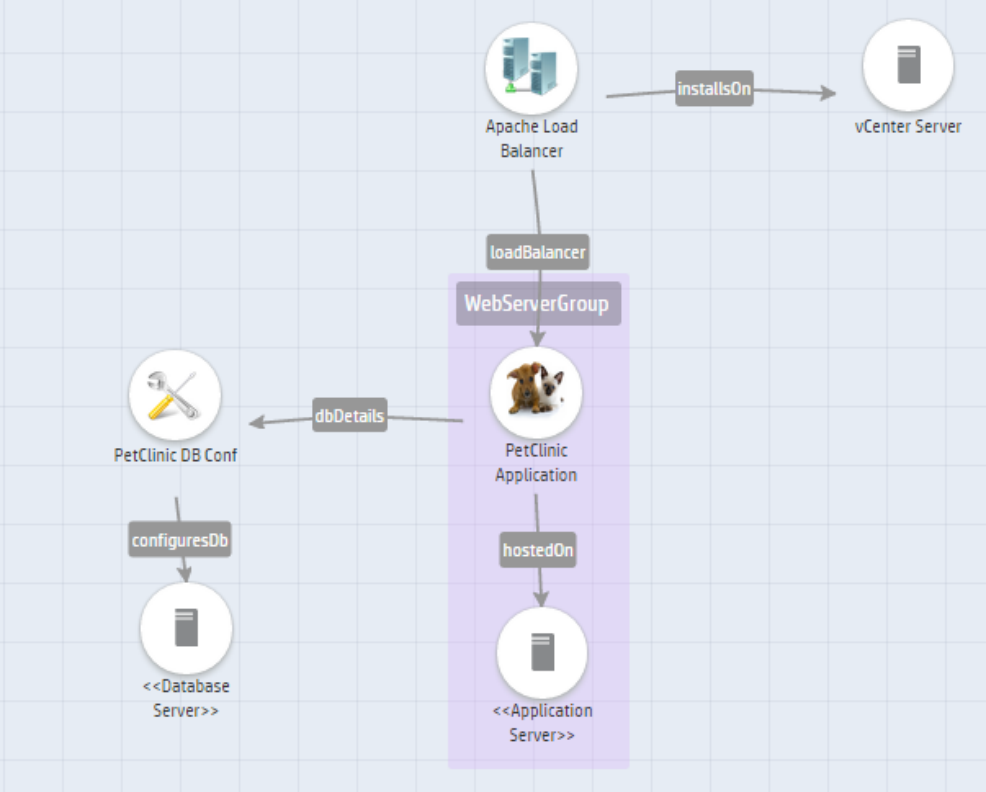
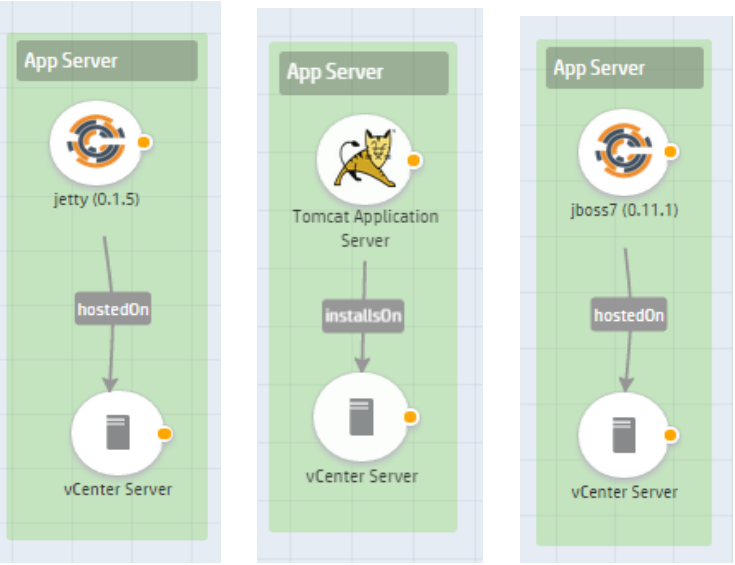


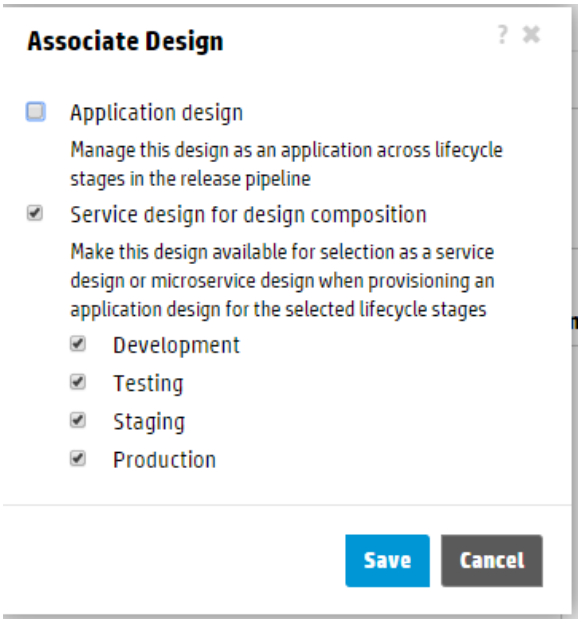
Figure 12: Micro service or application server infrastructure designs that can cater to the partial design above



Associate lifecycle with environments

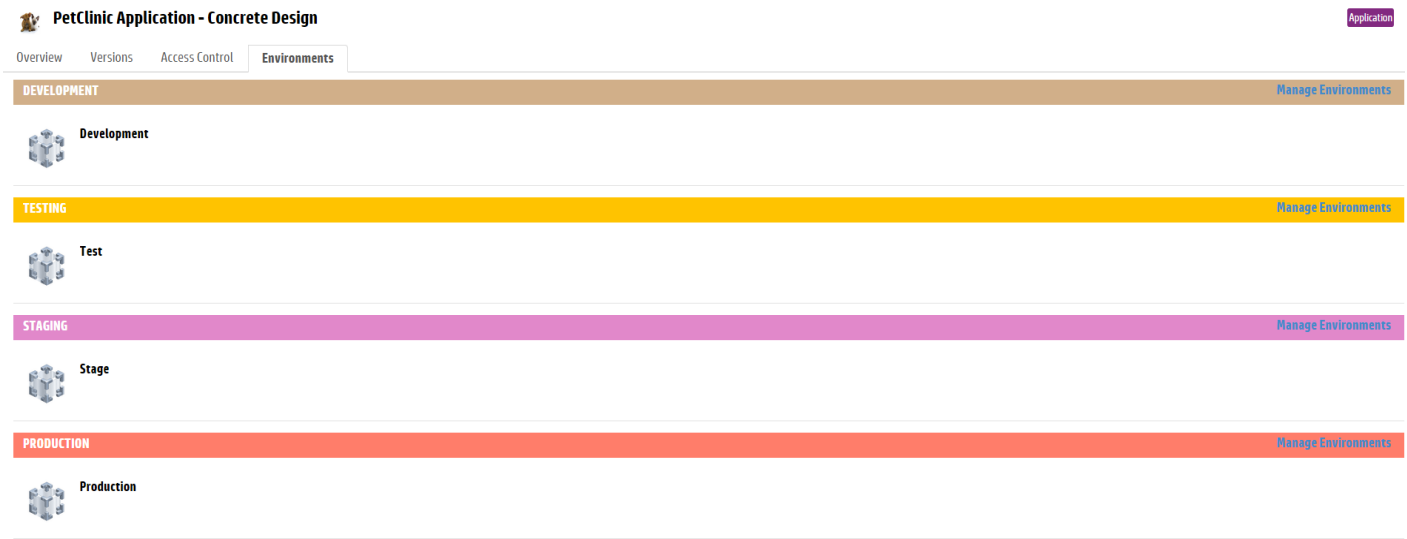
All the infrastructure designs or micro services must be associated with the lifecycle stages as follows:

Fig: 13: Associate infrastructure design with the lifecycle stages



Based on the association between micro service designs and environment, the infrastructure designs are displayed only on those lifecycle stages.

Fig: 14: Associate environments with lifecycle



Benefits

While deploying an application, a developer is given an option to choose the infrastructure design that is created for the development environment, a tester is given an option to choose the infrastructure design created for the test environment, and an application release manager is given an option to choose between the stage and production environments. This way a developer will not choose environments that are associated with other lifecycle stages and a release manager will not choose an incorrect development environment infrastructure.

Figure 14 a: Packages view on all the lifecycle stages (captured from admin view)

Figure 14 b: PetClinic application partial design package view for a developer

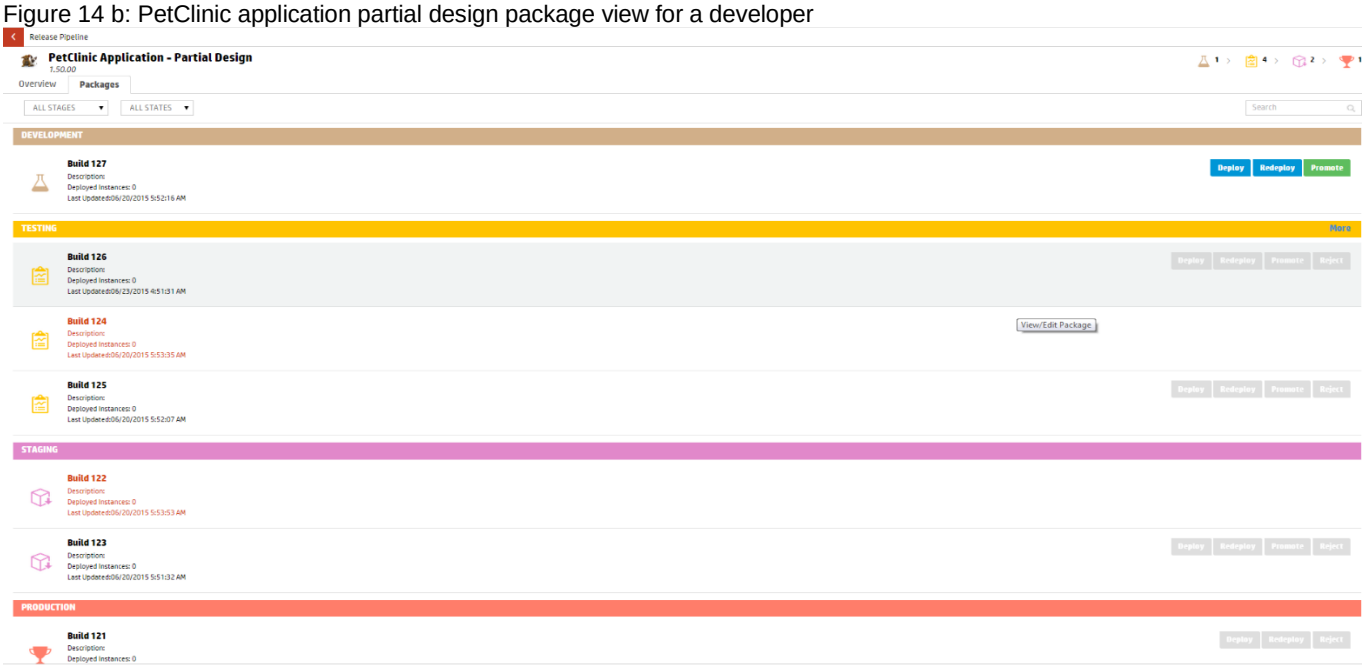


Figure 14 c: PetClinic application partial design package view for a tester/QA

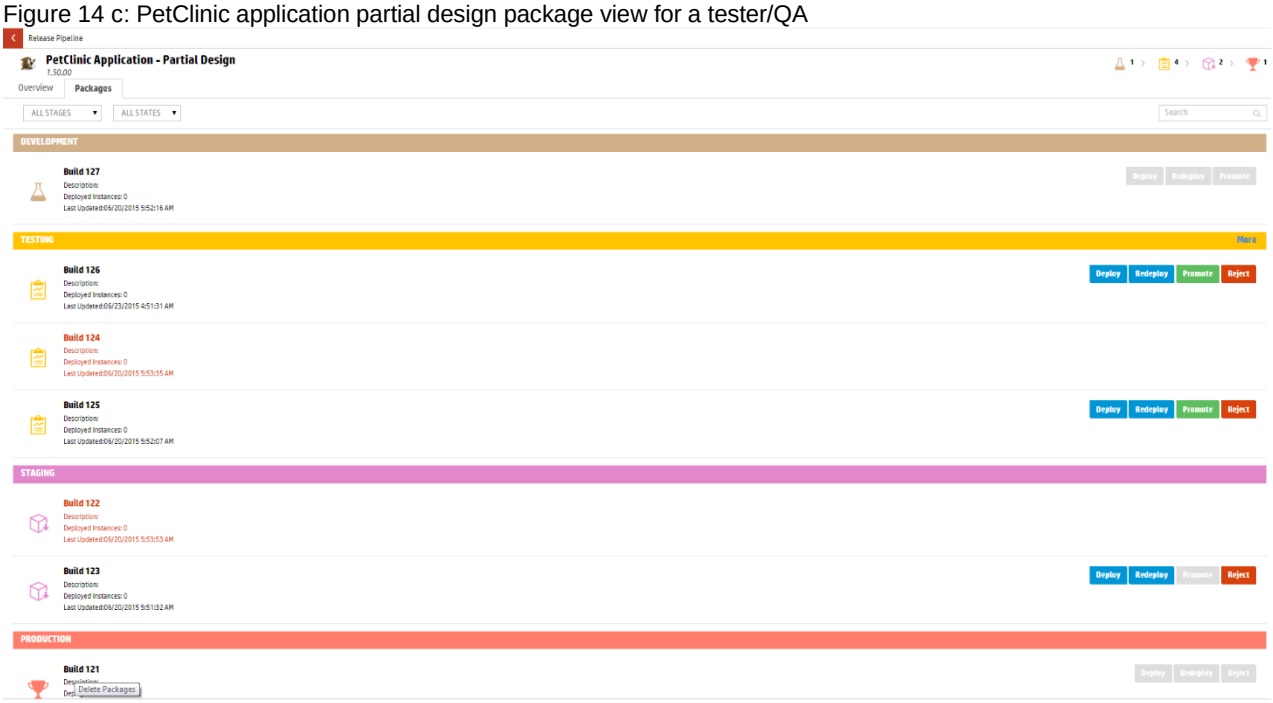


Figure 14 d: PetClinic application partial design package view for an application release manager

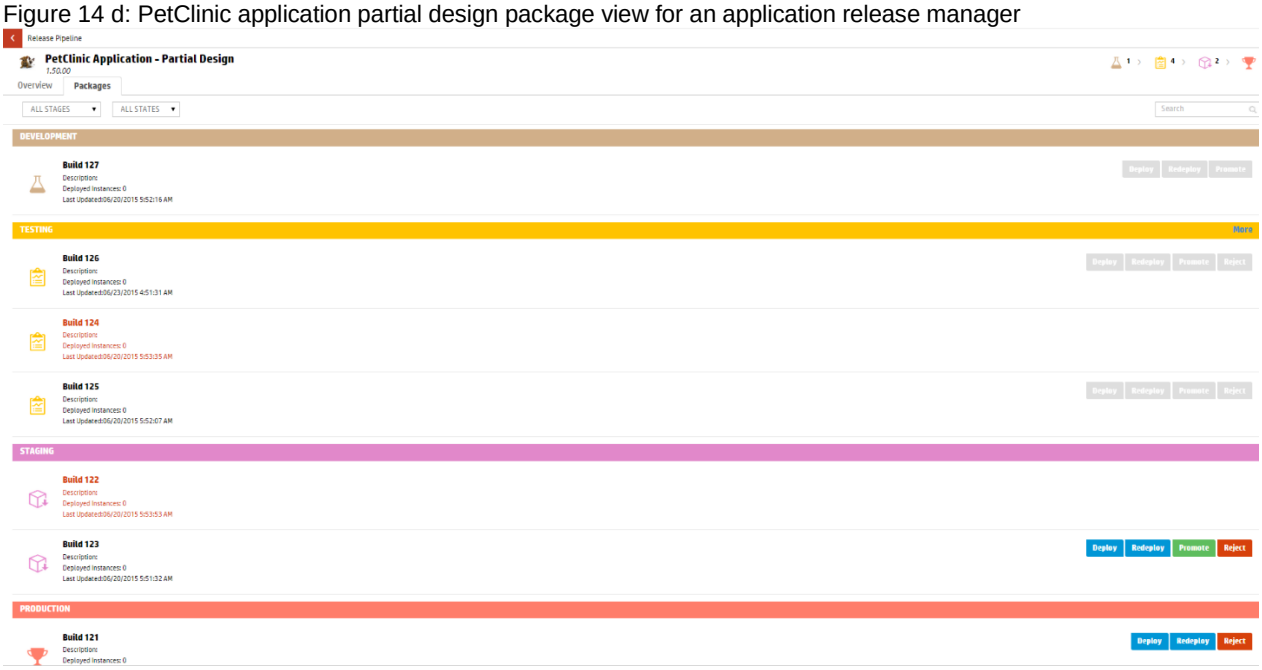
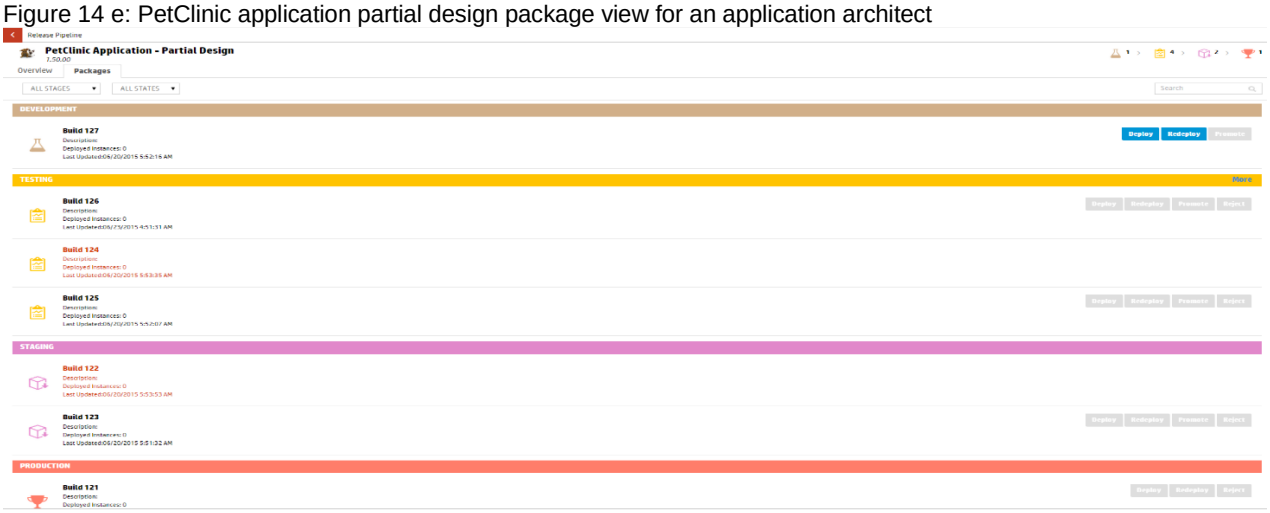
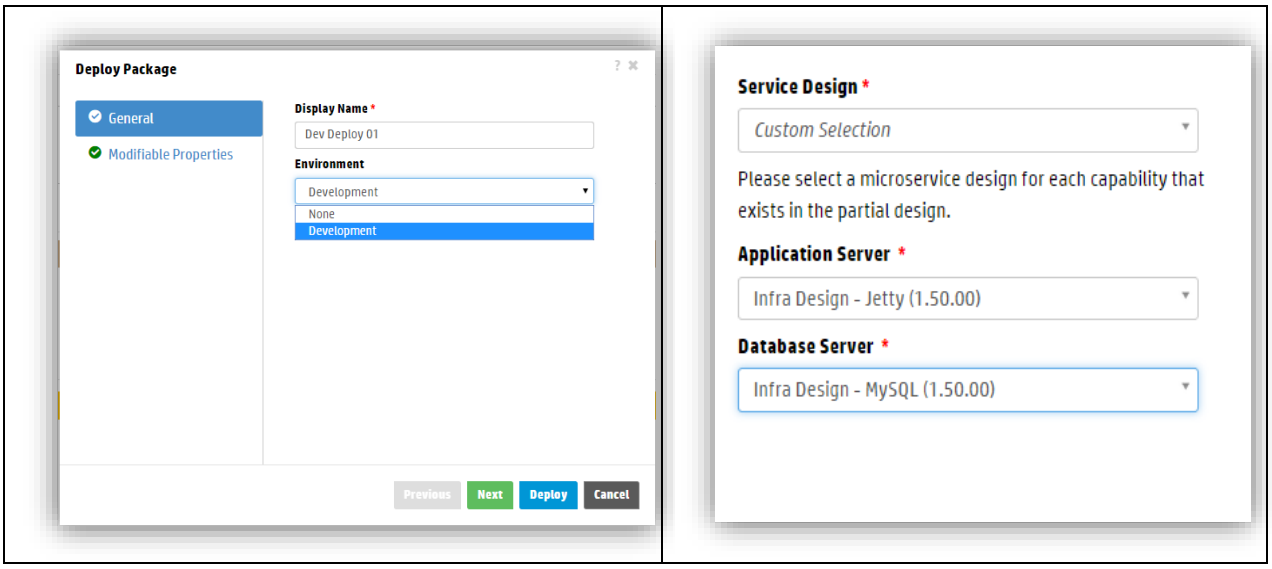


Figure 14 e: PetClinic application partial design package view for an application architect



When a developer deploys the application using partial design, only the infrastructure design created for the development environment is displayed as shown in Figure 15.

Figure 15: Infrastructure designs listed for a developer while deploying the application on a development environment



When the QA team deploys the application, infrastructure designs created for the test environment is displayed.

Note: The QA team will have access to deploy the application on the stage environment; however, it will not have the privilege to promote the package to the next stage.

Figure 16: Infrastructure designs listed for a tester deploying on a test environment

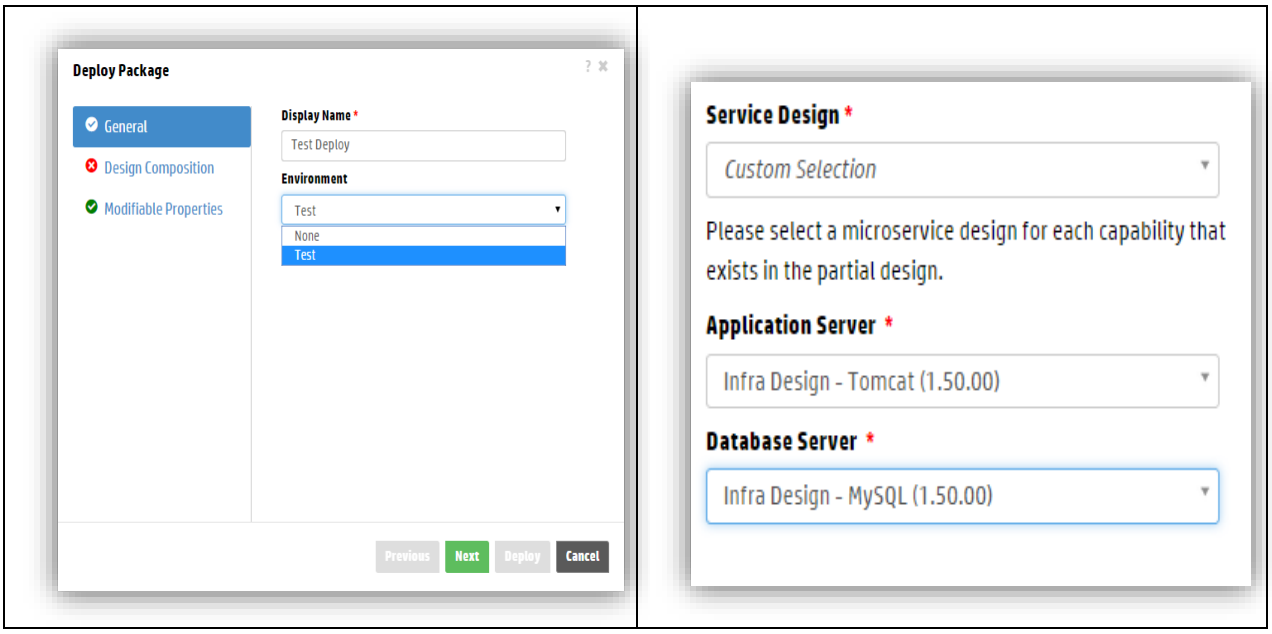
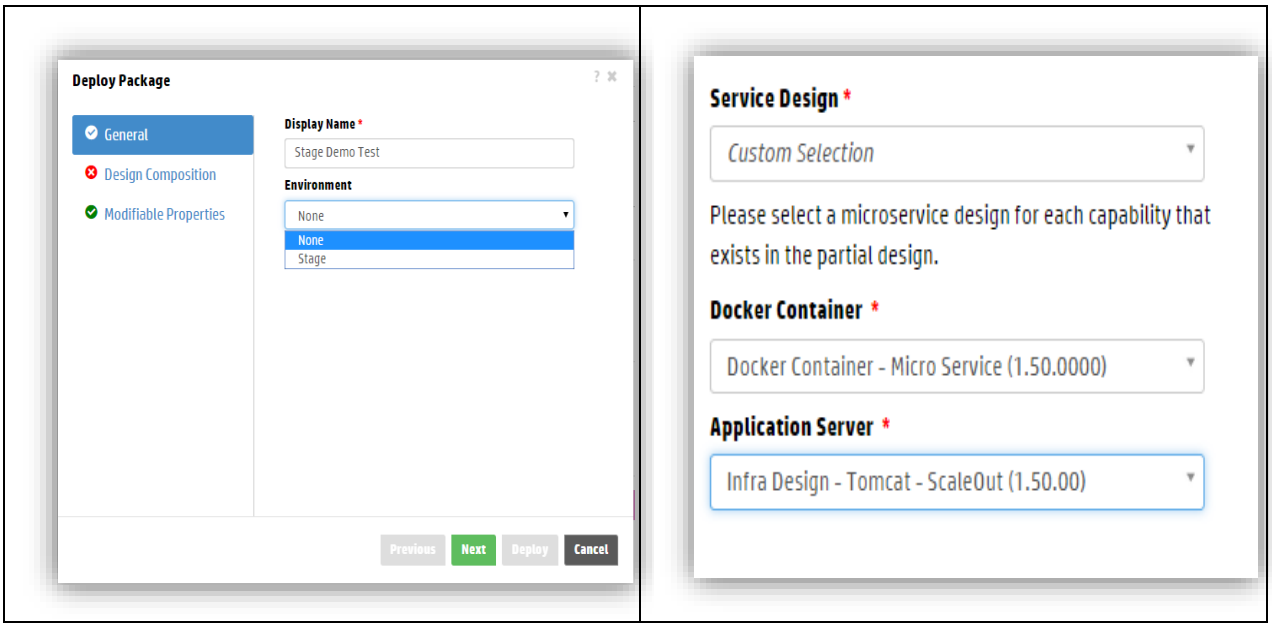


Figure 17: Infrastructure designs listed for an application release manager deploying on the staging or production environment



Conclusion

The topology composition feature helps end users choose the right infrastructure that must be used for the lifecycle environment and prevents developers and testers from deploying their application on elevated environments that are more secure. This feature helps end users list all the micro services that match the infrastructure design based on the capabilities configured in partial design.