DevOps

DevOps is a set of practices that bridge application development and operational behavior to reduce time to market without compromising on quality and operational effectiveness. It allows application developers and business owners to quickly respond to customer needs, develop a quicker feedback cycle, and ultimately achieve business value faster.

DevOps encourages a culture of collaboration between development, quality, and operations teams to reduce or eliminate barriers through fundamental practices such as continuous integration, continuous delivery, and continuous deployment. Adopting these practices and the tools to support them creates a standardized deployment process so that you can deploy predictable, high-quality releases.

Pega Platform provides the tools necessary to support continuous integration, delivery, and deployment through Deployment Manager, which provides a low-code, model-driven experience to configure and run continuous integration and delivery (CI/CD) workflows or deployment pipelines for your application. Deployment Manager provides out-of-the-box tools to enforce best CI/CD practices for your application. You can fully automate deployment pipelines, starting with automated integration of developer changes through branch merging and validation, application packaging, artifact repository management, deployments, test execution, guardrail compliance, and test coverage enforcement.

Pega Platform also includes support for open DevOps integration using popular third party tools such as Jenkins and Microsoft Azure DevOps by providing an open platform, with all the necessary hooks and services. With open DevOps integration, you can build a deployment pipeline using third-party tools to automate branch merging, application packaging and deployment, test execution, and quality metric enforcement.

For more information about configuring DevOps workflows, see the following topics:

- Understanding DevOps concepts

  The software industry evolves quickly and organizations are forced to rapidly respond to their changing markets. DevOps is a set of industry practices that have evolved to support this demand for rapid change. The primarily goal of DevOps is to reduce an organization’s time-to-market for new features without compromising quality or end-user-experience. DevOps, like the Pega Platform, allows organizations to focus their efforts on adding customer value.

- Applying DevOps concepts in the Pega Platform

  Use DevOps practices such as continuous integration and continuous delivery to quickly move application changes from development through testing to deployment on your production system. Use Pega Platform tools and common third-party tools to implement DevOps.

- Using Deployment Manager for model-driven DevOps

  Deployment Manager is the standard way to test and deploy Pega applications. It exposes all capabilities of the Pega Platform necessary to automate your DevOps workflows, including branch merging, application packaging, applying quality gates, and promoting an application to different environments. Deployment Manager leverages market-leading case management technology to build and run continuous integration and continuous delivery (CI/CD) pipelines with a familiar, model-driven experience.

- Automating deployment pipelines using Open DevOps solutions

  Use DevOps practices such as continuous integration and continuous delivery to quickly move application changes from development, through testing, and to deployment. Use Pega Platform tools and common third-party tools to implement DevOps.

- Testing Pega applications

  Having an effective automation test suite for your application in your continuous delivery DevOps pipeline ensures that the features and changes that you deliver to your customers are of high-quality and do not introduce regressions.

Understanding DevOps concepts

The software industry evolves quickly and organizations are forced to rapidly respond to their changing markets. DevOps is a set of industry practices that have evolved to support this demand for rapid change. The primarily goal of DevOps is to reduce an organization’s time-to-market for new features without compromising quality or end-user-experience. DevOps, like the Pega Platform, allows organizations to focus their efforts on adding customer value.

DevOps components

Adopting DevOps is an ongoing journey as opposed to a limited engagement project. There should be no point when you should stop seeking to improve efficiency. An evaluation of three main components is required when embarking on the DevOps journey:

- **People:** Before embarking on a DevOps journey, it’s important to get buy-in from the entire organization. The cultural shift to embrace change and shared-responsibility is one of the first challenges which needs to be overcome

before auditing process and adopting new technologies.
- **Process:** Process is what makes each organization's software delivery process unique. Processes should be regularly reviewed for bottlenecks and automation opportunities.
- **Technology:** Technology is what has made DevOps transformations possible and widely available. Technology enables automation of nearly every aspect of the software lifecycle from planning, to production, to monitoring.

- Understanding development

  Pega Platform developers use Agile practices to create applications and commit the changes into branches in a shared development environment. Automated and manual testing provides rapid feedback to developers so that they can improve the application.

- Understanding continuous integration

  With continuous integration, application developers frequently check in their changes to the source environment and use an automated build process to automatically verify these changes. Continuous integration identifies issues and pinpoints them early in the cycle. Use Jenkins with the prpcServiceUtils tool and the execute test service to automatically generate a potentially deployable application and export the application archive to a binary repository such as JFrog Artifactory.

- Understanding continuous delivery

  With continuous delivery, application changes run through rigorous automated regression testing and are deployed to a staging environment for further testing to ensure that there is a high confidence the application is ready to deploy on the production system.

- Understanding deployment

  After an application change passes the testing requirements, use Jenkins and the prpcServiceUtils tools to migrate the changes into production after complete validation through automated testing on the staging system. Use application release guidelines to deploy with minimal downtime.

Understanding development

Pega Platform developers use Agile practices to create applications and commit the changes into branches in a shared development environment. Automated and manual testing provides rapid feedback to developers so that they can improve the application.

Follow these best practices to optimize the development process:

- Leverage multiple built-on applications to develop and process smaller component applications. Smaller applications move through the pipeline faster and are easier to develop, test, and maintain.
- Create one Pega Platform instance as a source environment that acts as a single source of truth for the application. This introduces stability into the developer environment and ensures that a problem in one developer environment does not affect other environments.
- Use Pega Platform developer tools, for example:
  - The rule compare feature allows you to see the differences between two versions of a specific rule.
  - The rule form search tool allows you to find a specific rule in your application.
- Follow branch-based development practices:
  - Developers can work on a shared development environment or local environments.
  - Content in branches migrates from the development environments to merge into the source environment.
  - Create an archive by exporting and storing backup versions of each branch in a separate location in the application repository. If a corrupted system state requires you to restore the source environment to a previous known good application version, the branches can be down-merged to reapply the changes in those branches that were lost as part of the restore.
  - Use unit tests to ensure quality.
- Ensure that the work on a ruleset is reviewed and that the changes are validated. Lock every complete and validated ruleset.
- Regularly synchronize the development environments with the source environment.

For more information, see the following articles:

- Application development
  - Using multiple built-on applications
  - Searching for a rule
  - Checking out a rule
  - Checking in a rule
  - Comparing rule versions
  - Understanding best practices for version control in the DevOps pipeline
- Branching
  - Rule development in branches
  - Merging branches into target rulesets
  - Using the RuleSet Stack Landing Page Lock and Roll features for managing ruleset versions
  - Adding a branch from a repository

    - Publishing a branch to a repository
    - Creating a toggle
- Testing
    - Pega Platform application testing in the DevOps pipeline
    - PegaUnit testing

Understanding continuous integration

With continuous integration, application developers frequently check in their changes to the source environment and use an automated build process to automatically verify these changes. Continuous integration identifies issues and pinpoints them early in the cycle. Use Jenkins with the prpcServiceUtils tool and the execute test service to automatically generate a potentially deployable application and export the application archive to a binary repository such as JFrog Artifactory.

During continuous integration, maintain the following best practices:

- To automatically generate a valid application, properly define the application Rule-Admin-Product rule and update the rule whenever the application changes. The prpcServiceUtils tool requires a predefined Rule-Admin-Product rule.
- To identify issues early, run unit tests and critical integration tests before packaging the application. If any one of these tests fails, stop the release pipeline until the issue is fixed.
- Publish the exported application archives into a repository such as JFrog Artifactory to maintain a version history of deployable applications.

For more information, see the following articles and help topics:

- PegaUnit tests
    - Running test cases and suites with the Execute Tests service
- Application packaging
    - Packaging a release on your development environment
    - Using prpcServiceUtils and Jenkins for automated application deployment

Understanding continuous delivery

With continuous delivery, application changes run through rigorous automated regression testing and are deployed to a staging environment for further testing to ensure that there is a high confidence the application is ready to deploy on the production system.

Use Jenkins with the prpcServiceUtils tool to deploy the packaged application to test environments for regression testing or for other testing such as performance testing, compatibility testing, acceptance testing, and so on. At the end of the continuous delivery stage, the application is declared ready to deploy to the production environment. Follow these best practices to ensure quality:

- Use Docker or a similar tool to create test environments for user acceptance tests (UAT) and exploratory tests.
- Create a wide variety of regression tests through the user interface and the service layer.
- Check the tests into a separate version control system such as Git.
- If a test fails, roll back the latest import.
- If all the tests pass, annotate the application package to indicate that it is ready to be deployed. Deployment can be done either automatically with Jenkins and JFrog Artifactory or manually.

For more information, see the following articles and help topics:

- Deploying to a staging system
    - Deploying application changes to your staging or production environment
    - Using prpcServiceUtils and Jenkins for automated application deployment
    - Using restore points to enable error recovery

Understanding deployment

After an application change passes the testing requirements, use Jenkins and the prpcServiceUtils tools to migrate the changes into production after complete validation through automated testing on the staging system. Use application release guidelines to deploy with minimal downtime.

For more information, see the following articles and help topics:

- Deploying to the production system
    - Understanding best practices for version control in the DevOps pipeline
    -
    - Deploying application changes to your staging or production environment
    - Using prpcServiceUtils and Jenkins for automated application deployment
    - Migrating application changes
    - Understanding application release changes, types, and processes
- Enabling changes to the production system
    - Updating access groups by submitting a request to an active instance

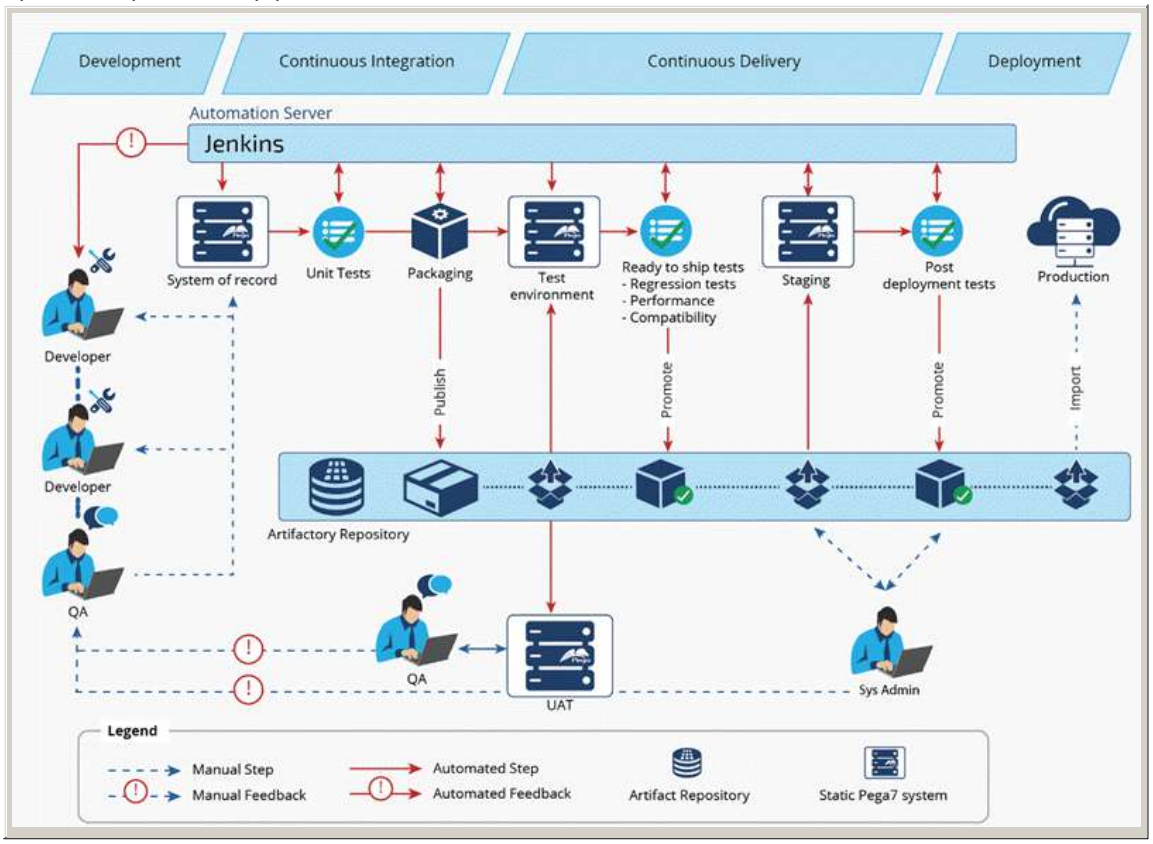
Applying DevOps concepts in the Pega Platform

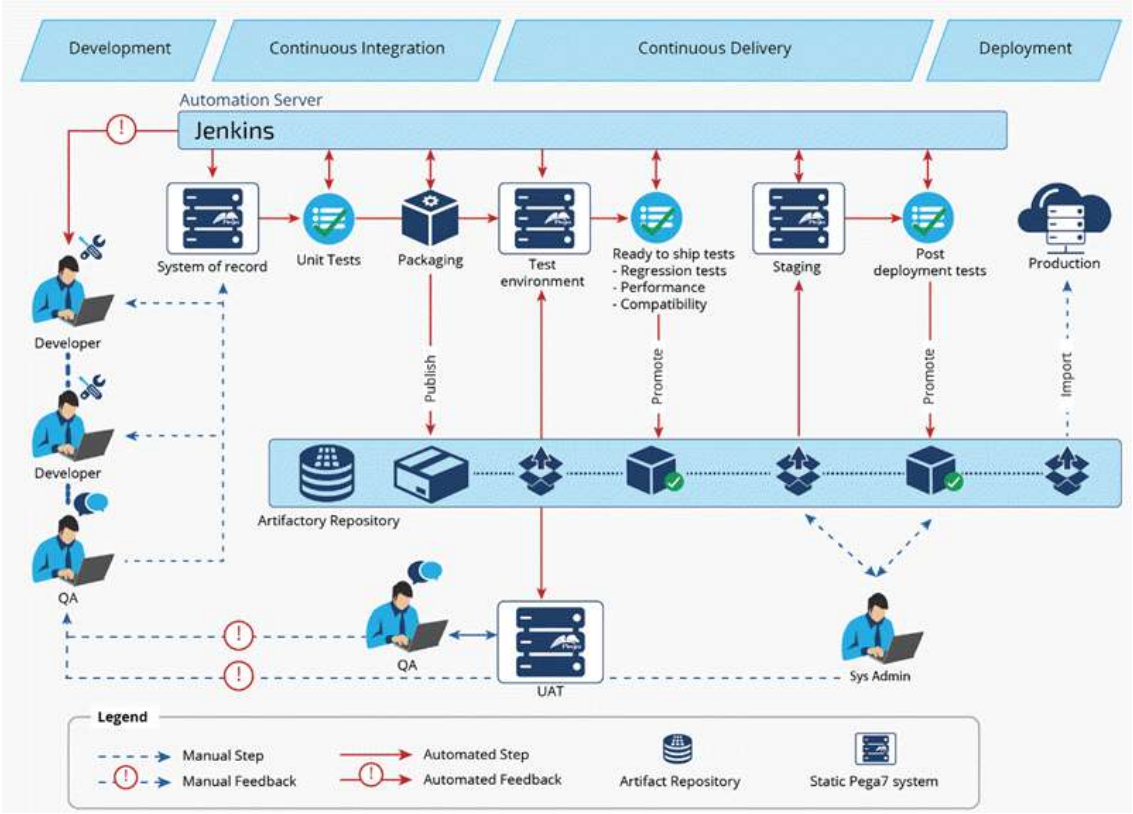
Use DevOps practices such as continuous integration and continuous delivery to quickly move application changes from development through testing to deployment on your production system. Use Pega Platform tools and common third-party tools to implement DevOps.

The release pipeline in the following diagram illustrates the best practices for using Pega Platform for DevOps. At each stage in the pipeline, a continuous loop presents the development team with feedback on testing results. This example includes the following assumptions:

- Pega Platform manages all schema changes.
- Jenkins is the automation server that helps to coordinate the release pipeline, and JFrog Artifactory is the application repository; however, other equivalent tools could be used for both.

Open DevOps release pipeline overview





- [Developing Pega applications]

  Pega Platform developers use Agile practices to create applications and commit the changes into branches in a shared development environment. Automated and manual testing provides rapid feedback to developers so that they can improve the application.

- [Understanding the DevOps release pipeline]

  Use DevOps practices such as continuous integration and continuous delivery to quickly move application changes from development through testing to deployment on your production system. Use Pega Platform tools and common third-party tools to implement DevOps.

- [Development workflow]

  In a DevOps workflow, the most important best practice for application developers to adopt is continuous integration. Continuous integration is the process by which development changes to an application are integrated as frequently as possible, at least once a day and preferably multiple times a day, every time developers complete a meaningful unit of work.

- [Understanding best practices for version control in the DevOps pipeline]

  As a best practice, use semantic versioning when changing the version number, because it offers a logical set of rules about when to increase each version number.

- [Migrating application changes]

  With minimal disruption, you can safely migrate your application changes throughout the application development life cycle, from development to deployment on your staging and production environments. In the event of any issues, you can roll back the deployment and restore your system to a state that was previously known to be working.

- [Deploying application changes to your staging or production environment]

  As part of the Standard Release process, after you set up and package a release on your shared development environment, you can deploy your application changes to your staging or production environment.

- [Packaging a release on your development environment]

  As part of the Standard Release process for migrating your application changes from development to production, you set up and package the release on your shared development environment.

- [Understanding application release changes, types, and processes]

  The following tables provide information about the types of changes that you can make within a release, the release types, and the release management process to follow based on the types of changes that you want to deploy.

Developing Pega applications

Pega Platform developers use Agile practices to create applications and commit the changes into branches in a shared development environment. Automated and manual testing provides rapid feedback to developers so that they can improve the application.

Follow these best practices to optimize the development process:

- Leverage multiple built-on applications to develop and process smaller component applications. Smaller applications move through the pipeline faster and are easier to develop, test, and maintain.
- Create one Pega Platform instance as a source environment that acts as a single source of truth for the application. This introduces stability into the developer environment and ensures that a problem in one developer environment does not affect other environments.
- Use Pega Platform developer tools, for example:
  - The rule compare feature allows you to see the differences between two versions of a specific rule.
  - The rule form search tool allows you to find a specific rule in your application.
- Follow branch-based development practices:
  - Developers can work on a shared development environment or local environments.
  - Content in branches migrates from the development environments to merge into the source environment.
  - Create an archive by exporting and storing backup versions of each branch in a separate location in the application repository. If a corrupted system state requires you to restore the source environment to a previous known good application version, the branches can be down-merged to reapply the changes in those branches that were lost as part of the restore.
  - Use unit tests to ensure quality.
- Ensure that the work on a ruleset is reviewed and that the changes are validated. Lock every complete and validated ruleset.
- Regularly synchronize the development environments with the source environment.

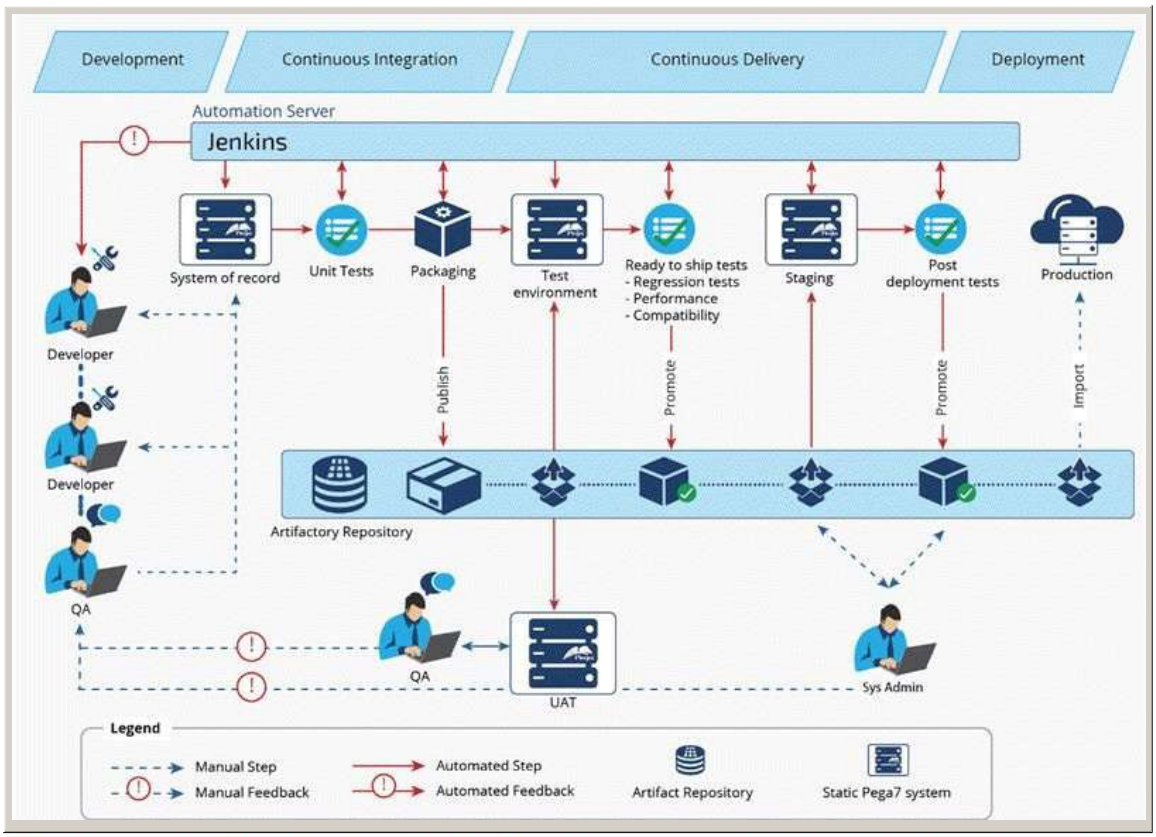
Understanding the DevOps release pipeline

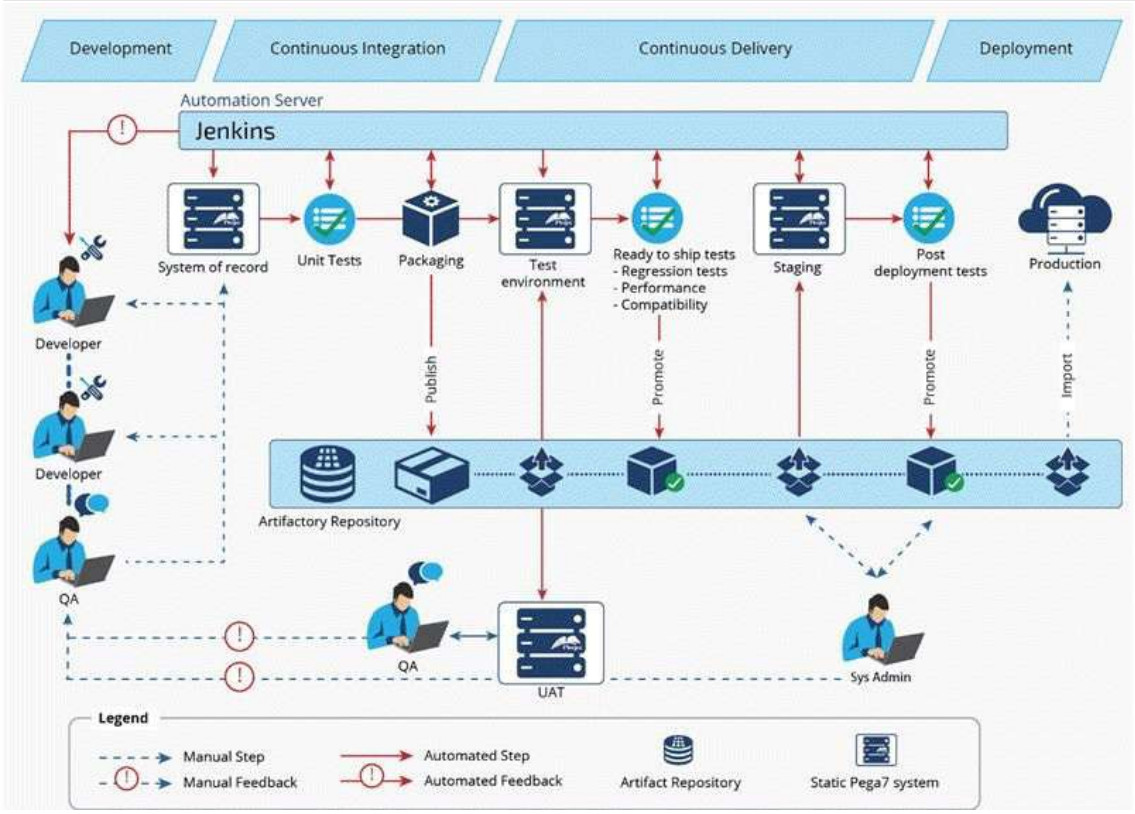
Use DevOps practices such as continuous integration and continuous delivery to quickly move application changes from development through testing to deployment on your production system. Use Pega Platform tools and common third-party tools to implement DevOps.

The release pipeline in the following diagram illustrates the best practices for using Pega Platform for DevOps. At each stage in the pipeline, a continuous loop presents the development team with feedback on testing results. This example includes the following assumptions:

- Pega Platform manages all schema changes.
- Jenkins is the automation server that helps to coordinate the release pipeline, and JFrog Artifactory is the application repository; however, other equivalent tools could be used for both.

  Open DevOps release pipeline overview





Development workflow

In a DevOps workflow, the most important best practice for application developers to adopt is continuous integration. Continuous integration is the process by which development changes to an application are integrated as frequently as possible, at least once a day and preferably multiple times a day, every time developers complete a meaningful unit of work.

To enforce best practices when developing an application and to ensure that application changes are of high quality, developers should use Pega Platform features such as branches. Before merging branches and integrating changes, developers should also verify that the application meets guardrail compliance and that unit tests pass. If the validation of

development changes passes, the branch is merged into the application ruleset.

However, if validation fails, then the merge is rejected, and developers should be notified so that they can address the failure and resubmit their changes. The feedback cycle of validating and integrating development changes should be as fast as possible, preferably 15 minutes or less, because it increases productivity in the following ways:

- Developers do not spend unnecessary time to see that their changes are valid, which enables them to make incremental changes.
- Incremental changes tend to be easier to validate, debug, and integrate.
- Other developers spend reduced time coordinating making changes and can be confident that they are building on validated functionality.

How you implement best practices for continuous integration depends on whether you have a smaller scale development with one or two scrum teams using a shared development environment or multiple distributed development teams. See the following topics for more information:

- Understanding development best practices working in a shared environment

  Development environments can be shared by one or more teams collaborating on the production application. To practice continuous integration, use a team application layer, branches, and release toggles.

- Understanding development best practices in a distributed development environment with multiple teams

  If you have multiple teams working on the same application, each team should have a separate, remote development server on which developers work. A central Pega Platform server acts as a source development system, which allows teams to integrate features into the application in a controlled manner and avoid unexpected conflicts between teams working in the same rulesets.

- Adding a branch from a repository

  If you are working in a continuous integration and delivery (CI/CD) pipeline, you can add a branch from a repository to your development application. You cannot add a branch that contains branched versions of a ruleset that is not in your application stack.

- Publishing a branch to a repository

  If you are using a continuous integration and delivery (CI/CD) pipeline with third-party tools such as Jenkins, you can publish a branch from your development application to a Pega repository on the main development system (remote system of record) to start a merge.

- Understanding rule rebasing

  If you are using continuous integration in a CI/CD pipeline with either Deployment Manager or third-party automation servers such as Jenkins, after you merge branches, you can rebase your development application to obtain the most recently committed rulesets. Rebasing allows development teams working in separate development environments to share their changes and keep their local rule bases synchronized. Having the most recently committed rules on your development system decreases the probability of conflicts with other development teams.

Understanding development best practices working in a shared environment

Development environments can be shared by one or more teams collaborating on the production application. To practice continuous integration, use a team application layer, branches, and release toggles.

- Build a team application layer that is built on top of the main production application. The team application layer contains branches, tests, and other development rulesets that are not intended to go into production. For more information, see Using multiple built-on applications.
- Create a branch of your production ruleset in the team application. For more information, see Adding development branches to your application.
- **Optional:** Use release toggles to disable features that are not ready for general use. Using toggles allows you to merge branch content frequently even if some content is not final. For more information, see Toggling features on and off.
- **Optional:** Create formal review tasks for other members of the development team to review your content. For more information, see Creating a branch review.
- **Optional:** Use the branch developer tools to review the content and quality of your branch. For more information, see Reviewing branches.
- **Optional:** Lock the branch. For more information, see Locking a branch.
- Frequently merge the branch from the team application layer to the production rulesets. For more information, see Merging branches into target rulesets.

Note: No more than two or three scrum teams should share a development environment to improve version control stability.

Understanding development best practices in a distributed development

environment with multiple teams

If you have multiple teams working on the same application, each team should have a separate, remote development server on which developers work. A central Pega Platform server acts as a source development system, which allows teams to integrate features into the application in a controlled manner and avoid unexpected conflicts between teams working in the same rulesets.

Development systems

Follow these best practices on the remote development systems:

- Multiple teams can share development systems, which can depend upon geographical distribution of teams, system load, risk of teams making system-wide changes, and demand for system restarts.
- Build a team application layer that is built on top of the main production application. The team application layer contains branches, tests, and other development rulesets that are not intended to go into production. For more information, see Using multiple built-on applications.
- Put all necessary configuration information for the development server in a development application that you can maintain, package, and deploy on demand so that you can quickly start up new remote development systems.
- Create a branch of your production ruleset in the team application. For more information, see Adding branches to your application.
- Name the branch with the feature or bug ID from your project management tool so that you can associate changes with a corresponding feature or bug.
- Perform all development work in the branch in versioned rules. Use branches for targeted collaboration and so that you can use development best practices such as conducting branch reviews and monitoring application quality metrics. You can also quickly test and roll back changes before you merge branches.
- Do not do branch development on the source development system to avoid having multiple versions of the same branch on the both the source development system and remote development system. The same branch might contain different contents that conflict with each other.
- Avoid developing rules in unlocked rulesets. Lock rulesets to ensure that rules are not accidentally and directly changed, because changes should be introduced only when branches are merged. Use a continuous integration server such as Deployment Manager to ensure that passwords to locked rulesets are not shared publicly. For more information, see Configuring ruleset version settings.
- **Optional:** Use release toggles to disable features that are not ready for general use. Using toggles allows you to merge branch content frequently even if some content is not final. For more information, see Toggling features on and off.
- **Optional:** Create formal review tasks for other members of the development team to review your content. For more information, see Creating a branch review.
- **Optional:** Use the branch developer tools to review the content and quality of your branch. For more information, see Reviewing branches.
- **Optional:** Lock the branch before you migrate it to the source development system. For more information, see Locking a branch.
- Avoid deleting a branch before you migrate it into the main development system.
- Delete a branch after you import it into the main development system so that you do not import older data or rule instances and unintentionally merge them into the main application.
- Maintain a branch only as long as you need it. The longer that you keep a branch, the likelihood increases that the branch will have conflicts, which can be difficult to resolve.
- Be aware that you cannot make some rule updates in branches, such as updates to application records, classes, libraries, and schema. Senior application architects on the team should make these changes directly on the main development system.

Source development system

Follow these best practices on the source development system:

- Use an established and reliable backup and restore process.
- Maintain high availability on the source development system so that development teams are not affected by extended periods of downtime.
- Limit and restrict developer access to the main development system so that developers cannot make impromptu application changes without going through the DevOps workflow.

Adding a branch from a repository

If you are working in a continuous integration and delivery (CI/CD) pipeline, you can add a branch from a repository to your development application. You cannot add a branch that contains branched versions of a ruleset that is not in your application stack.

1. In the navigation panel, click **App**, and then click **Branches**.
2. Right-click the application into which you want to import a branch and select **Add branch from repository**.
3. In the **Add branch from repository** dialog box, from the **Repository** list, select the repository that contains the branch that you want to import.
4. In the **Branch name** field, press the Down Arrow key and select the branch that you want to import.

5. Click **Import**.
6. Click **OK**.
7. If you are using multiple branches, reorder the list of branches so that it matches the order in which rules should be resolved.
   For more information, see Reordering branches.
8. Create rules and add them to your branch.
   When you create rules, you can select the branch and ruleset into which you want to save them. Rulesets are automatically created. For more information, see Rule development in branches.

Publishing a branch to a repository

If you are using a continuous integration and delivery (CI/CD) pipeline with third-party tools such as Jenkins, you can publish a branch from your development application to a Pega repository on the main development system (remote system of record) to start a merge.

For more information, see *Using continuous integration and delivery builds with third-party tools* on Pega Community:

1. In the navigation panel, click **App**, and then click **Branches**.
2. Right-click the branch that you want to push to a repository and click **Publish to repository**.
3. In the **Push branch to repository** dialog box, select the repository from the **Repository** list.
4. Click **Publish**. **Result:** You receive a message if the repository is not configured properly or is down, and you cannot push the branch to the repository.
5. Click **Close**.

Understanding rule rebasing

If you are using continuous integration in a CI/CD pipeline with either Deployment Manager or third-party automation servers such as Jenkins, after you merge branches, you can rebase your development application to obtain the most recently committed rulesets. Rebasing allows development teams working in separate development environments to share their changes and keep their local rule bases synchronized. Having the most recently committed rules on your development system decreases the probability of conflicts with other development teams.

For example, you can publish a branch from your development application to a Pega repository on a source development system, which starts a job on Jenkins as your automation server and merges branches. You can also use the Merge branches wizard to start a Deployment Manager build by first merging branches in a distributed or nondistributed, branch-based environment. After the merge is completed, you can rebase the rulesets on your development application to obtain the merged rulesets.

- Configuring settings for rebasing

  Before you can rebase your development system, you must first configure a Pega repository and then enable ruleset versions for them. You must also have the appropriate permissions for rebasing.

- Enabling the Pega repository type

  When you use continuous integration and delivery (CI/CD pipelines) third-party automation servers, you use Pega Platform as a binary repository for rule artifacts during development. You also use Pega repositories when you are rebasing your development application when you are using third-party automation servers or Deployment Manager.

- Enabling ruleset versions for Pega repositories for rebasing

  When you rebase rules, you must enable ruleset versions for Pega repositories so that they can host ruleset versions. To enable ruleset versions, configure the HostedRulesetsList dynamic system setting on the remote development system on which you are merging branches.

- Rebasing rules to obtain latest versions

  If you are using a continuous integration and continuous delivery (CI/CD) pipeline with Deployment Manager or third-party automation servers such as Jenkins, you can rebase your development application to obtain the most recently committed rulesets through Pega repositories after you merge branches on the source development system.

Configuring settings for rebasing

Before you can rebase your development system, you must first configure a Pega repository and then enable ruleset versions for them. You must also have the appropriate permissions for rebasing.

1. If you are using Pega repositories with third-party automation servers such as Jenkins, enable the Pega repository type.
   You do not need to enable Pega repositories if you are using Deployment Manager. For more information, see Enabling the Pega repository type.
2. Create a connection to a Pega type repository that supports ruleset version artifacts. For more information, see Adding a Pega repository.
3. Enable ruleset versions for Pega repositories by configuring the *HostedRulesetsList* dynamic system setting on the

system of record. For more information, see Enabling ruleset versions for Pega repositories for rebasing.
4. Ensure that you the *pxCanRebase* privilege so that you can rebase rules. This privilege is associated with the sysadmin4 role.
   If you do not have this privilege, you can add it to your role. For more information, see Specifying privileges for an Access of Role to Object rule.

Enabling the Pega repository type

When you use continuous integration and delivery (CI/CD pipelines) third-party automation servers, you use Pega Platform as a binary repository for rule artifacts during development. You also use Pega repositories when you are rebasing your development application when you are using third-party automation servers or Deployment Manager.

If you are using Deployment Manager, the Pega repository type is already enabled; otherwise, you must first enable it for your application by completing the following steps:

1. Click **Records Decision When** and open the *pyCanRebase* rule that applies to *@baseclass*.
2. Click **Save As Specialize by class or ruleset** .
3. Choose a ruleset in your application, then click **Create and open**.
4. In the Conditions tab, click **Actions Edit** and change the condition to true.
5. Click **Submit**.
6. Click **Save**.
7. If you are rebasing rules to refresh your development system with the latest rulesets that are hosted on a remote development system, enable ruleset versions for Pega repositories. For more information, see Enabling ruleset versions for the Pega repository.

Enabling ruleset versions for Pega repositories for rebasing

When you rebase rules, you must enable ruleset versions for Pega repositories so that they can host ruleset versions. To enable ruleset versions, configure the *HostedRulesetsList* dynamic system setting on the remote development system on which you are merging branches.

1. Complete one of the following tasks:
   - Open the *HostedRulesetsList* dynamic system setting if it exists.
     1. Click **Records SysAdmin Dynamic System Settings** .
     2. Click the record with the HostedRulesetsList Setting Purpose and the Pega-ImportExport Owning Ruleset.
   - Create this record if it does not exist.
     1. Click **Create SysAdmin Dynamic System Settings** .
     2. Enter a short description.
     3. In the **Owning Ruleset** field, enter Pega-ImportExport.
     4. In the **Setting Purpose** field, enter HostedRulesetsList.
     5. Click **Create and open**.
2. On the **Settings** tab, in the **Value** field, enter a comma-separated list of the rulesets on the remote development system. Enclose each ruleset value within quotation marks, for example, “HRApp.”
3. Click **Save.**

- Enabling the Pega repository type
- Understanding rule rebasing

Rebasing rules to obtain latest versions

If you are using a continuous integration and continuous delivery (CI/CD) pipeline with Deployment Manager or third-party automation servers such as Jenkins, you can rebase your development application to obtain the most recently committed rulesets through Pega repositories after you merge branches on the source development system.

Before you begin: To rebase rules, you must first merge branches to make changes to rules. Changes made to rules in an unlocked ruleset version will not be visible by the rebase functionality. **Note:** Only one rebase event at a time is supported per development system to prevent accidentally overriding a rebase event that is in progress.**Note:** You can improve rebase performance by frequently incrementing the application patch version. To rebase rules, complete the following steps:

1. If you are migrating and merging branches separately, manually migrate branches for an application that has a new major or minor version. Rebase only pulls the ruleset version that is visible to your current application. **For example:** For example, if you previously migrated a branch for an application of version 1.x but are now working on a 2.x application version, migrate the 2.x branch ruleset to the main development system before rebasing. Otherwise, rebase refreshes your development system with the 1.x ruleset versions.
2. In the header of Dev Studio click the name of your application, and then click **Definition**.
3. On the **Definition** tab, click **Get latest ruleset versions**.
4. In the **Select repository** list, select the repository from which to retrieve rules to see a list of ruleset versions that will be rebased.
5. Click **Rebase**.
   - If there are no import conflicts, your development application is refreshed with the rules.
   - If there are import conflicts, the system displays them. For example, a conflict can occur if you made a change to

the same ruleset version on your local development system or if you modified a non-resolved rule in the ruleset, such as the Application record. To resolve a conflict, complete the following step.

6. If you have conflicts, you must resolve them before rebasing continues, either by overwriting or rejecting the changes on your development system. Complete the following steps to import the ruleset and either overwrite or reject the changes that you made to the ruleset on the development system:
   a. For each ruleset, click the **Download Archive** link and save the .zip file.
   b. Click the **Click here to launch the Import wizard** link at the top of the Rebase rule form to open the Import wizard, which you use to import the .zip files. For more information, see Importing rules and data by using the Import wizard.
   c. In the wizard, specify whether to use the older version of the ruleset or overwrite the older version with the newer version.
   d. After you resolve all conflicts, restart the rebase process by starting from step 1.

- Understanding rule rebasing

Understanding best practices for version control in the DevOps pipeline

As a best practice, use semantic versioning when changing the version number, because it offers a logical set of rules about when to increase each version number.

When you use semantic versioning, the part of the version number that is incremented communicates the significance of the change. Additional information about semantic versioning is available on the web.

The version number, in the format NN-NN-NN, defines the major version (first two digits), minor version (middle digits), and patch version (last digits), for example, 03-01-15.

- Major versions include significant features that might cause compatibility issues with earlier releases.
- Minor versions include enhancements or incremental updates.
- Patch versions include small changes such as bug fixes.

Rulesets include all versions of each rule. Skimming reduces the number of rules by collecting the highest version of rules in the ruleset and copying them to a new major or minor version of that ruleset, with patch version 01. For more information about skimming, see Rule skimming for higher ruleset versions.

Best practices for development

Follow these best practices for version control in development:

- Work in branches.
- Consider creating a major version of your application if you upgrade your application server or database server to a major new version.
- For small single scrum teams:
  - Increment both the patch and the minor version during every merge.
  - Developers merge into the next incremented patch version.
- For multiple scrum teams:
  - Assign a user the role of a release manager, who determines the application version and release strategy. This user should be familiar with concepts and features such as rulesets, ruleset versioning, application records, and application migration strategies.
  - The release manager selects a development ruleset version number that includes a patch version number.
  - Developers merge into the highest available ruleset version.
  - Frequently increment ruleset versions to easily track updates to your application over time.
  - Maintain an application record that is capped at major and minor versions of its component rulesets.

Best practices for deployment

Follow these best practices when you deploy your application to production:

- Define target ruleset versions for production deployment.
- Use lock and roll to password-protect versions and roll changes to higher versions. For more information, see RuleSet Stack tab.
- Increment the ruleset version every time you migrate your application to production, unless the application is likely to reach the patch version limit of 99.
- Create restore points before each deployment. For more information about restore points, see Using restore points to enable error recovery.

- Using the RuleSet Stack Landing Page Lock and Roll features for managing ruleset versions

  The RuleSet Stack landing page simplifies following the best practices of ruleset locking and application versioning. The landing page provides a summary of the locking status of the ruleset versions in your application, as well as the Lock and Roll button.

Using the RuleSet Stack Landing Page Lock and Roll features for

managing ruleset versions

The RuleSet Stack landing page simplifies following the best practices of ruleset locking and application versioning. The landing page provides a summary of the locking status of the ruleset versions in your application, as well as the Lock and Roll button.

Best practices for ruleset versioning include:

- When possible, all versions in a ruleset should remain locked. Use branch development instead of updating unlocked rulesets directly.
- Cases where unlocked rulesets may be required:
  - Reports
  - Delegated rules
  - System configurations stored in rules
- If a ruleset requires unlocked versions, then only the highest version of a ruleset should be unlocked.
- Once locked, a ruleset version should not later be unlocked.

1. In Dev Studio click **Configure Application Structure RuleSet Stack.**
2. Observe the icons in the Current Application section.
   - If a column contains an unlocked warning icon, then a version lower than the highest is unlocked. You will be able to use the Lock and Roll function on this lower ruleset version to lock, but not to roll (increment). Click the unlock warning icon to view a list detailing the currently unlocked versions. Once the lower versions are locked, you can again use the roll portion of the Lock and Roll function on the highest ruleset version.
   - If the column contains a warning icon, then the ruleset version does not exist, but is included in your application. Check your application to determine if the version of the ruleset was incorrectly entered, or if the ruleset was incorrectly entered. Once this error is corrected, then you can run the Lock and Roll function on this ruleset version, if needed.
3. Click the **Lock and Roll** button.
4. In the **Application Lock and Roll** window, in the **Lock** column, select the check box next to the rulesets that you want to lock. If a ruleset is currently locked, has a lower unlocked version, or does not exist, the corresponding icon will appear instead of a check box.
5. In each unlocked ruleset that you selected for locking, a text box appears under the **Password** column. Enter a password for the ruleset version.
   Note: The password that you enter appears in plain text to ensure that the password is entered correctly.
6. In the **Roll** column, check the box next to the rulesets which are to be rolled to a higher version. If the ruleset is not locked or selected to be locked, the check box for this ruleset will not be available.
7. Observe the value in the **Roll to Version** column. By default, the value in this column will be the next highest patch version from the current ruleset. If you are rolling a selected ruleset is to a higher minor or major version, change this value to the appropriate version number.
8. Select the **Application update option** from the list below the rulesets appropriate for the environment in which your development is taking place. For more information on these options, see Application Structure landing page.
9. Click **Run**.

   You might need to log out and back in to the application before the new version of any incremented rulesets become visible.

Migrating application changes

With minimal disruption, you can safely migrate your application changes throughout the application development life cycle, from development to deployment on your staging and production environments. In the event of any issues, you can roll back the deployment and restore your system to a state that was previously known to be working.

The process that you use to release changes to your application is different depending on the types of changes that you are making. This topic describes the Standard Release process that you can use to deploy changes to rules, data instances, and dynamic system settings. The Standard Release process is a self-service way to deploy changes without downtime. Other methods for releasing changes to your application are not covered in this article. For more information, see Application release changes, types, and processes.

This Standard Release process applies to both on-premises and Pega Cloud Services environments. As a Pega Cloud Services customer, if you use this self-service process to release changes to your application, you are responsible for those changes. For more information, see Change management in Pega Cloud Services and Service level agreement for Pega Cloud Services.

The Standard Release process includes the following steps and is scalable to the number and types of environments that you have:

1. Package the release on your shared development environment. For more information, see Packaging a release on your development environment.
2. Deploy the changes to your staging or production environment. For more information, see Deploying application changes to your staging or production environment.

- Understanding application migration requirements

Before you migrate your application, both your environment and application must meet certain requirements. For example, you must be able to download a RAP archive to a file system location with the required available space.

- Understanding application migration scenarios

  The Standard Release migration process supports the following scenarios:

Understanding application migration requirements

Before you migrate your application, both your environment and application must meet certain requirements. For example, you must be able to download a RAP archive to a file system location with the required available space.

Your environments and applications must meet the following requirements:

- You have at least two existing Pega Platform environments. These environments can be any combination of sandbox and production environments, and can be on-premises or in the Pega Cloud virtual private cloud (VPC).
- You can log in to a machine from which you will complete the release process, such as your organization's laptop, workstation, or server.
- You have a location on a file system with enough available space to store the RAP archives. You must be able to download the RAP archive to this location and upload a RAP archive to another system from this location.
- You have complied with stated application guideline and guardrail requirements.
- Your application rule must specify rulesets that have a patch version. Most application rules have the ruleset version in the stack set to 01-01, such as "Ruleset: 01-01". You must change this to specify your rulesets to the patch level of their version, which is "Ruleset: 01-01-01". This is required for your top-level application rule and all built-on applications, except the base PegaRULES built-on application. This application structure gives you greater control over the release of your software, minimizes the impact of the release on application users, and provides for the smoothest recovery path in case of a troubled deployment.

Understanding application migration scenarios

The Standard Release migration process supports the following scenarios:

- All developers in the organization use a single shared development environment (recommended by Pegasystems).
- The organization follows a distributed development model, where individual developers or development teams have their own isolated development environments.

The release process works for either development scenario, because it begins after changes have been merged into the appropriate ruleset versions. Regardless of development scenario or team size, development teams must use branching and merging for releasing applications. Otherwise, you cannot take full advantage of the tools and capabilities of the Pega Platform. For more information, see Understanding application migration scenarios.

Deploying application changes to your staging or production environment

As part of the Standard Release process, after you set up and package a release on your shared development environment, you can deploy your application changes to your staging or production environment.

This Standard Release process applies to both on-premises and Pega Cloud Services environments. As a Pega Cloud Services customer, if you use this self-service process to release changes to your application, you are responsible for those changes. For more information, see Change management in Pega Cloud Services and Service level agreement for Pega Cloud Services.

This process involves completing the following steps:

1. Deploying the application archives
2. Testing the deployment
3. Activating the release for all users

In the event of any issues, you can roll back the deployment and restore your system to a state that was previously known to be working.

Before you deploy application changes, you must know about the types of changes that you can make within a release, the release types, and the release management process to follow based on the changes you want to deploy. For example, SQL changes that remove columns from database tables or remove data types can interrupt service for users of the application. You must deploy these types of changes during scheduled downtime when users are offline. For more information, see Understanding application release changes, types, and processes.

- Deploying the application archives

  After you create the application archives, deploy them to your target system. This process is the best way to deploy changes into your staging or production environment, control their activation, and recover from problematic deployments.

- Testing the deployment

  After you deploy the changes, the release engineer and specified users can test the changes. For the staging environment, test the performance and the user interface, run automated tests, and do acceptance testing. For the production environment, perform validation tests.

- Activating the release for all users

  After your Rules archive and Data archive are successfully deployed, changes are activated in various ways. Activation is the process by which a category of changes becomes usable by appropriate users of the system, if they have access.

- Rolling back a problematic deployment

  In the event of a problematic deployment, the first goal is to prevent further issues from occurring. Then you can roll back the deployment and restore your system to a state that was previously known to be working.

Deploying the application archives

After you create the application archives, deploy them to your target system. This process is the best way to deploy changes into your staging or production environment, control their activation, and recover from problematic deployments.

The user who imports the archives must have the zipMoveImport and SchemaImport privileges on the target system.

1. Ensure that you have connectivity to both the target system and to the location where the archives are stored.
2. Use the prpcServiceUtils command line utility to import the archives to the target system:
   - Deploy the Rules archive by following the steps in Importing rules and data by using a direct connection to the database. Your changes are sent to the system, imported to the database, and ready for activation.
   - Deploy the Data archive by following the steps in Rolling back and committing tracked data. When you deploy the Data archive, you use the same tool that you used to deploy the Rule archive but with different properties. You can roll back these changes if required.

Result: For information about allowing automatic schema changes, see Editing administrator privileges for importing archives with schema changes into production.

Testing the deployment

After you deploy the changes, the release engineer and specified users can test the changes. For the staging environment, test the performance and the user interface, run automated tests, and do acceptance testing. For the production environment, perform validation tests.

Do the following steps:

1. On the target system, create a copy of the access group for your application. This step is a one-time process, because now this access group is available anytime you deploy changes.
2. Update the copied access group so that it references the new application version.
3. Find the operator ID record for a test user and give that operator ID record access to the access group that you just created.

Result: You can now safely test your changes in the system at the same time as other users who are running on the previous version.

When you are satisfied that your release was deployed successfully, Release Engineering can activate the release for all users in the production environment.

If you experience any issues, see Rolling back a problematic deployment.

Activating the release for all users

After your Rules archive and Data archive are successfully deployed, changes are activated in various ways. Activation is the process by which a category of changes becomes usable by appropriate users of the system, if they have access.

Data changes, including schema changes, take effect immediately after being imported to the system. Your application might be able to access these fields immediately. After testing and when you are sufficiently comfortable, you should commit these changes. To commit data changes, follow the steps in Rolling back and committing tracked data.

To activate rule changes, you need to update the access groups that point to the prior version of your application rule:

1. In Dev Studio, click **Records**.
2. Click **Security Access Group** .
3. Search for the access groups to be updated by specifying your application name in the search box and filtering the list.
4. After you locate the access group, open the record and increment the version number for your application to the new release version.

5. Click **Save**.

Result: If you deploy code changes that need to be compiled, you must restart the system. Code changes cannot be made without downtime, and your System Administrator must perform a system restart. For information about the types of changes that you can make within a release, the release types, and the release management process to follow, see Understanding application release changes, types, and processes.

Rolling back a problematic deployment

In the event of a problematic deployment, the first goal is to prevent further issues from occurring. Then you can roll back the deployment and restore your system to a state that was previously known to be working.

Do the following steps:

1. Ensure that no new operators can access the problematic application. You can temporarily disable access to the entire system. For more information, see How to temporarily disallow new interactive logins with a Dynamic System Setting.
2. Roll back the problematic rules changes. You can roll back changes by updating the access group for your application and specifying the previous version of your application.

Result: Roll back the data instances that you changed to their previous version. To roll back data changes, use the prpcServiceUtils command line utility. For more information, see Rolling back and committing tracked data. This process replaces those modified data instances with their prior definition, rolling back your data changes to the last known, good state.

Packaging a release on your development environment

As part of the Standard Release process for migrating your application changes from development to production, you set up and package the release on your shared development environment.

This Standard Release process applies to both on-premises and Pega Cloud Services environments. As a Pega Cloud Services customer, if you use this self-service process to release changes to your application, you are responsible for those changes. For more information, see Change management in Pega Cloud Services and Service level agreement for Pega Cloud Services.

This process involves completing the following steps:

1. Creating the release target (ruleset version)
2. Locking the release
3. Creating the application archives

After you set up and package the release, you are ready to deploy the changes to your staging or production environment.

- Creating the release target

  When developers merge changes by using the Merge Wizard, they must select the ruleset version to which to merge them. The release engineer is responsible for ensuring that each release has an unlocked ruleset version that acts as the release target and into which these merges can be performed. Developers are responsible for merging their branches into the correct, unlocked ruleset version and addressing any conflicts.

- Locking the release

  After all merges are completed, the release engineer locks the applications and rulesets to be released. They are also responsible for creating the new, higher-level ruleset versions and higher-level application rules for the next release.

- Creating the application archives

  For each release, you create one or two RAP archives, depending on the changes you made to your application. The user who exports the archives must have the zipMoveExport privilege.

Creating the release target

When developers merge changes by using the Merge Wizard, they must select the ruleset version to which to merge them. The release engineer is responsible for ensuring that each release has an unlocked ruleset version that acts as the release target and into which these merges can be performed. Developers are responsible for merging their branches into the correct, unlocked ruleset version and addressing any conflicts.

Locking the release

After all merges are completed, the release engineer locks the applications and rulesets to be released. They are also responsible for creating the new, higher-level ruleset versions and higher-level application rules for the next release.

To lock the release, do the following steps:

1. In Dev Studio, click **Application Structure Ruleset Stack** .
2. Click **Lock & Roll**.
   Note: As a best practice, lock your built-on applications first, and then work your way up the stack to your top-level application. This way, as each higher version application rule is created, you can open that rule and update the version of the built-on application.
3. For each ruleset:
   a. Click **Lock** and provide a password.
   b. Click **Roll**.
4. Click **Create a new version of my Application** .
5. Click **Run**.

Result: The application rules and ruleset versions for the current release are locked and require passwords to make changes. Also, you will have created the higher-level ruleset versions and application rules that will be used for the next release.

Creating the application archives

For each release, you create one or two RAP archives, depending on the changes you made to your application. The user who exports the archives must have the zipMoveExport privilege.

These RAP archives include:

- The Rules RAP, which contains the Rules portion of your release, instances from Rules- types only, and all rules changes.
- The Data RAP, which contains the Data portion of your release, instances from Data classes only, and all data changes.

Splitting the release into a Rules RAP and a Data RAP provides you with more control over the deployment and activation of your changes on other systems.

More RAP archives might be created during the development process. Import these RAP archives to a single system from which the Rules RAP and Data RAP will be created. This method provides the greatest level of control over the release by separating the release process from the development process.

1. Define the RAPs by using the Application Packaging wizard or by copying an existing RAP rule. For more information, see Product rules.
2. Export each RAP as an archive:
   a. Export the rules. For more information, see Exporting an application, product rule, or ruleset to an archive or repository
   b. Provide a standard name for the archive, such as **Application-01-01-02.zip**.
   c. Store these archives in a standard location to which you have access, and that will be accessible during deployment.

Result: A Rules archive and a Data archive are created as the result of this process.

Understanding application release changes, types, and processes

The following tables provide information about the types of changes that you can make within a release, the release types, and the release management process to follow based on the types of changes that you want to deploy.

Types of changes within a release

Change type	Technical changes	Activates by access group	Activates immediately	Activation requires restart	Release frequency	Requires a Support Request (Pega Cloud Services only)
Rules (including non-rule-resolved rules)	Rule- Rule-Application-Rule Rule-Obj-Class Rule-Ruleset-Name Rule-Ruleset-	Yes	Yes	No	Daily/Weekly	No

Change type	Version Rule-Access-Role-Obj Technical changes Rule-Access-Deny-Obj	Activates by access group	Activates immediately	Activation requires restart	Release frequency	Requires a Support Request (Pega Cloud Services only)
~~Data instances~~	~~Data-~~	~~No~~	~~Yes~~	~~No~~	~~Weekly~~	~~No~~
Dynamic system settings	Data-Admin-System-Settings	No	YesCertain dynamic system settings activate only on system restart and require you to follow the Environment release process.	NoCertain dynamic system settings activate only on system restart and require you to follow the Environment release process.	MonthlyCertain dynamic system settings activate only on system restart and require you to follow the Environment release process.	No
			Yes Treat functional changes that reference code as a Code release, which requires a system restart to activate if you are making code changes. • **Change type** – This column lists the high-level category of changes that you can make in a release. • **Technical changes** – Technical changes describe the rule types or artifacts for a change type. Rule- and Data- include all subtypes under that parent type, unless specifically identified for a different change type. • **Activates by access group** – Rule resolution for this change type is controlled by the access groups of an operator.	No Treat functional changes that reference code as a Code release, which requires a system restart to activate if you are making code changes. • **Change type** – This column lists the high-level category of changes that you can make in a release. • **Technical changes** – Technical changes describe the rule types or artifacts for a change type. Rule- and Data- include all subtypes under that parent type, unless specifically identified for a different change type. • **Activates by access group** – Rule resolution for this change type is controlled by the access groups of an operator. • **Activates**		

Change type Functional	Technical changes Rule-Utility-Function Rule-Utility-Library	Activates by access group Yes	• **Activates immediately** – Rule resolution uses this change type immediately after deployment. • **Activation requires restart** – This change type requires a system restart before it is available to the rule resolution process. • **Release frequency** – Release frequency indicates the period in which you can deploy this type of change to production. • **Requires a Support Request (Pega Cloud only)** – As a Pega Cloud customer, you are responsible for any application changes that you make; however, as a best practice, inform and engage Pega Support before releasing application changes. You can open a Support Request on My Support Portal. For more information, see My Support Portal FAQ.	• **Activates immediately** – Rule resolution uses this change type immediately after deployment. • **Activation requires restart** – This change type requires a system restart before it is available to the rule resolution process. • **Release frequency** – Release frequency indicates the period in which you can deploy this type of change to production. • **Requires a Support Request (Pega Cloud only)** – As a Pega Cloud customer, you are responsible for any application changes that you make; however, as a best practice, inform and engage Pega Support before releasing application changes. You can open a Support Request on My Support Portal. For more information, see My Support Portal FAQ.	Release frequency Monthly	Requires a Support Request (Pega Cloud Services only) Yes
Data model	SQL	No	Yes	No	Monthly	Yes
Code	Java JAR file Java .class file	No	No	Yes	Monthly	Yes
	Changes outside					

Environment Change type	of Pega (JVM, XML configuration) Technical changes	No Activates by access group	No Activates immediately	Yes Activation requires restart	Quarterly Release frequency	Yes Requires a Support Request (Pega Cloud Services only)

Understanding release types

Release type	Activates for users	Application users affected	Release frequency	New application version	Self-service	Requires a Support Request (Pega Cloud only)
Bug fix	Immediately	All	Daily	No	Yes	No
Standard release	On access group update	By access group	Weekly	Yes	Yes	No
Database release	Immediately	All	Monthly	Yes	No	Yes
Code release	After restart	All	Monthly	Yes	No	Yes
Environment release	After restart	All	Quarterly	Yes	No	Yes
	Per change type Activation of a Major release occurs based on the change types that the release contains. For information about how each change type is activated, see Table 1. • **Release type** – This column lists the high-level category of releases that you can deploy. • **Activates for users** – This column indicates when this release type takes effect for users. • **Application users affected** – This column provides the scope of application users that see the effect of this release type. • **Significant UX impact** – This release type might require users to significantly	Per change type Activation of a Major release occurs based on the change types that the release contains. For information about how each change type is activated, see Table 1. • **Release type** – This column lists the high-level category of releases that you can deploy. • **Activates for users** – This column indicates when this release type takes effect for users. • **Application users affected** – This column provides the scope of application users that see the effect of this release type. • **Significant UX impact** – This release type might require users to significantly relearn a process or has significant				

Release type	Activates for users	Application users affected	Release frequency	New application version	Self-service	Requires a Support Request (Pega Cloud only)
Major release	relearn a process or has significant layout changes. • **Release frequency** – This column provides the frequency of this type of release. • **New application version** – This column indicates whether you must create a new application version for this release. • **Self-service** – A user with appropriate permissions can execute this release type using the Pega Platform, and a Pega System Administrator is not required to roll back changes. • **Requires a Support Request (Pega Cloud Services only)** – As a Pega Cloud Services customer, you are responsible for any application changes that you make; however, as a best practice, inform and engage Pega Support before releasing application changes. You can open a Support Request on My Support Portal. For more information, see My Support Portal FAQ.	layout changes. • **Release frequency** – This column provides the frequency of this type of release. • **New application version** – This column indicates whether you must create a new application version for this release. • **Self-service** – A user with appropriate permissions can execute this release type using the Pega Platform, and a Pega System Administrator is not required to roll back changes. • **Requires a Support Request (Pega Cloud Services only)** – As a Pega Cloud Services customer, you are responsible for any application changes that you make; however, as a best practice, inform and engage Pega Support before releasing application changes. You can open a Support Request on My Support Portal. For more information, see My Support Portal FAQ.	Quarterly	Yes	No	Yes

Understanding the process to follow based on types of changes

Does your release contain the following changes?								Follow this release process	Requires a Support Request (Pega Cloud Services only)
Rules	**Data instances**	**Dynamic system settings**	**Functional**	**Data model**	**Code**	**Environment**	**Significant UX impact**		
X	X	-	-	-	-	-	-	Bug fix	No
X	X	X	X	-	-	-	-	Standard release	No
-	X	-	-	X	-	-	-	Database release	Yes
			- Treat functional changes that reference code as a Code release, which requires a system restart to activate if you are making code changes. • **Rules** – Are you deploying Rules- records in this release? • **Data instances** – Are you deploying Data- records in this release? • **Dynamic System Settings** – Are you loading Data-Admin-System-Settings records in this release? • **Significant UX impact** – Will users need to significantly relearn a process, or are there significant layout changes? • **Code** – Are you loading JAR files as part of this release? • **Data model** – Are there changes to						

Does your release contain the following changes?									
-	-	-	your data model in this release (SQL)? • **Environment changes** – Will there be operating system or application server changes in this release? • **Follow this release process** – Based on your answers to these questions, follow this release process. • **Requires a Support Request (Pega Cloud Services only)** – As a Pega Cloud Services customer, you are responsible for any application changes that you make; however, as a best practice, inform and engage Pega Support before releasing application changes. You can open a Support Request on My Support Portal. For more information, see My Support Portal FAQ.	-	X	-	-	Code release Follow this release process	Requires a Support Request (Pega Cloud Services only) Yes
-	-	- Certain Dynamic System Settings activate only on system restart and require you to follow the Environment release	-	-	-	X	-	Environment release	Yes

		process.							Requires a Support Request (Pega Cloud Services only)
X	X	X	X	X	X	X	X	Major release	Yes

Does your release contain the following changes?

Follow this release process

Using Deployment Manager for model-driven DevOps

Deployment Manager is the standard way to test and deploy Pega applications. It exposes all capabilities of the Pega Platform necessary to automate your DevOps workflows, including branch merging, application packaging, applying quality gates, and promoting an application to different environments. Deployment Manager leverages market-leading case management technology to build and run continuous integration and continuous delivery (CI/CD) pipelines with a familiar, model-driven experience.

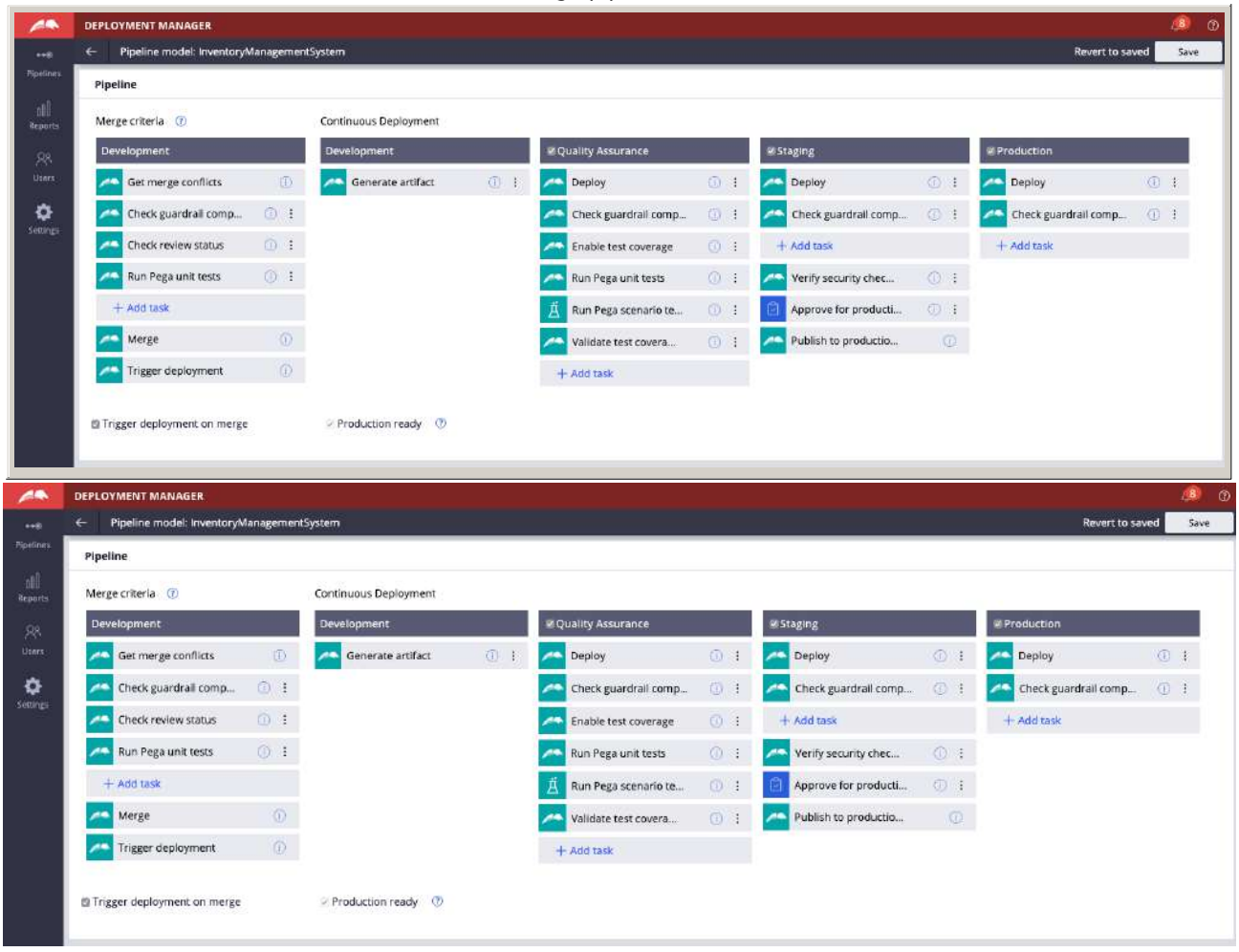
Getting started

Available to both on-premises and Pega Cloud customers, Deployment Manager makes it easy to set up and deploy your Pega Applications without the need for any third-party automation tools.

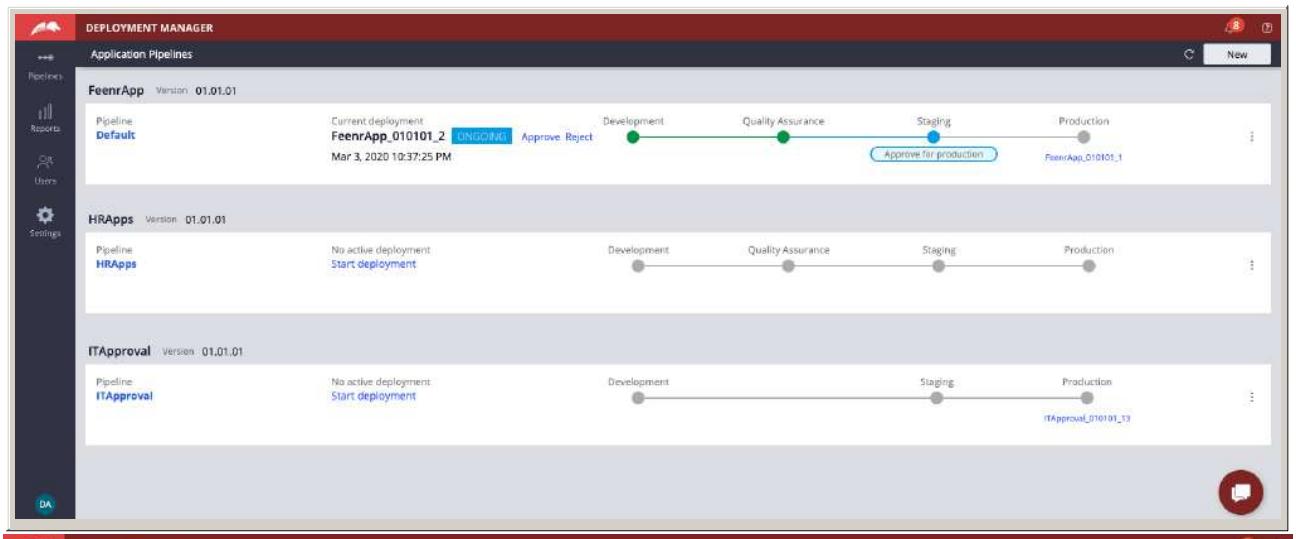
Built for Pega, by Pega, Deployment Manager integrates seamlessly with the Pega Infinity platform.

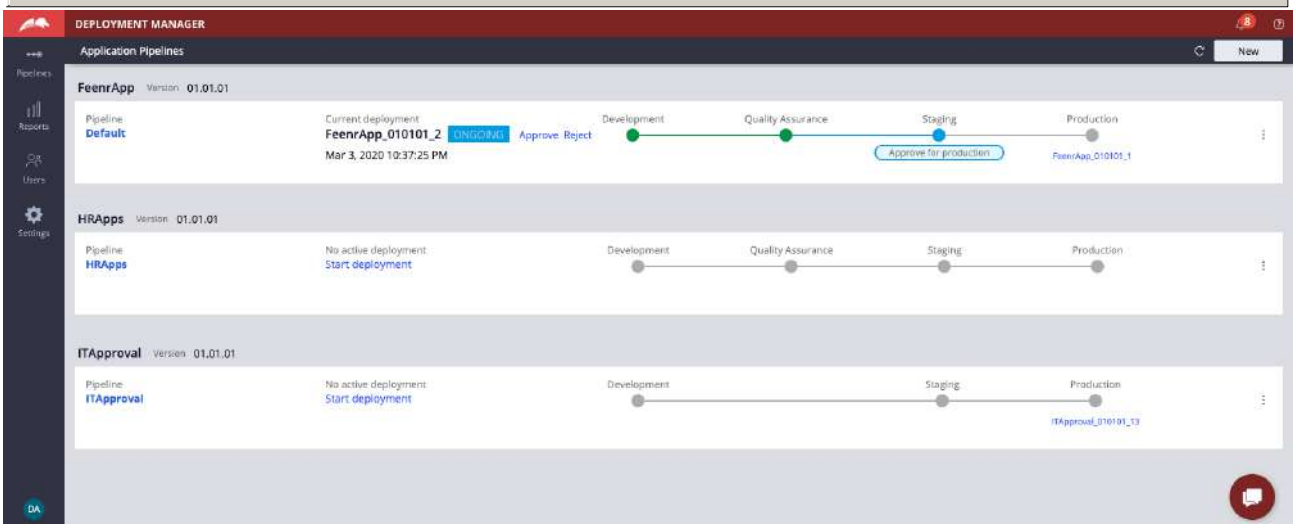
Capabilities

Defining a CI/CD pipeline using Deployment Manager is quick and easy with customizable stages and tasks, as well as default recommendations for a standard Pega pipeline.

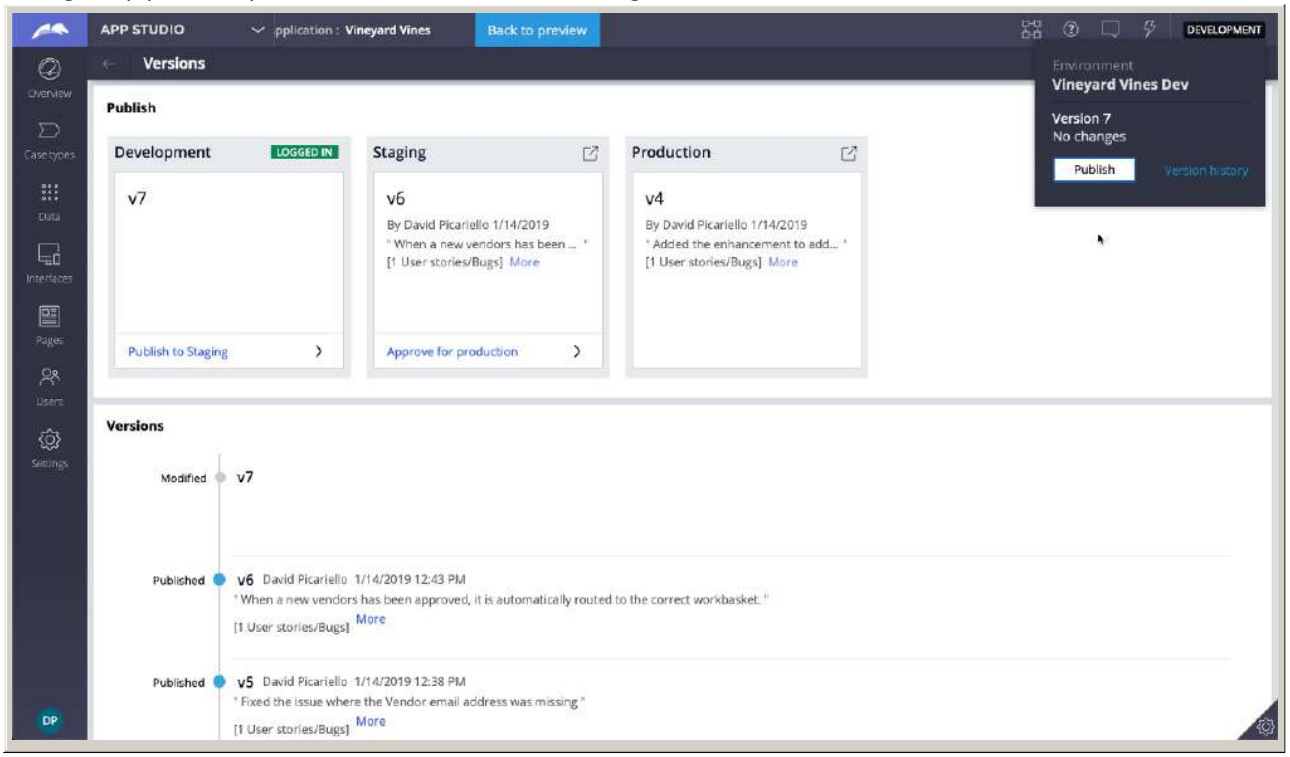


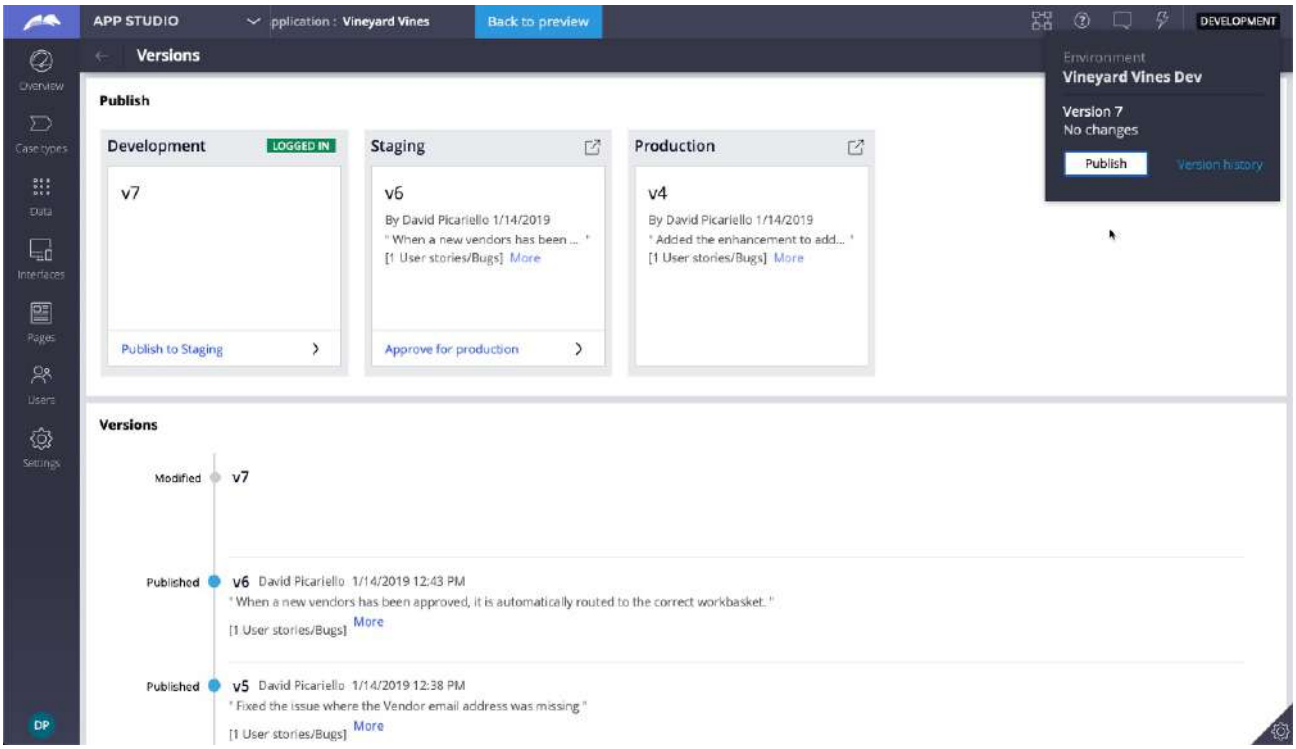
View the status of all your Pega applications and environments at a glance from the Pipelines overview page.





Integration with App Studio and Dev studio allows citizen developers and system architects to utilize Deployment Manager's pipeline capabilities from within their existing workflows.





- Getting to know the systems and components

  You should understand some of the high-level components necessary for a successful Deployment Manager implementation before installation. This article provides descriptions of each component you need.

- Installing or upgrading Deployment Manager 5.x

  You must import the Deployment Manager components if you are using it on-premises. Because Pega Cloud services manages the orchestration server in any Pega Cloud subscription, Pega Cloud services manages the installation and upgrades of Deployment Manager orchestration servers.

- Configuring pipelines and deploying applications in Deployment Manager

  Use Deployment Manager to create continuous integration and delivery (CI/CD) pipelines, which automate tasks so that you can quickly deploy high-quality software to production.

- Configuring additional settings in 5.1.x

  As part of your pipeline, users can optionally receive notifications through email when events occur. For example, users can receive emails when tasks or pipeline deployments succeed or fail. For more information about the notifications that users can receive, see .

- Accessing API documentation

  Deployment manager provides REST APIs for interacting with resources that in the Deployment Manager interface. Use these APIs to create and manage pipelines by using automated scripts or external information.

- Release notes

  These release notes provide information about enhancements, known issues, issues related to updating from a previous release, and issues that were resolved in each release of Deployment Manager.

- Past release documentation

  The Deployment Manager releases for the corresponding versions of documentation are no longer available to be downloaded from Pega Marketplace.

Getting to know the systems and components

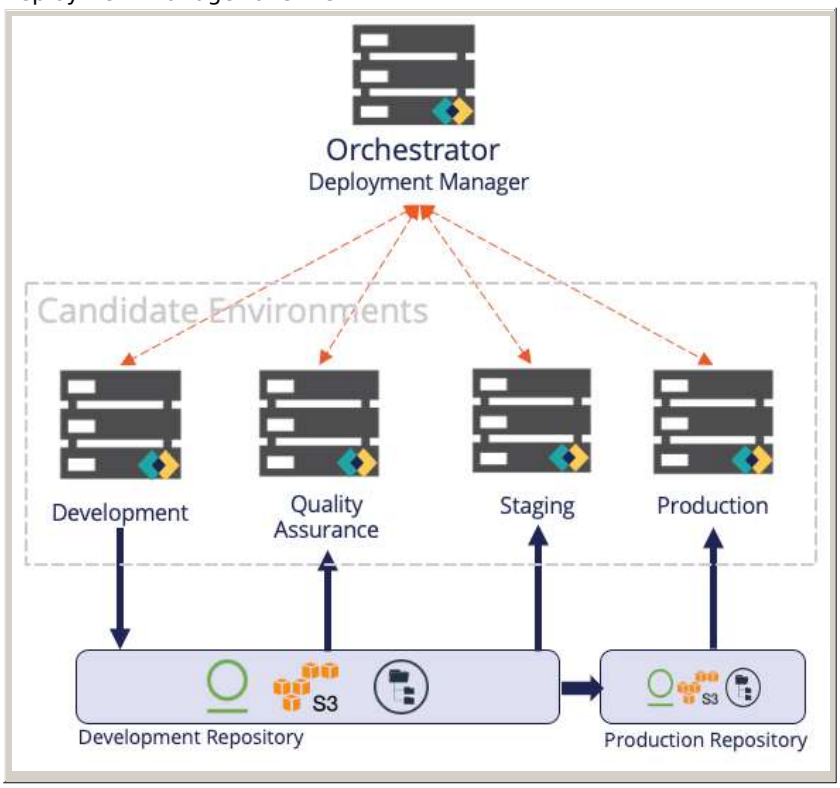
You should understand some of the high-level components necessary for a successful Deployment Manager implementation before installation. This article provides descriptions of each component you need.

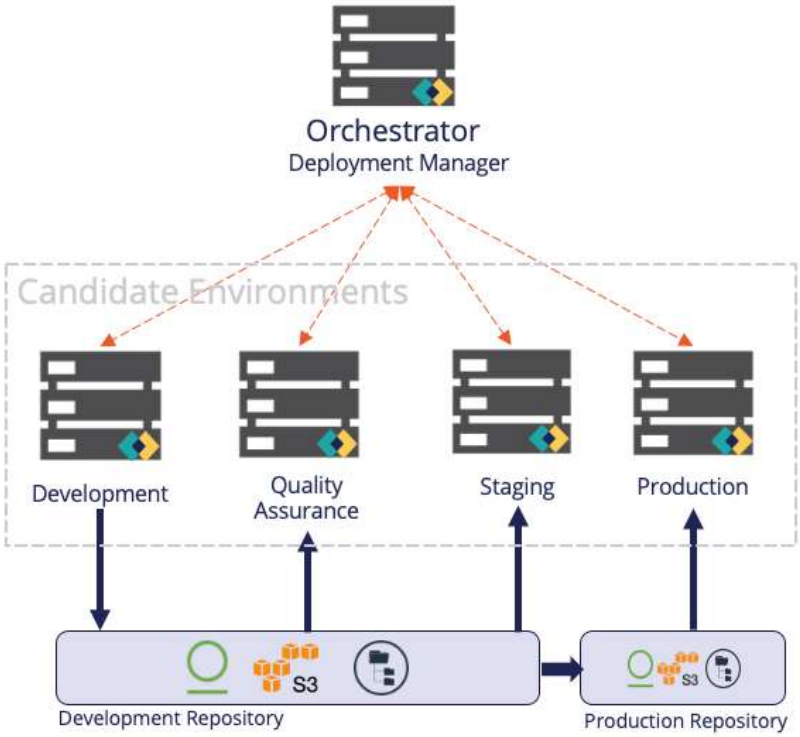
Required systems

The following systems are required for implementation:

- **Orchestrator**
  - Orchestrates all the elements of a deployment pipeline from initial branch merge to production deployment.
  - Exposes the pipeline modeling experience for users to design their deployment pipelines.
  - Resides on a dedicated Pega server which is configured to interact with each of the candidate system.
- **Candidate environments**
  - Each Pega environment managed by Deployment Manager is referred to as a Candidate system.
  - Candidate systems execute most deployment tasks at the direction of the orchestrator.
- **Development**
  - Among the candidate environments, development uses special responsibilities as it is the environment on which merges are performed, and all application packages are generated.
  - The development environment is referred to as the System of Record (SoR) to highlight these responsibilities.
- **Repositories**
  - Candidate systems use repositories to store application artifacts as they are being promoted between environments.
  - The deployment process versions application artifacts automatically.
  - A separate repository definition can store artifacts that a CICD pipeline determines “Production Ready”.

Deployment Manager overview





Rule Components

Deployment Manager is made up of two rule components

- The *PegaDeploymentManagerApplication* installed on the orchestrator environment includes all rules necessary for authoring and executing Deployment Manager pipelines. Users logging into the application will log into the Deployment Studio.
- The **PegaDevOpsFoundation** application is installed on each candidate environment. Each managed application must include **PegaDevOpsFoundation** as a built-on application. This component facilitates deployments and adds merge wizard and App Studio integration.

To get started with Deployment Manager, see Installing or upgrading Deployment Manager 5.x.

- Understanding default authentication profiles and operator IDs

  When you install Deployment Manager on all the systems in your ecosystem participating in the release process, there are applications, operator IDs, and authentication profiles installed by default. Authentication profiles enable communication between the orchestration server and candidate systems.

Understanding default authentication profiles and operator IDs

When you install Deployment Manager on all the systems in your ecosystem participating in the release process, there are applications, operator IDs, and authentication profiles installed by default. Authentication profiles enable communication between the orchestration server and candidate systems.

On the orchestration server, the following items are installed:

- The Pega Deployment Manager application.
- The DMReleaseAdmin operator ID, which release managers use to log in to the Pega Deployment Manager application. You must enable this operator ID and specify its password.
- The DMAppAdmin authentication profile. The orchestration server uses this authentication profile to communicate with candidate systems so that it can run tasks in the pipeline. You must update this authentication profile to use the password that you specified for the DMAppAdmin operator ID, which is configured on all the candidate systems.
- From version 5.x, Deployment Manager Service is the core system that the Portal and Agents are built on. The following items are installed:
  - The DMStudioUser authentication profile. The Deployment Manager Portal (GUI) uses this authentication profile to communicate with the Deployment Manager Services hosted on orchestrator system. Since the authentication profile type is oAuth2, the client secret gets updated by clicking the **Generate client secret** (Update authentication profile) button through the Deployment Manager Portal.
  - The DMAgentUser authentication profile. The Deployment Manager agent responsible for communicating task processes to candidate systems uses this authentication profile to fetch pending tasks from the Deployment Manager Services hosted on the orchestrator.

On all candidate systems, the following items are installed:

- The PegaDevOpsFoundation application.
- The DMAppAdmin operator ID, which points to the PegaDevOpsFoundation application. You must enable this operator ID and specify its password.
- The DMReleaseAdmin authentication profile. All candidate systems use this authentication profile to communicate with the orchestration server about the status of the tasks in the pipeline. You must update this authentication profile to use the password that you specified for the DMReleaseAdmin operator ID, which is configured on the orchestration server.
- From version 5.x, Deployment Manager Service is the core system that the Portal and Agents are built on. The following items are installed:
    - The DMReleaseAdmin_OAuth2 authentication profile. All candidate systems use this authentication profile to communicate with the orchestration server about the status of the tasks in the pipeline. Since the authentication profile type is oAuth2, the client secret must be updated with the client secret generated on the orchestrator (using the **Generate client secret** option from general settings on the Deployment Manager Portal).

Note: The DMReleaseAdmin and DMAppAdmin operator IDs do not have default passwords.

Installing or upgrading Deployment Manager 5.x

You must import the Deployment Manager components if you are using it on-premises. Because Pega Cloud services manages the orchestration server in any Pega Cloud subscription, Pega Cloud services manages the installation and upgrades of Deployment Manager orchestration servers.

Before you begin: Note: For Pega Cloud Services users, please see the Quick-start guide for Deployment Manager on Pega Cloud ServicesDeployment Manager 5.x has the following dependencies (see the capability matrix below for more information):

- Orchestrator is supported only on Pega Platform 8.5.2 or later.
- Orchestrator environment must have HTTPS enabled.
- Candidate environments must have Pega Platform 8.1 or later.
    - Candidate environments between Pega Platform 8.1 and 8.5.1 require Pega Deployment Manager Foundation 4.8.4.
    - Candidate environments on Pega Platform 8.5.2 and higher require Pega Deployment Manager Foundation 5.1.

To install Deployment Manager on-premises, do the following steps:

1. Navigate to the Deployment Manager Pega Marketplace page and download the latest version of Deployment Manager.
2. Extract the contents of the downloaded file.
3. Use the Import wizard to import files into the appropriate systems. For more information about the Import wizard, see Importing rules and data by using the Import wizard.
    a. On the orchestration environment, import the following files in the order below:
        - **PegaDevOpsFoundation_5.x.zip**
        - **PegaDeploymentManager_5.x.zip**
4. On the candidate environment, import the **PegaDevOpsFoundation_5.x.zip** file.
    a. If you are configuring a candidate environment on Pega Platform version 8.1 through 8.5.1, import **PegaDevOpsFoundation_8.1-04.08.x.zip** instead,
5. It is mandatory that the orchestrator environment should be restarted post-RAP imports.
6. Do one of the following actions, as applicable:
    - **If you are not upgrading** , continue the setup procedures.
    - **If you are upgrading from version 3.x** , you must first launch the Deployment Manager studio in 4.x and allow the automatic migration steps to complete, before completing the 4.x upgrade procedure.
    - **If you are upgrading from version 4.x** :
        - Update the DMReleaseAdmin access group to use SuperAdmin as the default access group.
        - Migrate pipelines and roles to use the new data model. See Migrating data to Deployment Manager 5.x for more information.

    See Securely authenticating in Deployment Manager for more information on setting up a secure communication with services.**Note:** On the first login to Deployment Manager 5.1 Studio, you see authentication errors due to an unestablished communication with Deployment Manager. Continue with the setup procedures to resolve.

Platform and Deployment Manager components

Platform and Deployment Manager components

Pega Platform	8.1 to 8.5.1	8.5.2 and later
Orchestrator 5.x		Supported
Candidate 5.x		Supported
Orchestrator 4.8.4	Supported	
	Supported (*can be managed by a	

Candidate 4.8.4 Pega Platform	5.x orchestrator on Platform 8.5.2+) 8.1 to 8.5.1	8.5.2 and later

- Quick-start guide for Deployment Manager on Pega Cloud Services

  Your Pega Cloud Services account comes with everything you need to start deploying your application to production. This guide summarizes all the steps necessary to deploy your new Pega Application to production.

- Quick-start guide for Deployment Manager on-premises and client-managed cloud

  This guide summarizes all of the required steps for getting started with Deployment Manager for on premises and client cloud deployments. For customers with familiarity with Deployment Manager and the Pega Platform it can act as a checklist for quickly setting up a new Deployment Manager implementation.

- Setting up repositories

  Deployment Manager stores application artifacts created throughout the lifecycle of a deployment within a repository. Using Deployment Manager requires the usage of a supported artifact repository.

- Setting up the orchestrator

  The orchestrator is a standalone Pega environment that allows modeling and execution of continuous integration and continuous delivery (CI/CD) pipelines.

- Setting up candidate environments

  Candidate environments are any Pega environment that a Deployment Manager Pipeline manages. Most pipelines consist of Dev, QA, Staging, and Production environments.

- Securely authenticating in Deployment Manager

  Deployment Manager 5.1 now supports an OAuth 2.0 token-based authentication process for a more secure operator experience within the orchestrator and candidate environments. This authentication and authorization model provides many benefits, including the ability to audit user operations within Deployment Manager, as all actions are now connected to an operator ID instead of a generic authentication profile, such as DMReleaseAdmin.

- Configuring an application

  There are a few requirements and recommendations for structuring an application that is being managed using Deployment Manager. See this brief overview to get started with configuration.

- Migrating data to Deployment Manager 5.x

  Use the migration assistant feature to safely migrate data, including roles, operator records, and pipelines to the version 5.x model.

Quick-start guide for Deployment Manager on Pega Cloud Services

Your Pega Cloud Services account comes with everything you need to start deploying your application to production. This guide summarizes all the steps necessary to deploy your new Pega Application to production.

Note: Determine two passwords that will be used throughout these instructions for the following operators:

- *DMReleaseAdmin*
- *DMAppAmin*

Configure the orchestration environment

1. Log in to the DevOps environment as an administrator.
2. Enable the DMReleaseAdmin operator and set the password.
3. Open the DMAppAdmin authentication profile and set the password.
4. Log in as DMReleaseAdmin.
5. Navigate to **Settings General settings** .
    1. Ensure that the orchestrator URL is correct.
    2. Click **Generate client secret**.
    3. Download the secret.
    4. Click **Update authentication profiles**.
6. Upload the JKS keystore on DMKeyStore rule. For information on keystore and truststore setup, see Securely authenticating in Deployment Manager.

Configure the candidate environments

1. Log in to each candidate environment as an administrator.
    - For the default Pega Cloud deployment, this will be dt1, stg, and prd.
2. Enable the DMAppAdmin operator and set the password.

3. Open the DMReleaseAdmin_oAuth2 authentication profile and set the client secret, access token endpoint, and revoke token endpoint downloaded from the orchestrator.
4. Upload the JKS keystore on DMKeystore rule. For information on keystore and truststore setup, see Securely authenticating in Deployment Manager.

See Setting up candidate environments for more information on configuring the candidate environments.

Configure the development environment

1. Log in to the dt1 environment as an administrator.
2. Update the *OrchestratorURL* Dynamic System Setting to the URL of the DevOps environment. The URL should end in */prweb*.
3. Open the application record with which you want to manage with a Deployment Manager pipeline, and set **PegaDevOpsFoundation version 5** as a built-on application

See Setting up candidate environments for more information on configuring the candidate environments.

Configure the application

1. The application that is being promoted by a Deployment Manager pipeline should be built-on the latest version of the PegaDevOpsFoundation application.

See Configuring an application for more information on configuring the candidate environments.

Create your first pipelines

1. Log in to the *devops* environment as DMReleaseAdmin.
2. Click **New** in the top-right corner and choose **Deployment pipeline**.
3. On the first screen, enter the following information as it applies to your application:
    1. **Application name** and **version**
    2. **Access group**
    3. **Product name** and **version** that defines your entire application.
    4. **Pipeline name**
4. Click **Create** to create the pipeline.

See Creating pipelines for more information on creating pipelines.

Test and execute the pipeline

1. In the top right corner of the rule form, in the action menu, diagnose the pipeline to ensure your set up is valid.
2. Click **Start deployment** to move your newly updated application through the pipeline.**Note:** This active deployment promotes changes from development to higher environments, as defined in the pipeline.

You are now ready to learn more about customizing Deployment Manager pipelines.

Quick-start guide for Deployment Manager on-premises and client-managed cloud

This guide summarizes all of the required steps for getting started with Deployment Manager for on premises and client cloud deployments. For customers with familiarity with Deployment Manager and the Pega Platform it can act as a checklist for quickly setting up a new Deployment Manager implementation.

Note: Determine two passwords that will be used throughout these instructions for the following operators:

- *DMReleaseAdmin*
- *DMAppAmin*

Import the Deployment Manager applications

On the orchestration environment

1. Import the PegaDevOpsFoundation RAP.
2. Import the PegaDeploymentManager RAP.
3. Restart the orchestrator after imports are completed.

On each candidate environment

1. Import the PegaDevOpsFoundation RAP.
    - PegaDevOpsFoundation 4.8.4 for Pega Platform 8.1 - 8.4
    - PegaDevOpsFoundation 5.1 for Pega Platform 8.5.2 or higher.

See Installing or upgrading Deployment Manager 5.x for a more detailed installation guide.

Configure the orchestration environment

1. Log in to the orchestration environment as an administrator.
2. Enable the DMReleaseAdmin operator and set the password.
3. Open the DMAppAdmin authentication profile and set the password.
4. Log in as DMReleaseAdmin.
5. Navigate to **Settings General settings** .
    1. Ensure that the orchestrator URL is correct.
    2. Click **Generate client secret**.
    3. Download the client secret.
    4. Click **Update authentication profiles**.
6. Upload the JKS keystore on DMKeyStore rule. For information on keystore and truststore setup, see Securely authenticating in Deployment Manager.

See Setting up the orchestrator for more information on configuring the orchestrator.

Configure the candidate environments

1. Log in to each candidate environment as an administrator.
2. Enable the DMAppAdmin operator and set the password.
3. Open the DMReleaseAdmin_oAuth2 authentication profile and set the client secret, access token endpoint, and revoke token endpoint downloaded from the orchestrator.
4. Upload the JKS keystore on DMKeystore. For information on keystore and truststore setup, see Securely authenticating in Deployment Manager.
5. Update the alias and password for the keystore on DeploymentManagerClientJWTProfile.

See Setting up candidate environments for more information on configuring the candidate environments.

Configure the development environment

1. Log in to the dt1 environment as an administrator.
2. Update the *OrchestratorURL* Dynamic System Setting to the URL of the DevOps environment. The URL should end in */prweb*.
3. Open the application record with which you want to manage with a Deployment Manager pipeline, and set **PegaDevOpsFoundation version 5** as a built-on application

See Setting up candidate environments for more information on configuring the candidate environments.

Configure the application

1. The application that is being promoted by a Deployment Manager pipeline should be built-on the latest version of the PegaDevOpsFoundation application.

See Configuring an application for more information on configuring the candidate environments.

Create your first pipelines

1. Log in to the orchestration environment as DMReleaseAdmin.
2. Click **New** in the top-right corner and choose **Deployment pipeline**.
3. On the first screen, enter the following information as it applies to your application:
    1. **Application name** and **version**
    2. **Access group**
    3. **Product name** and **version** that defines your entire application.
    4. **Pipeline name**
4. Click **Next** to progress to the Environment details screen and enter the following information:
    1. **Environment URL** and **Authentication profile** for each environment you want included in the pipeline.
    2. **Development repository**
    3. **Production repository** (can be the same as development)
5. Click **Next** to progress to the Model pipeline screen and press **Create** to create the pipeline.

See Creating pipelines for more information on creating pipelines.

Test and Execute the pipeline

1. In the top right corner of the rule form, in the action menu, diagnose the pipeline to ensure your set up is valid.
2. Click **Start deployment** to move your newly updated application through the pipeline.**Note:** This active deployment promotes changes from development to higher environments, as defined in the pipeline.

You are now ready to learn more about customizing Deployment Manager pipelines.

Setting up repositories

Deployment Manager stores application artifacts created throughout the lifecycle of a deployment within a repository. Using Deployment Manager requires the usage of a supported artifact repository.

Note: Pega Cloud services issues clients with a default file storage repository with their instance of Deployment Manager. This article can be skipped for Pega Cloud clients, unless a custom artifact repository is desired.

Repository types

Pega Platform supports a variety of repository vendors. Deployment Manager packages and promotes the application changes that you configure in a product rule for each pipeline deployment. Deployment Manager generates the application package on the development environment, publishes to the repository, and deploys to each subsequent stage in the pipeline.

Note: Production-ready artifacts do not require their own repository. When a pipeline marks an artifact as production-ready, it copies the artifact from the development repository location to the production repository.

Deployment Manager supports the following repository vendors:

- Microsoft Azure
- JFrog Artifactory
- Amazon S3
- Nexus 2/3
- File system

Deployment Manager does not support the following repository vendors:

- Pega
- defaultstore (repository instance included in each Pega deployment)

Pega provides the option to implement a custom repository type if required. For more information, see Creating and using custom repository types for Deployment Manager.

Prerequisites

- You have one of the listed supported repositories available
- Candidate and orchestrator environments can access the repositories

Steps

1. Log in to each candidate and orchestrator environment and create a repository record that connects to your desired artifact repository.
    - Ensure the same name is used for the repository on each environment.
    - For some repository types, such as JFrog and Nexus, you must select the Preemptive authentication check box on the Authentication Profile.
2. Test connection to ensure that the current system can connect to the target repository.

Optional Steps

- If you intend to use a separate repository for Production artifacts, create a second record pointed to the production repository on the development, production and orchestrator environments.

What to do next:

Previous: Installing or upgrading Deployment Manager 5.x	Next: Setting up the orchestrator

Setting up the orchestrator

The orchestrator is a standalone Pega environment that allows modeling and execution of continuous integration and continuous delivery (CI/CD) pipelines.

Before you begin: Both the **PegaDevOpsFoundation** and **PegaDeploymentManager** applications should be installed. See below for steps on setting up and configuring the orchestrator.**Note:** Pega Cloud DevOps environments already include both **PegaDevOpsFoundation** and **PegaDeploymentManager**.

1. Log in to the orchestrator environment as an administrator and enable the DMReleaseAdmin operator ID and specify its password.
    a. Log in to the orchestration server as an administrator.
    b. In the header of Dev Studio click **Records Organization Operator ID** , and then click **DMReleaseAdmin**.

   c. On the **Edit Operator ID** rule form, click the **Security** tab.
   d. Clear the **Disable Operator** check box.
   e. Click **Save**.
   f. Click **Update password**.
   g. In the **Change Operator ID Password** dialog box, enter a password, reenter it to confirm it, and then click **Submit**.

   This authentication profile is used to connect to candidate environments and execute tasks. Note the password for future use when setting up candidates.
2. From the Deployment Manager portal, navigate to **Settings General Settings** .
3. In the **Set orchestrator URL** field, update the orchestrator URL to establish connectivity for services.
4. Click **Save**.
5. Enable the DMAgentUser operator ID and specify its password.
   a. Log in to the orchestration server as an administrator.
   b. From the Dev Studio header, click **Records Organization Operator ID** , and then click **DMAgentUser**.
   c. On the **Edit Operator ID** rule form, click the **Security** tab.
   d. Clear the **Disable Operator** check box.
   e. Click **Save**.
   f. Click **Update password**.
   g. In the **Change Operator ID Password** dialog box, enter a password, reenter it to confirm it, and then click **Submit**.
   h. Log out of the orchestration server.
6. Modify the DMAppAdmin authentication profile to use the new password. The orchestration server uses this authentication profile to communicate with candidate systems so that it can run tasks in the pipeline.
   a. Log in to the orchestration server with the DMAppAdmin user name and the password that you specified.
   b. From the Dev Studio header, click **Records Security Authentication Profile** .
   c. Click **DMAppAdmin**.
   d. On the **Edit Authentication Profile** rule form, click **Set password**.
   e. In the **Password** dialog box, enter the password, and then click **Submit**.
   f. Save the rule form.

Generate client secret on the orchestrator

The client secret is a dynamic hexadecimal string that establishes communication between Deployment Manager services. You should not share this information publicly.

To generate the client secret from within Deployment Manager:

1. Navigate to **Settings General settings Generate client secret** .
2. Click on **Generate client secret**.
3. Download the client secret by clicking **Download client secret**.
   a. It is important to download the client secret as you must update this information in the authentication profile of candidate environments. Once you navigate away from this page or after you update the authentication profiles on the orchestrator, you must regenerate the client secret.

   Note: It is important to download the client secret as you must update this information in the authentication profile of candidate environments. Once you navigate away from this page or after you update the authentication profiles on the orchestrator, you must regenerate the client secret.

4. Click **Update authentication profiles** to automatically populate the **Client secret**, **Access token endpoint**, and **Revoke token endpoint** on the orchestrator system. If the **Update authentication profile** button does not work, you must perform the following manual steps on the orchestrator:
   a. Navigate to **Operator Switch to Dev Studio** .
   b. Open **DeploymentManagerClient** client registration by selecting **Records Security OAuth 2.0 Client Registration** , and click **Regenerate client secret** to regenerate the client secret.
   c. Click the **View and download** button to view the client secret and download the client secret information.
   d. Save the **DeploymentManagerClient** rule.
   e. Update the **Client secret**, **Access token endpoint**, and **Revoke token endpoint** with the information for the **DMStudioUser** and **DMAgentUser** authentication profiles.

   For additional configuration options, see the following optional steps:
   - Create additional operators by giving them the PegaDeploymentManager:Administrators access group. For more information about user roles and privileges, see .
   - To receive email notifications for deployments, configure email accounts on the orchestration server. For more information, see Configuring email accounts on the orchestration server.
   - Configure Jenkins so that it can communicate with the orchestration server. For more information, see Configuring Jenkins in 5.1.x.
   - Configure a separate authentication profile for each environment if you do not wish to use a single authentication profile.

What to do next:

Previous: Setting up repositories		Next: Setting up candidate environments

Setting up candidate environments

Candidate environments are any Pega environment that a Deployment Manager Pipeline manages. Most pipelines consist of Dev, QA, Staging, and Production environments.

Before you begin: Install the **PegaDevOpsFoundation** application on each candidate environment. See below for setting up a candidate environment.**Note:** If you did not enable SSL on the candidate environment, then you must deselect the "Require TLS/SSL for REST services in this package" for both the *cicd* and *api* service packages. Pega does not recommend this configuration.

1. On each candidate system, enable the DMAppAdmin operator ID.
   If you want to create your own operator IDs, ensure that they point to the PegaDevOpsFoundation application.
   a. Log in to each candidate system as an administrator.
   b. From the Dev Studio header, click **Records Organization Operator ID** , and then click **DMAppAdmin**.
   c. On the **Edit Operator ID** rule form, click the **Security** tab.
   d. Clear the **Disable Operator** check box.
   e. Click **Save**.
   f. Click **Update password**.
   g. In the **Change Operator ID Password** dialog box, enter a password, reenter it to confirm it, and then click **Submit**.
   h. Log out of each candidate system.
2. For development environments, update the OrchestratorURL Dynamic System Setting in the *PegaDevopsShared* ruleset to point to the orchestrator. Use this setting for Dev Studio and App Studio integration. The URL should end in /prweb (though this is customizable).
3. Create and configure a keystore named *DMKeyStore*.
4. If your target environment is SSL-enabled with private certificates, configure the Deployment Manager connectors so that they can receive and process information by setting the keystore:
   a. Click **Switch to Dev Studio Record explorer** to create and configure a keystore. For more information, see Creating a keystore for application data encryption
   b. Configure the *PegaDeploymentManagerIntegrationsTrustStore* dynamic system setting to reference the keystore ID by clicking **Records SysAdmin Dynamic System Settings** .

   For more information about dynamic system settings, see Creating a dynamic system setting.

5. If the candidate system is between Pega Platform 8.1 and Pega Platform 8.5.1, the candidate must have 4.8.4 Pega DevOps Foundation. If candidates are managed by an orchestrator on version 5 or later, you must create the PegaDevopsShared configuration and set the value to True. If not set, the candidate will fall back to using the older 4.x APIs for interactions with Deployment Manager and the pipelines will not be functional if using a 5.x orchestrator. Having the configuration created and set to true will ensure the candidates would leverage the 5.x API service.
   - Owning ruleset: PegaDevopsShared
   - Purpose: deploymentmanager/orchestrator/managed_by_5x/enabled
   - Value: True

Setting client secret on the candidate environment

Deployment Manager cannot automatically populate the client secret to candidate environments as we do not recommend that you share this information across systems.

To manually update the client secret information (from Step 2 above) on your candidate environment:

1. In Dev Studio, from the Records Explorer, navigate to **Security Authentication Profile** to receive a list of profile names on the candidate environment.
2. Select the **DMReleaseAdmin_OAuth2** authentication profile.
3. Update client secret on the authentication profile, and follow the steps below as applicable:
   a. If your candidate system is on Pega Platform 8.3 or above, on the OAuth 2.0 tab under the **Client configuration** section, enter the client secret in the **Client secret** field.
   b. If your candidate system is on Pega Platform 8.2 or below, update the client secret in the **DMReleaseAdmin_OAuth2** authentication profile. Update **Access token endpoint** and **Revoke token endpoint** in **DMCustom O Auth 2.0 Provider**.
4. Under the **Endpoint configuration** section, enter the **Access token endpoint** and **Revoke token endpoint**.
5. Click **Save**.

What to do next:

Previous: Setting up the orchestrator		Next: Configuring an application

Securely authenticating in Deployment Manager

Deployment Manager 5.1 now supports an OAuth 2.0 token-based authentication process for a more secure operator experience within the orchestrator and candidate environments. This authentication and authorization model provides many benefits, including the ability to audit user operations within Deployment Manager, as all actions are now connected

to an operator ID instead of a generic authentication profile, such as DMReleaseAdmin.

Keystore and truststore setup

Enabling encryption between nodes secures the data that is transferred across nodes so that an unauthorized host cannot access the data. Create a keystore.jks for the private key and the associated certificate or certificate chain. Ensure that you have your keystore.jks and truststore.jks files readily available for upload before beginning this step.

Ensure that you save the alias and passwords used to create the JKS files as they are used to setup Deployment Manager. For more information on creating these files, see Creating the keystore.jks and truststore.jks files.

Deployment Manager establishes secure token-based communication with the Deployment Manager Service APIs. The key store and trust store configuration is setup to ensure the portal functions correctly.

On the orchestrator:

1. In the navigation pane of Dev Studio, click **Records Security Keystore** .
2. Open the *DMKeyStore* rule to upload the keystore and update the keystore password.
3. Click **Save**.
4. To enable communication from Deployment Manager:
    1. From Dev studio, open **Token Profile** from the Records menu and **Security** sub menu.
    2. Open the *DeploymentManagerClientJWTProfile* token profile rule.
    3. Under the Security section, ensure that the keystore refers to *DMKeyStore*.
    4. Update the alias that you defined when creating the keystore, and update the password to the password that you set when setting up the truststore.
    5. Click **Save**.

On candidate environments

1. To establish communication between non-trusted systems:
    - Create and configure the *DMKeyStore* as you did on the orchestrator.
    - Update *PegaDeploymentManagerIntegrations* TrustStore dynamic system setting to *DMKeyStore*.

Troubleshooting

1. What should I do if deployments are stuck **INPROGRESS** with a java.lang.IllegalArgumentException: Empty key error in the logs?
    - This is a known issue in Pega Platform 8.5.2 related to OAuth 2.0. As a workaround, perform the following steps.**Note:** This has been resolved in Pega Platform 8.5.2 as part of Hotfix-69607.
        1. Log in to Deployment Manager with the SUPERADMIN role.
        2. From the navigation pane, click **Switch to Dev Studio Records explorer Security OAuth 2.0 Client Registration** .
        3. Open **DeploymentManagerClient** client registration.
        4. Click **Revoke access and refresh token**.

Previous: Setting up candidate environments		Next: Configuring an application

Configuring an application

There are a few requirements and recommendations for structuring an application that is being managed using Deployment Manager. See this brief overview to get started with configuration.

Add PegaDevOpsFoundation as a built-on application

The PegaDevOpsFoundation application provides the capabilities to execute pipeline tasks and adds pipeline integration to the Dev Studio and App Studio. Any application that is being managed by Deployment Manager must include it as a built-on application.

Note: Use the major version of the application so patches and minor versions can be taken without updating the application. For example, include PegaDevOpsFoundation:5 as a built-on instead of PegaDevOpsFoundation:5.1.1.

For new applications

New applications only need to be updated on the development environment. Deployment Manager is capable of migrating the whole application to new environments as part of deployment pipelines.

For applications that have already been deployed to all environments

If the application has already been deployed to all environments, it is required to manually update the application record

on each environment to be built on *PegaDevOpsFoundation*. The next deployment will sync up the application changes and will result in an aged update message on Deployment Manager.

Create a development layer

The target application is regularly packaged and promoted to higher environments so it is important to exclude branches and any other work in progress. A development layer consists of:

- An application that is built on the testing, or target application. Consider following the naming convention: *{TargetApp}Dev*.
- An unlocked ruleset for any developer tools, or work that is never intended to be promoted to higher environments.
- An access group that logs into the development application.

Development layers can be assigned to developers as needed, whether it is one per application, team or developer.

Using a development layer also allows you to keep the target application locked to prevent unintentional changes.

Create a test layer

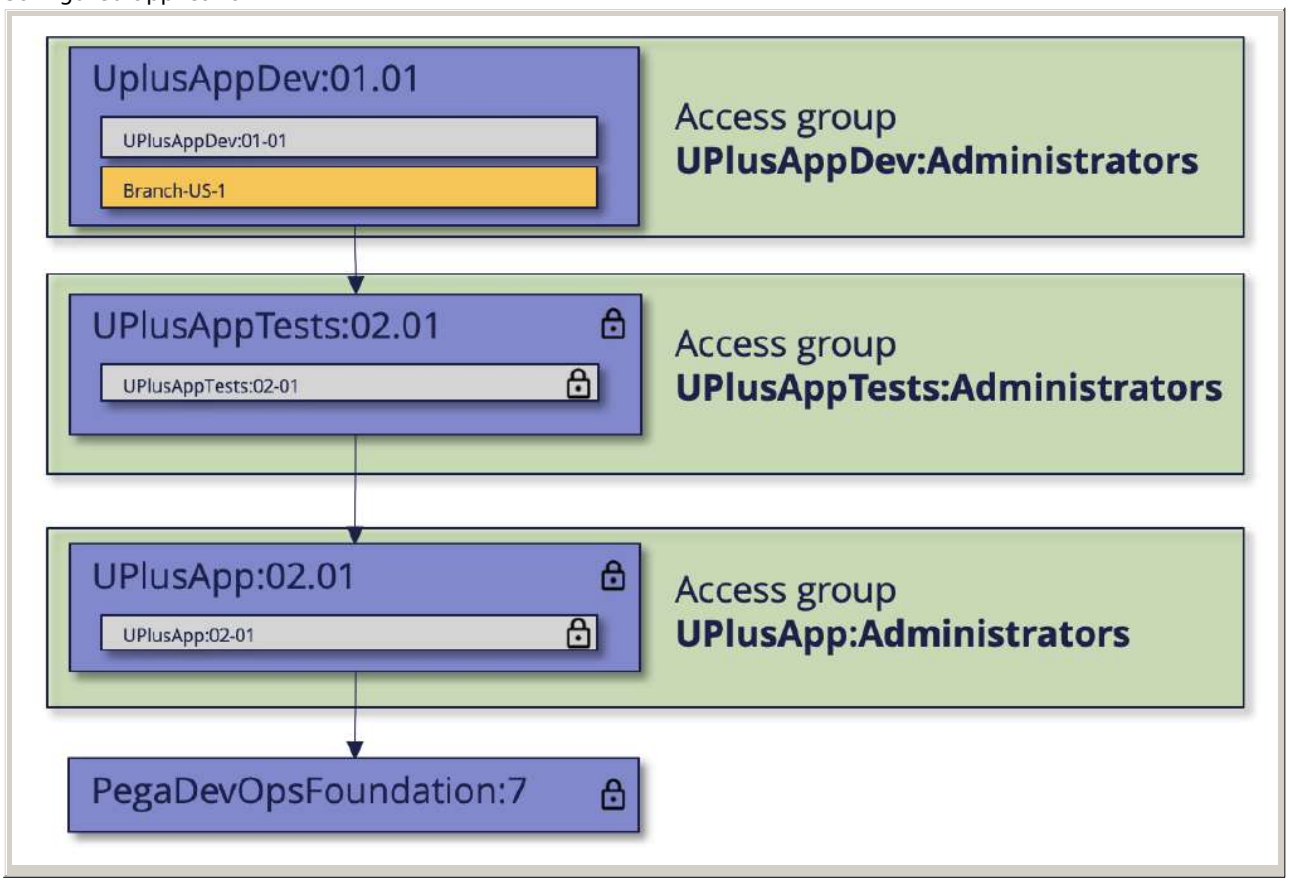
Testing assets are managed by Deployment Manager pipelines, but they are only deployed to a subset of environments. To manage and run test cases it is necessary to create a test layer. A test layer consists of:

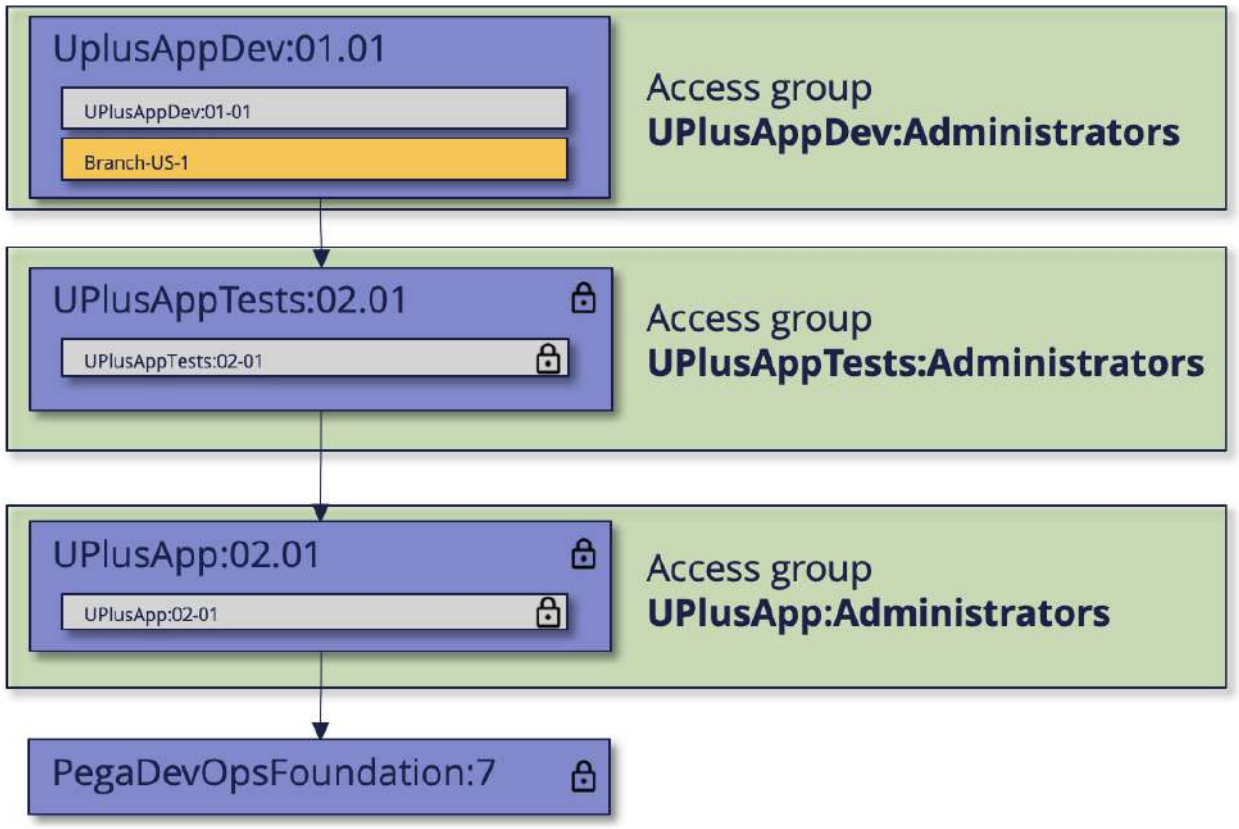
- A locked **application** that is built on the target application. Consider following the naming convention *{TargetApp}Tests*.
- A ruleset for all test cases and test assets.
- An access group that logs into the test application.
- A product rule that defines which assets get promoted by the pipeline.

Summary

If you have followed all the recommendations in this article, you should end up with an application that is structured like the one seen below.

Configured application





Configuring the development system for branch-based development in 5.1.x

If you are using branches in either a distributed or non-distributed branch-based environment, configure the development system so that you can start deployments when branches are merged. Configuring the development system includes defining the URL of the orchestration server, creating development and target applications, and locking application rulesets.

1. On the development system (in nondistributed environment) or the main development system (in a distributed environment), create a dynamic system setting to define the URL of the orchestration server, even if the orchestration server and the development system are the same system.
   a. Click **Create Records SysAdmin Dynamic System Settings** .
   b. In the **Owning Ruleset** field, enter PegaDevOpsShared.
   c. In the **Setting Purpose** field, enter OrchestratorURL.
   d. Click **Create and open**.
   e. On the **Settings** tab, in the **Value** field, enter the URL of the orchestration server in the format http://*hostname:port*/prweb/.
   f. Click **Save**.

   For more information about dynamic system settings, see Creating a dynamic system setting

2. Complete the following steps on either the development system (in a non-distributed environment) or the remote development system (in a distributed environment).
   a. Use the New Application wizard to create a new development application that developers will log in to. This application allows development teams to maintain a list of development branches without modifying the definition of the target application.
   b. Add the target application of the pipeline as a built-on application layer of the development application by first logging into the application.
   c. In the header of Dev Studio, click the name of your application, and then click **Definition**.
   d. In the **Built-on application** section, click **Add application**.
   e. In the **Name** field, press the Down arrow key and select the name of the target application.
   f. In the **Version** field, press the Down arrow key and select the target application version.
   g. Click **Save**.
3. Lock the application rulesets to prevent developers from making changes to rules after branches have been merged.
   a. In the header of Dev Studio, click the name of your application, and then click **Definition**.
   b. In the **Application rulesets** section, click the **Open icon** for each ruleset that you want to lock.
   c. Click **Lock and Save**.
4. Copy the development repository that you configured on the remote development system to the source development system.

5. If you are managing test cases separately from the target application, create a test application. For more information, see Managing test cases separately in Deployment Manager.
6. **Optional:** To rebase your development application to obtain the most recently committed rulesets after you merge your branches, configure Pega Platform so that you can use rule rebasing.
   For more information, see Understanding rule rebasing.

What to do next:

Previous:Securely authenticating in Deployment Manager	Next:Migrating data to Deployment Manager 5.x

Migrating data to Deployment Manager 5.x

Use the migration assistant feature to safely migrate data, including roles, operator records, and pipelines to the version 5.x model.

Note: Only users with the Super Admin role can perform pipeline, operator and roles migration.

Pipeline migration

1. Import the Deployment Manager 5.x RAPs and restart the server.
2. Log in to Deployment Manager using the *DMReleaseAdmin* operator.
3. In the lower-left corner of the application, click the **Operator** icon, and then select **Switch to Dev Studio**.
4. Open the operator record, and then update the default access group to **SuperAdmin**.
5. Log out of the application, and then log in as the *DMReleaseAdmin* operator.
6. On the Deployment Manager banner, click the **info** icon.
7. Click **Migration assistant**.
8. In the **Pipeline Migration** section, select the pipelines that you want to migrate.

   Note: Migrate dependent pipelines first. **View dependencies** link provides details of all pipeline dependencies.

9. Click **Migrate**.
10. Confirm the pipeline migration by clicking **Submit**.
11. When the migration is complete, the pipelines indicate the status of their migration:

    Success
    : Indicates that the migration of the pipeline is successful. Expanding the row provides additional details on the pipeline migration.

    Fail
    : Indicates that the creation of pipeline failed. Click **View log** to understand the cause of the failure.

    Not migrated
    : Indicates that the pipeline is not migrated and the user can start the migration as required.

12. If pipeline migration fails due to a loading error, ensure that the open ruleset in which pipeline case types are stored is present in the Deployment Manager 5.x application stack.

Note:

- Stop all active deployments. Any deployments in progress are automatically aborted during the upgrade.

Migrating the operator

After a successful upgrade to the 5.x version, you can upgrade operators from the **Migration assistant** page. Operator migration migrates the existing roles to associate them with selected operators.

1. On the Deployment Manager banner, click the **info** icon.
2. Click **Migration assistant**.
3. In the **Operator Migration** section, select the operators that you want to migrate.
4. Click **Migrate**.
5. Confirm the operator migration by clicking **Submit**.
6. The operators table provides the status of operator migration:

   Success
   : Indicates that the selected operator migration succeeded.

   Fail
   : Indicates that either role or operator migration failed. For more information, you can click **View upgrade logs** on the **Migration assistant** page.

   Not migrated
   : Indicates that the operator is not migrated and the user can start the migration as required.

     Note: Operators that you migrate manually to version 5.x are not tracked by the operator migration feature.

Updating access group

If you are upgrading to version 5.x, ensure the *PegaDevOpsFoundation:Administrators* access group is updated to point to version 5 of the PegaDevOpsFoundation application. Perform this update for all candidates that are upgraded to PegaDevOpsFoundation 5.x.

Rolling back to Deployment Manager 4.x from 5.x

After a successful upgrade to the 5.x version, you can rollback your migration to a 4.x version if you choose.

1. Ensure the candidate application points to a PDF 4 application and the *PegaDevOpsFoundation:Administrators* access group is pointed back to application PegaDevOpsFoundation 4.
2. Change *DMReleaseAdmin AccessGroup* to *PegaDeploymentManager:Administrator*. This points your PegaDeploymentManager application to 4.x.
3. If pipelines have already been migrated:
    - Log in to Deployment Manager 4.x portal. Go to the archived landing page, and manually activate the required pipelines.
    - Run the *DeleteCreatedPipelines* activity to clean up all pipelines created as part of the DM 5.x migration. This will ensure a smooth migration should you choose to return to 5.x.
4. If operators were migrated to 5.x, you can roll them back using the *UpdateDMOperatorsTo4x activity* with parameter *ALL*.

Frequently asked questions (FAQ)

1. Will custom tasks automatically migrate to Deployment Manager 5.x?
    - Custom tasks are not currently supported in Deployment Manager 5.1. Custom tasks are removed from the pipeline during migration, and the pipeline is disabled by default in Deployment Manager 5.1.x.
2. Why are pipelines disabled after migration to Deployment Manager 5.x?
    - There are a few possible reasons that pipelines may be disabled after migration.
        - If the original pipeline was disabled, it remains disabled after being migrated.
        - Pipeline included custom tasks. Custom tasks are not supported in Deployment Manager 5.x.
3. Why are pipelines archived after migration to Deployment Manager 5.x?
    - To ensure no deployments are triggered in Deployment Manager 4.x after migration to 5.x. Therefore:
        - If the original pipeline was in an archived state, it remains archived post-migration.
        - If the original pipeline was in an active state, it becomes archived post-migration.
4. What happens to original pipelines after migration?
    - Deployment Manager aborts any active deployments and archives the pipeline before migration.
5. Why does Deployment Manager create two new pipelines after the 5.x migration if merge is enabled in the original pipeline?
    - Deployment Manager 5.x and later support template-based pipeline creation to define a structured release process. The template-based model splits merge pipelines and deployment pipelines into two separate entities, differing from the combined model in previous releases. If you are migrating a pipeline with merge enabled, this is split into two pipelines to align with the new template-based model.
6. Is deployment history migrated to Deployment Manager 5.x?
    - Deployment history is not migrated. Only the latest and dependent artifacts are migrated. This data is still accessible by switching the Deployment Manager portal to the corresponding version of historical data.
7. Why is my pipeline migration is failing with a **Failed to load pipeline from DM 4.x** error?
    - This error can occur if pipeline case types are not present in the *PipelineData* ruleset. Include the custom ruleset and retry migration.
8. When I try to save a newly created merge pipeline from the 5.1 migration, the save fails with an error stating a task could not be found.
    - Pipelines created from the migration process do not enforce environment templates. However, Deployment Manager portal enforces the recommended and mandatory tasks on an environment template selected for the stage. Hence, if any edits are required on the pipeline after the merge, you must abide to the environment template tasks. For more information, see Pipeline templates.

Configuring pipelines and deploying applications in Deployment Manager

Use Deployment Manager to create continuous integration and delivery (CI/CD) pipelines, which automate tasks so that you can quickly deploy high-quality software to production.

On the orchestration server, release managers use the Deployment Manager landing page to configure CI/CD pipelines for their Pega Platform applications. The landing page displays all the running and queued application deployments, branches that are to be merged, and reports that provide information about your DevOps environment such as key performance indicators (KPIs).

- Navigating the Deployment Manager portal

  The Deployment Manager portal is where you can find all information about pipelines and deployments. This portal offers functionality to filter pipelines, view reports, and view deployment statuses.

- Understanding pipeline types

  Pipeline templates help you structure pipelines for their ultimate purpose, as well separate business needs for better organizational efficiency. Deployment Manager offers four different pipeline types, each with a different use case. See the articles below for more information on the different available pipeline templates and their use cases.

- Setting up users and roles

  Define roles and users to manage which users can access Deployment Manager and which features they can access. For example, you can create a role that does not permit users to delete pipelines for a specific application.

- Creating pipelines

  When you add a pipeline, you define all the stages and tasks that you want to do on each system. For example, if you are using branches, you can start a build when a branch is merged. If you are using a QA system, you can run test tasks to validate application data. Pipelines are the first step in getting started with setting up a process for deploying a change through the release. See the articles below for more information on creating each of the available pipeline types.

- Configuring an application pipeline in 5.1.x

  When you add a pipeline, you specify merge criteria and configure stages and steps in the continuous delivery workflow. For example, you can specify that a branch must be peer-reviewed before it can be merged, and you can specify that Pega unit tests must be run after a branch is merged and is in the QA stage of the pipeline.

- Interacting with application pipelines
- Configuring data migration pipelines

  Data migration tests provide you with significant insight into how the changes that you make to decision logic affect the results of your strategies. To ensure that your simulations are reliable enough to help you make important business decisions, you can deploy a sample of your production data to a dedicated data migration test environment.

- Interacting with data migration pipelines

  Create and run data migration pipelines in Deployment Manager 5.1.x to automatically export simulation data from a production environment into a simulation environment in which you can test simulation data. You can also use Deployment Manager to monitor and obtain information about your simulations, for example, by running diagnostics to ensure that your environment configurations are correct and by and viewing reports that display key performance indicators (KPIs).

- Performing ad hoc tasks

  By using ad hoc tasks, you can independently perform package and deploy actions outside of the context of a deployment, which is useful for exporting artifacts from the production environment and importing them into a lower environment, or for packaging and deploying artifacts if a Deployment Manager pipeline is not available.

- Understanding App Studio publishing

  App Studio publish enables Pega Platform users to publish and deploy application changes with no prior Deployment Manager experience necessary. This seamless functionality publishes minor application changes through deployment pipelines to create new artifacts without ever leaving App Studio.

- Troubleshooting issues with your pipeline

  Deployment Manager provides several features that help you troubleshoot and resolve issues with your pipeline.

Navigating the Deployment Manager portal

The Deployment Manager portal is where you can find all information about pipelines and deployments. This portal offers functionality to filter pipelines, view reports, and view deployment statuses.

- Logging in to Deployment Manager

  Deployment Manager provides a dedicated portal from which you can access features.

- Filtering pipelines in the dashboard

  The dashboard displays all active and disabled pipelines by default. Users can filter the pipelines by pipeline template or search by a pipeline name. To find an archived pipeline, select the Archived toggle option on the dashboard.

- Viewing deployment-level reports

  Deployment reports provide information about a specific deployment. You can view information such as the number of tasks that you configured on a deployment that have been completed and when each task started and ended. If there were schema changes on the deployment, the report displays the schema changes.

- Viewing pipeline-level reports

  Reports provide a variety of information about all the deployments in your pipeline. For example, you can view the frequency of new deployments to production.

Logging in to Deployment Manager

Deployment Manager provides a dedicated portal from which you can access features.

To log in to Deployment Manager, on the orchestration server, enter the DMReleaseAdmin operator ID and the password that you specified for it.

Filtering pipelines in the dashboard

The dashboard displays all active and disabled pipelines by default. Users can filter the pipelines by pipeline template or search by a pipeline name. To find an archived pipeline, select the **Archived** toggle option on the dashboard.

To filter the display of pipelines, perform the following steps:

1. Pipelines can be accessed from the landing page when you first log in. Alternatively, from the navigation pane of Deployment Manager click **Pipelines Application pipelines** .
2. At the top of the dashboard in the **search** box, enter the pipeline name to filter the available options. Alternatively, select pipeline types and template from the dropdown.

Viewing deployment-level reports

Deployment reports provide information about a specific deployment. You can view information such as the number of tasks that you configured on a deployment that have been completed and when each task started and ended. If there were schema changes on the deployment, the report displays the schema changes.

Do the following steps:

1. If the pipeline is not open, in the navigation pane of Deployment Manager, click **Pipelines** and then click the name of the pipeline.
2. Perform one of the following actions:
   - To view the report for the current deployment, click the **More** icon, and then click **View report**.
   - To view the report for a previous deployment, expand the **Deployment History** pane and click **View Reports** for the appropriate deployment.

Viewing pipeline-level reports

Reports provide a variety of information about all the deployments in your pipeline. For example, you can view the frequency of new deployments to production.

You can view the following key performance indicators (KPI):

- Deployment success factor – Percentage of deployments that are successfully deployed to production
- Pipeline completion frequency – Frequency of new deployments to production
- Deployment speed – Average time taken for a deployment to complete all stages of the pipeline
- Failure rate – Average number of task failures per deployment

1. Open the pipeline by doing one of the following actions:
   - If the pipeline open, click **Actions View report** .
   - If a pipeline is not open, in the navigation pane, click **Reports**. Next, in the **Pipeline** field, press the Down arrow key and select the name of the pipeline for which to view the report.
2. The pipeline report presents all deployments triggered in the last 30 days by default. If you want to expand or narrow the deployments to a certain time frame, select the dates from the date range filter.

Understanding pipeline types

Pipeline templates help you structure pipelines for their ultimate purpose, as well separate business needs for better organizational efficiency. Deployment Manager offers four different pipeline types, each with a different use case. See the articles below for more information on the different available pipeline templates and their use cases.

- Understanding deployment pipelines

  Deployment pipelines are an application pipeline type to generate artifacts from the system of record and deliver them through the various stages of the workflow. This pipeline template aligns with the continuous delivery (CI/CD) DevOps methodology, and deploys predictable, high-quality releases without using third-party tools.

- Understanding merge pipelines

Application developers make application changes and configurations that need to be integrated frequently with updates from other developers. These changes need to meet minimum standards of quality and compatibility ensuring that the changes made will not break existing functionality or cause a conflict with other changes. These users can leverage the merge pipeline template, built on the DevOps concept of Continuous Integration (CI).

- Understanding business change pipelines

  Business change pipelines are used to support everyday, business-as-usual changes to your Customer Decision Hub (CDH) application. This pipeline type allows users to respond to changing requirements by modifying and deploying application rules in a controlled manner.

- Understanding data migration pipelines

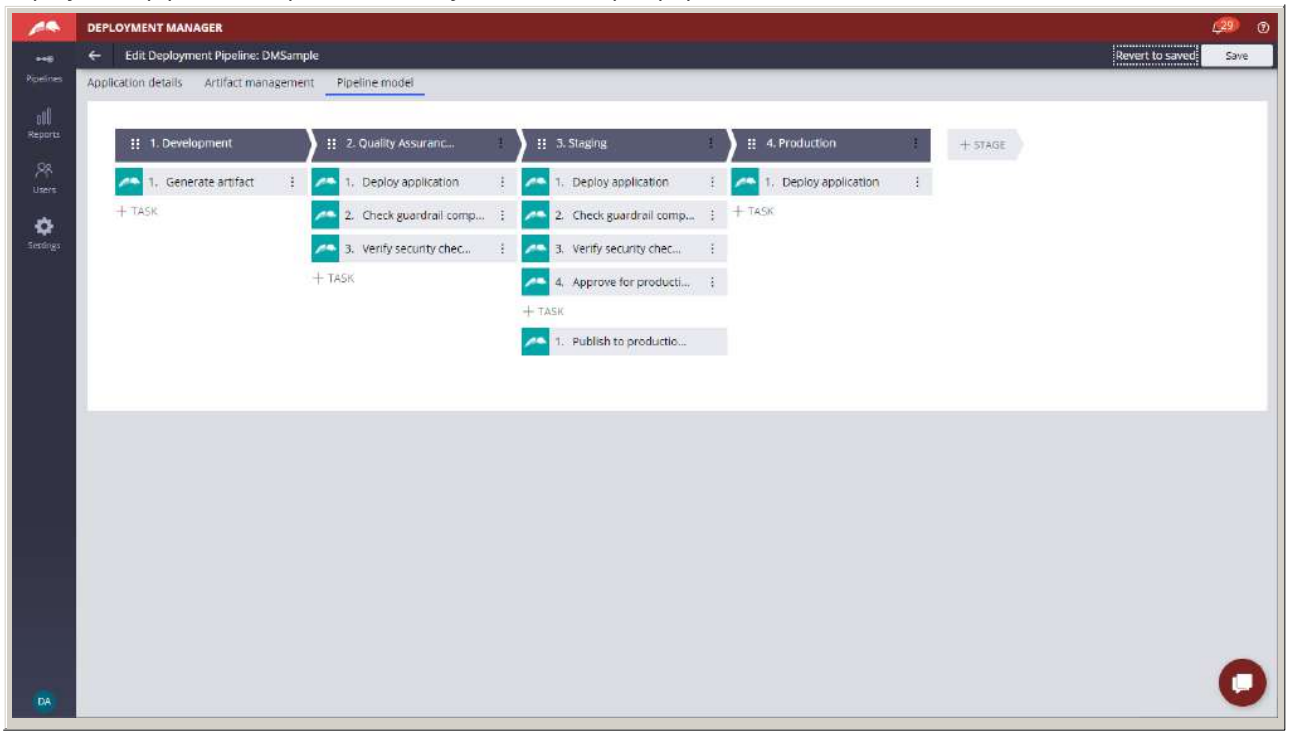
  Data migration pipelines are used to seek insight into how the changes that are made to decisioning logic affect the results of their strategies. To ensure that user simulations are reliable enough to help in making important business decisions, data migration pipelines can deploy a sample of the production data to a dedicated data migration test environment. This use case is powered by the data migration pipeline template. See for more information.

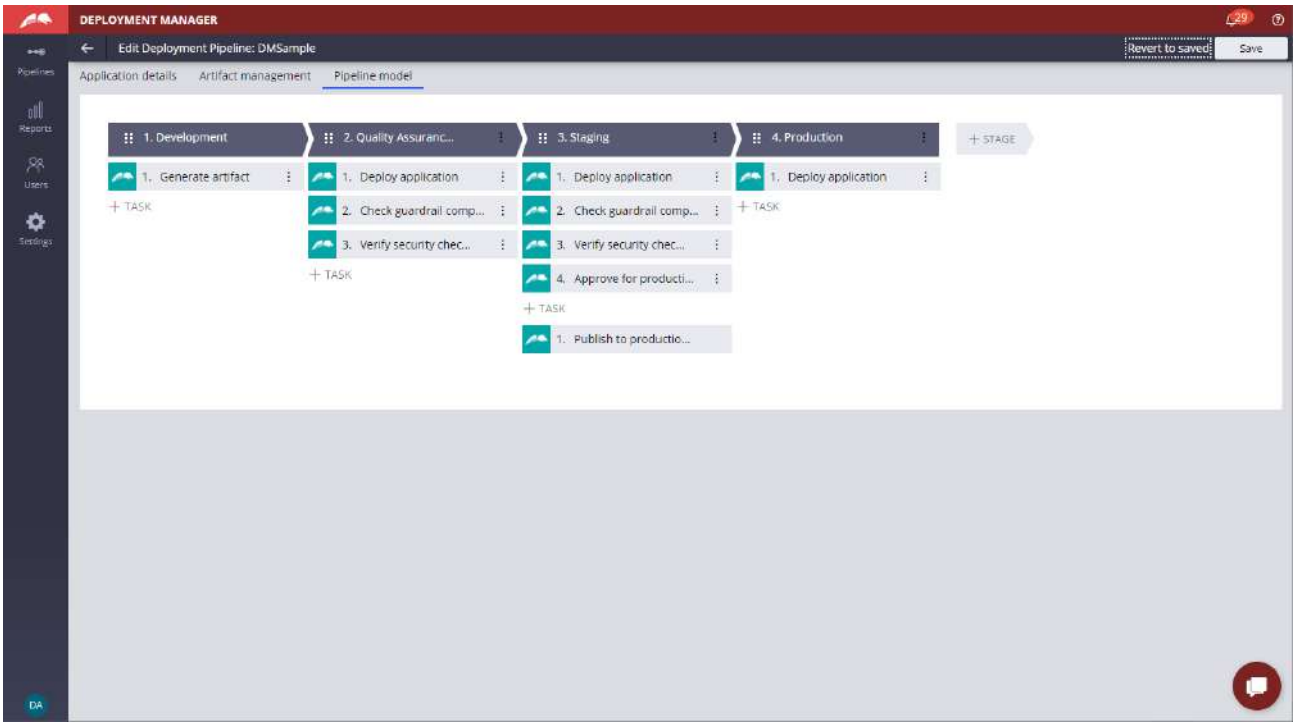
Understanding deployment pipelines

Deployment pipelines are an application pipeline type to generate artifacts from the system of record and deliver them through the various stages of the workflow. This pipeline template aligns with the continuous delivery (CI/CD) DevOps methodology, and deploys predictable, high-quality releases without using third-party tools.

Deployment pipelines are used to deploy applications that are released by development teams on various environments. The changes from the System of Record (SOR) (central or remote) are packaged into an artifact and then promoted to the testing environments, until completion in production environment(s). The deployment pipeline template is the template that offers a standard deployment model for promoting the artifacts through the stages of the pipeline. This template has built-in best practices that instantly creates a CD pipeline for an application. It provides immense possibilities to customize the process as guided by a governance model.

Deployment pipeline template loaded by default with pre-populated tasks





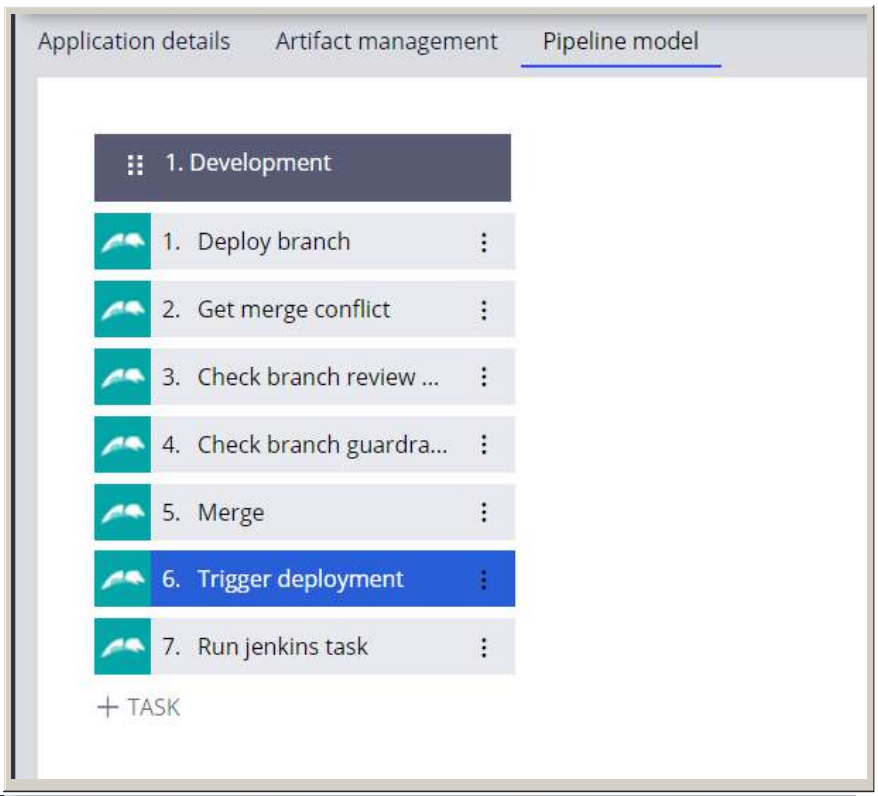
Deployment pipelines are an application pipeline type within Deployment Manager. To get started and create a deployment pipeline, see Creating a deployment pipeline.

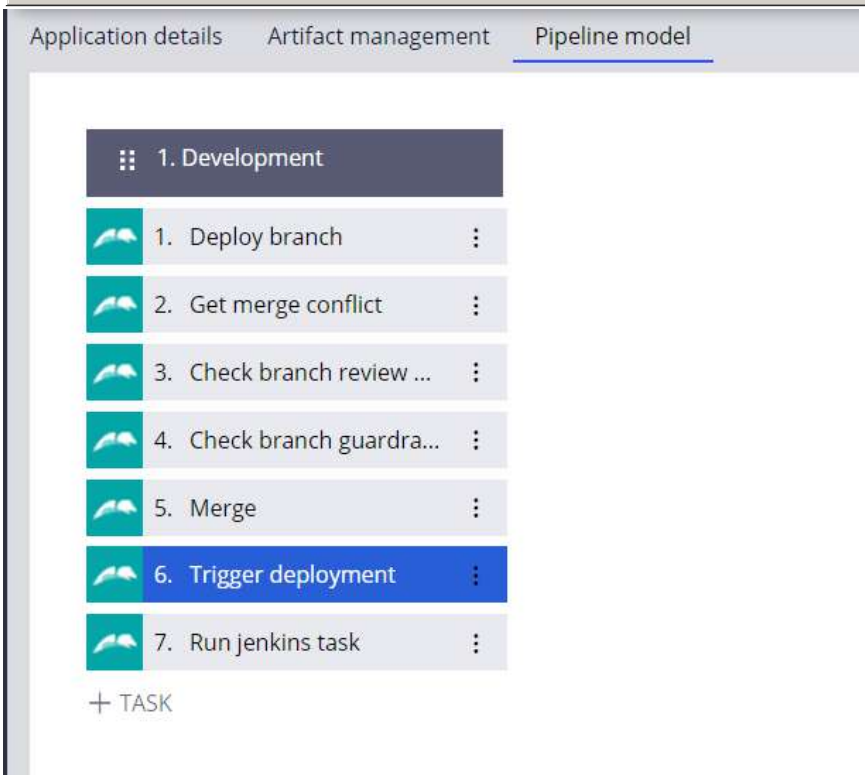
If you are new to Deployment Manager, see Installing or upgrading Deployment Manager 5.x to get started.

Understanding merge pipelines

Application developers make application changes and configurations that need to be integrated frequently with updates from other developers. These changes need to meet minimum standards of quality and compatibility ensuring that the changes made will not break existing functionality or cause a conflict with other changes. These users can leverage the merge pipeline template, built on the DevOps concept of Continuous Integration (CI).

Merge pipelines enable developers to submit rule branches for a merge after validating the changes against specific quality gates such as guardrails, branch review, and automated tests. If all merge criteria are successfully met, the branches merge and you can trigger a deployment to higher environments using the **Trigger deployment** task. If any of the quality gates fail, the merge request is rejected and the developer will be notified on the failure.





Merge pipelines are an application pipeline type within Deployment Manager. If you are looking to create a merge pipeline, see Creating a merge pipeline.

If you are new to Deployment Manager, see Installing or upgrading Deployment Manager 5.x to get started.

Understanding business change pipelines

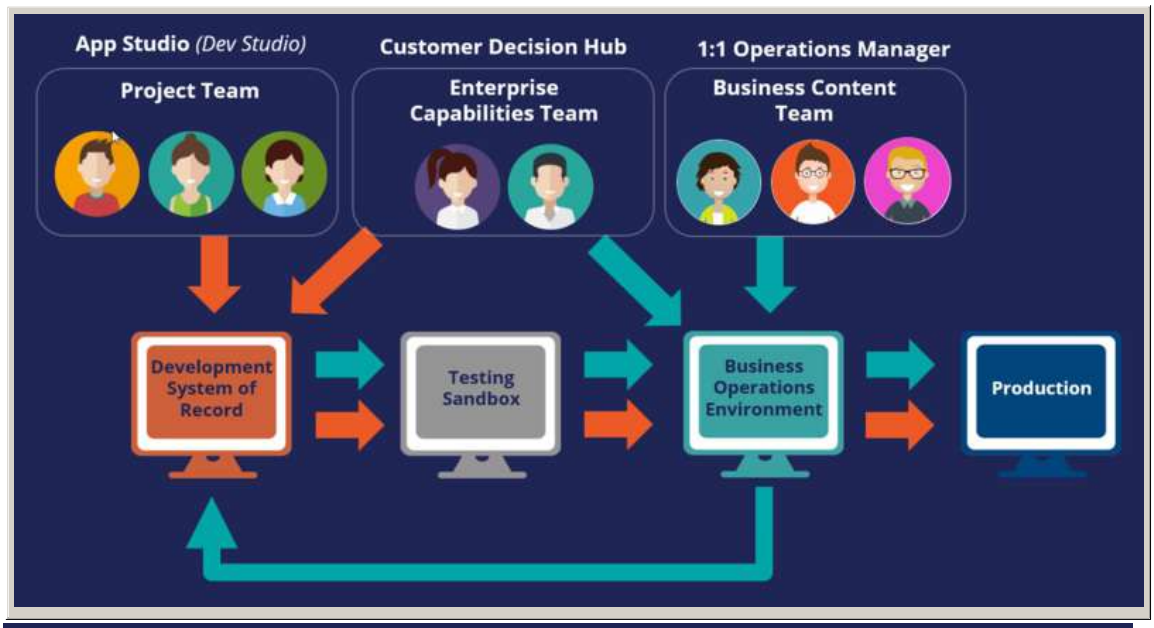
Business change pipelines are used to support everyday, business-as-usual changes to your Customer Decision Hub (CDH) application. This pipeline type allows users to respond to changing requirements by modifying and deploying application rules in a controlled manner.

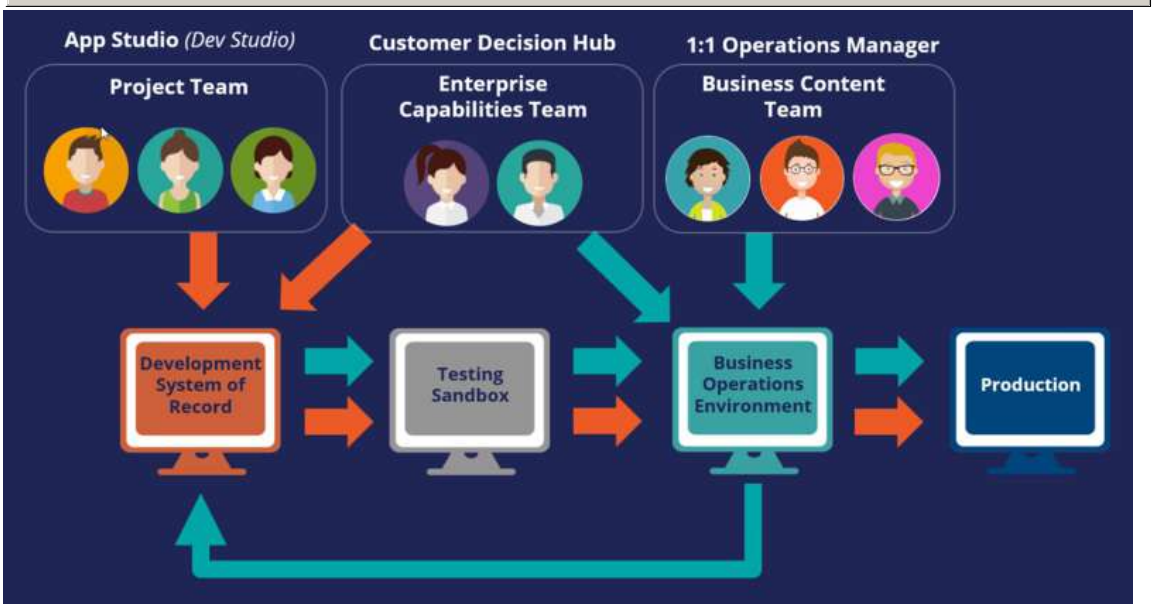
Business change pipelines offer CDH business users the means to deploy and test changes implemented to their applications outside the enterprise release cycles. This pipeline type enables a seamless experience from merging

changes to the development system of record, versioning the application and generating a release artifact. Deployment Manager then deploys the artifacts to each environment. Within the production environment, you are able to select a subset of users to test the modifications in a production environment and roll out the changes to all users when the test results are acceptable.

Note: The business change pipeline feature in version Deployment Manager 5.1 is supported with customers on Pega Platform 8.5.1 or later. Customers are not recommended to upgrade to Deployment Manager 4.8.4, as the suggested upgrade path from Deployment Manager 4.8.3 is to 5.1.

The business change cycle enables business users to respond quickly to changes in the external and internal factors that influence their business. Responses might include introducing new actions or propositions, imposing eligibility criteria, or modifying existing business strategies.





Business change pipelines are an application pipeline type within Deployment Manager. To get started and create a business change pipeline, see Creating a business change pipeline.

If you are new to Deployment Manager, see Installing or upgrading Deployment Manager 5.x to get started.

Understanding data migration pipelines

Data migration pipelines are used to seek insight into how the changes that are made to decisioning logic affect the results of their strategies. To ensure that user simulations are reliable enough to help in making important business decisions, data migration pipelines can deploy a sample of the production data to a dedicated data migration test environment. This use case is powered by the data migration pipeline template. See Configuring data migration pipelines for more information.

To get started and create a data migration pipeline, see Creating a data migration pipeline.

If you are new to Deployment Manager, see Installing or upgrading Deployment Manager 5.x to get started.

Setting up users and roles

Define roles and users to manage which users can access Deployment Manager and which features they can access. For example, you can create a role that does not permit users to delete pipelines for a specific application.

Deployment Manager provides two default roles, which you cannot modify or delete, that define privileges for super administrators and application administrators. Privileges for super administrators are applied across all applications, and privileges for application administrators are applied to specific applications. Super administrators can also add roles and specify the privileges to assign to them. Super administrators and application administrators can add users and assign them access to the applications that they manage.

- Adding and modifying roles

  If you are a super administrator, you can add and modify roles. Users within a role share defined responsibilities such as starting a deployment in a pipeline.

- Assigning privileges

  As of Deployment Manager 5.1.x, you can assign privileges to users to define or restrict access to the different features and functionality within the portal. See below for a list of privileges associated with each category of Deployment Manager.

- Providing access to Dev Studio to a role

  Deployment Manager provides a dedicated portal from which you can access features. From within Deployment Manager, when you configure pipeline details, you can open, modify, and create repositories and authentication profiles in Dev Studio if you have permissions to use the Dev Studio portal.

- Adding users and specifying their roles

  If you are a super administrator or application administrator, you can add users to Deployment Manager and specify their roles. Only super administrators can create other super administrators or application administrators who can access one or more applications. Application administrators can create other application administrators for the applications that they manage.

- Modifying user roles and privileges

  Super administrators can give other users super administrative privileges or assign them as application administrators to any application. Application administrators can assign other users as application administrators for the applications that they manage.

- Modifying your user details and password

  You can modify your own user details, such as first and last name, and you can change your password.

- Deleting users

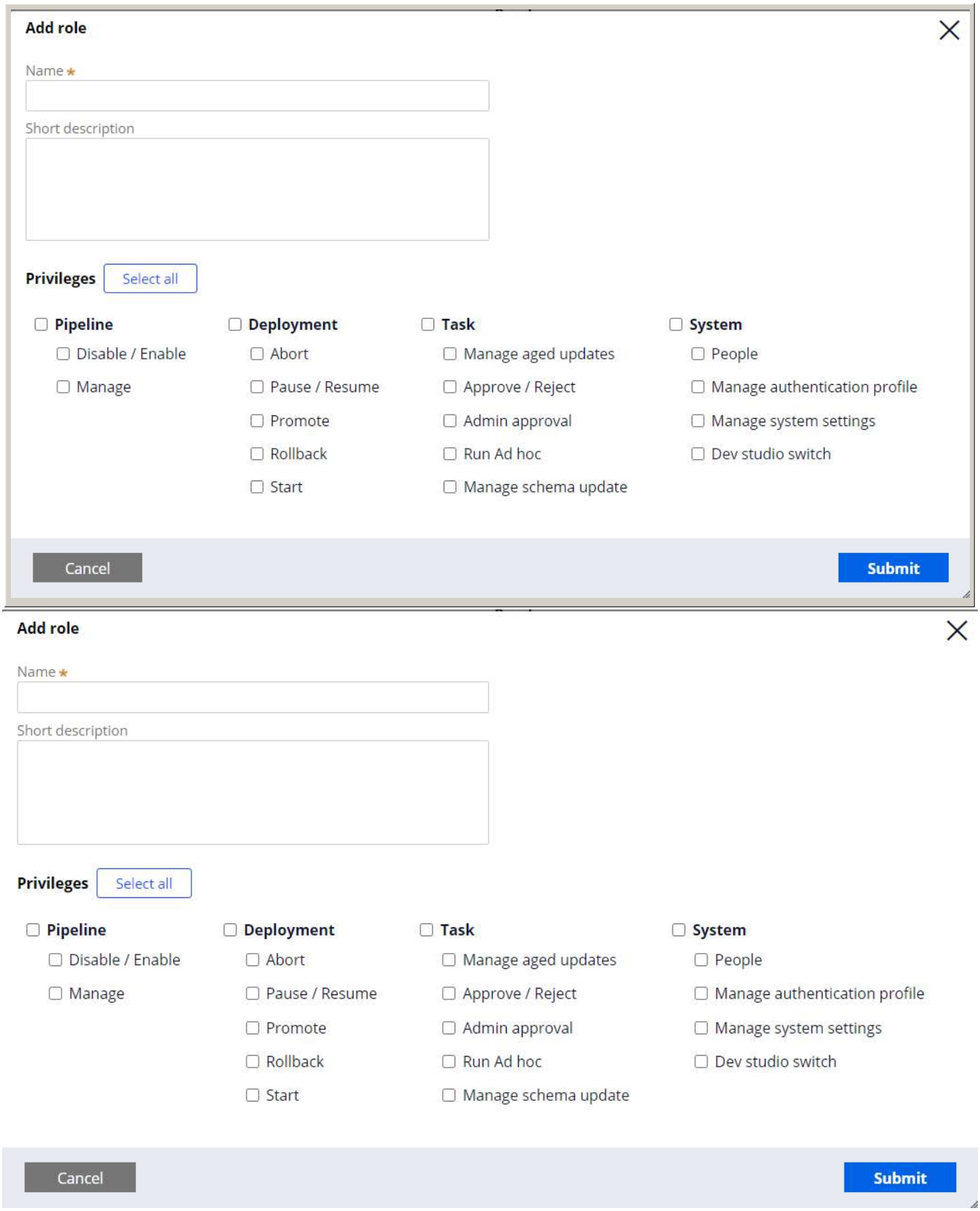
  If you are a super administrator or application administrator, you can delete users for the applications that you manage.

Adding and modifying roles

If you are a super administrator, you can add and modify roles. Users within a role share defined responsibilities such as starting a deployment in a pipeline.

If you are a super administrator, add or modify a role by doing the following steps:

1. In the navigation pane of Deployment Manager click **Users**, and then click **Roles and privileges**.
2. Do one of the following actions:
   - To add a role, click **Add role**.
   - To modify a role, click the gear icon for that role.
3. In the **Add role** dialog box, in the **Name** field, enter a name for the role and update the short description field with the purpose of that role. If you wish to edit the existing role, skip this step.
4. Select the privileges that you want to assign to the role.



5. Click **Submit**.

Assigning privileges

As of Deployment Manager 5.1.x, you can assign privileges to users to define or restrict access to the different features and functionality within the portal. See below for a list of privileges associated with each category of Deployment Manager.

Pipeline privileges

Pipeline privileges

Privilege	Description
Manage	Create a pipeline, edit pipeline settings and pipeline model, delete and archive an active pipeline. Additionally, can activate an archived pipeline for an authorized application.
Disable/ enable	Disable an active pipeline with active deployments. Enable a disabled pipeline.

Deployment privileges

Deployment privileges

Privilege	Description
Start	Start a deployment for a pipeline of an authorized application.
Abort	Abort an active or failed deployment for an authorized application
Rollback	Rollback a deployment for an authorized application.
Pause/resume	Pause an active deployment or resume a paused deployment of an authorized application.
Promote	Promote a deployment waiting for a manual transition to the next stage for any authorized application.

Task privileges

Task privileges

Privilege	Description
Approve/reject	Act on an approval\reject task in the case of a task being assigned to the user or user’s workgroup. Approve or reject a task in a deployment waiting for a user’s input.
Manage schema update	Act on schema updates encountered during deployment.
Managed aged update	Act on aged updates encountered during deployment.
Admin approval	Approve or rejet a task in case the assignee for the task has not provided input. Can approve or reject a task in a deployment awaiting user input.
Run ad hoc	Run an ad hoc task from the pipeline actions menu, allowing a user to deploy or package against a selected environment.
Skip on failure	Skip a failed task and resume deployment from next available task. **Note:** You cannot skip the Generate artifact and Publish to production tasks.

System privileges

System privileges

Privilege	Description
Manage system settings	Manage orchestrator system settings. Provides the option to manage roles and email configuration. This privilege is typically applied to super admins only.
Dev studio switch	Switch to Dev Studio from Deployment Manager portal.
People	Create, edit or delete users of Deployment Manager.
Manage authentication profile	Add, edit or delete Authentication Profiles required for inter-system communication.

Default roles

Default roles

Role	Description
SuperAdmin	Perform all actions on the Deployment Manager system for all applications.
AppAdmin	Perform all actions specific to an authorized application, such as manage People, Authentication Profiles, Pipelines, Deployments and Tasks.
PipelineUser	View a pipeline and start a deployment for an authorized application.

Providing access to Dev Studio to a role

Deployment Manager provides a dedicated portal from which you can access features. From within Deployment Manager, when you configure pipeline details, you can open, modify, and create repositories and authentication profiles in Dev Studio if you have permissions to use the Dev Studio portal.

To provide access to the Dev Studio portal for a role, complete the following steps:

1. In the navigation pane of Deployment Manager, click **Users**, and then click **Roles and privileges**.
2. Do one of the following actions:
    - To add a role, click **Add role**.
    - To modify a role, click **Edit**.
3. In the **Add role** or **Edit Role** dialog box, in the **Name** field, enter a name for the role.
4. Click **Dev Studio switch**.
5. Click **Submit**.

Result: If you specify Dev Studio as a default portal for the *PegaDeploymentManager:Administrators* access group, all the users that you add in the Deployment Manager portal can access Dev Studio.

Adding users and specifying their roles

If you are a super administrator or application administrator, you can add users to Deployment Manager and specify their roles. Only super administrators can create other super administrators or application administrators who can access one or more applications. Application administrators can create other application administrators for the applications that they manage.

To add users, do the following steps:

1. In the navigation pane of Deployment Manager click **Users**, and then click **People**.
2. On the **People** page, click **Add user**.
3. In the **Add user** dialog box, click the **User** field, and do one of the following actions:
    - Press the Down arrow key and select the user that you want to add.

    - Enter an email address.
4. Click **Add**.
5. From the **Role** list, select the role to assign to the user.
6. If you selected the App admin role or a custom role, in the **Applications** field, enter the application name that the user can access.
7. Click **Send invite** to send an email, which contains the user name and a randomly generated password for the user to log in to Deployment Manager with, to the user.

Modifying user roles and privileges

Super administrators can give other users super administrative privileges or assign them as application administrators to any application. Application administrators can assign other users as application administrators for the applications that they manage.

To modify user roles and privileges, do the following steps:

1. In the navigation pane of Deployment Manager click **Users**, and then click **People**.
2. On the **People** page, click the user.
3. In the **Roles and privileges** section, modify the user role and applications that they can access, as appropriate.
4. Click **Save**.

Modifying your user details and password

You can modify your own user details, such as first and last name, and you can change your password.

To update your information, do the following steps:

1. In the navigation pane of Deployment Manager click **Users**, and then click **People**.
2. On the **People** page, click your user name.
3. In the **Personal details** section, modify your name, email address, and phone number, as appropriate.
4. To change your password:
    a. Click **Update password.**
    b. In the **Change operator ID** dialog box, enter your new password, reenter it to confirm it, and then click **Submit**.
5. Click **Save**.

Deleting users

If you are a super administrator or application administrator, you can delete users for the applications that you manage.

To delete users, do the following steps:

1. In the navigation panel of Deployment Manager, click **Users**, and then click **People**.
2. On the **People** page, click the **Delete icon** for the user that you want to delete.

Creating pipelines

When you add a pipeline, you define all the stages and tasks that you want to do on each system. For example, if you are using branches, you can start a build when a branch is merged. If you are using a QA system, you can run test tasks to validate application data. Pipelines are the first step in getting started with setting up a process for deploying a change through the release. See the articles below for more information on creating each of the available pipeline types.

- Creating a deployment pipeline

  Deployment pipelines are an application pipeline type to generate artifacts from the system of record and deliver them through the various stages of the workflow. When you add a pipeline, you define all the stages and tasks that you want to do on each system. If you are using a QA system, you can run test tasks to validate application data. Pipelines are the first step in getting started with setting up a process for deploying a change through the release.

- Creating a merge pipeline

  Merge pipelines enable developers to submit rule branches for a merge after validating the changes against specific quality gates such as guardrails, branch review, and automated tests. For example, if you are using branches, you can start a build when a branch is merged. Pipelines are the first step in getting started with setting up a process for deploying a change through the release.

- Creating a business change pipeline

  Business change pipelines are used to support everyday, business-as-usual changes to your application. This pipeline type allows users to respond to changing requirements by modifying and deploying application rules in a controlled manner. When you add a pipeline, you define all the stages and tasks that you want to do on each system. Pipelines are the first step in getting started with setting up a process for deploying a business change outside of an enterprise release.

- [Creating a data migration pipeline]

  Create a pipeline by defining the production and simulation environments and the application details for the pipeline. By using a data migration pipeline, you can export and import simulation data automatically.

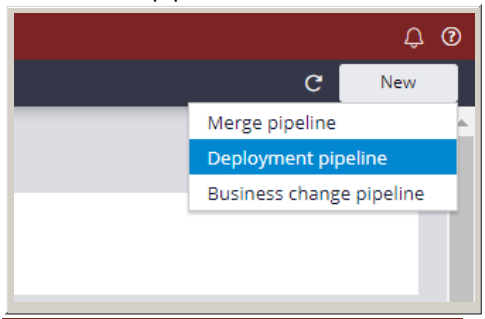
Creating a deployment pipeline

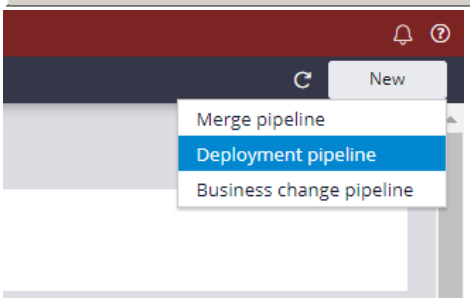
Deployment pipelines are an application pipeline type to generate artifacts from the system of record and deliver them through the various stages of the workflow. When you add a pipeline, you define all the stages and tasks that you want to do on each system. If you are using a QA system, you can run test tasks to validate application data. Pipelines are the first step in getting started with setting up a process for deploying a change through the release.

Add a new deployment pipeline

To add a new pipeline, follow the steps below:

1. Click **Pipelines** and select **Application pipelines**.
2. Click **New** on the top right.
3. Select the **Deployment pipeline** template. Pipeline templates drive the required information needed to complete the creation of a pipeline. For more information, see [Pipeline templates].



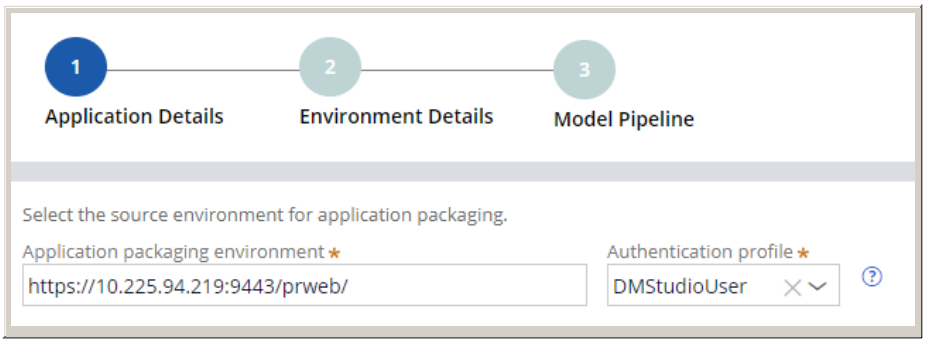


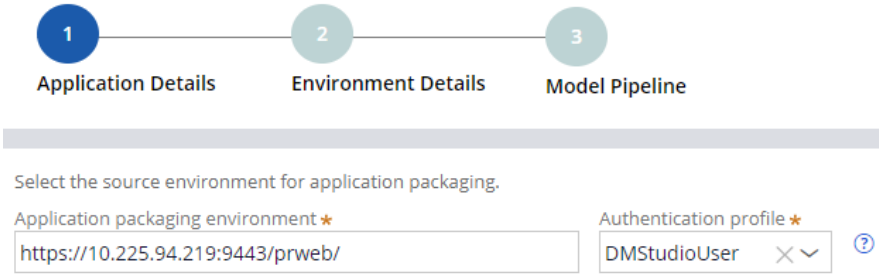
Note: Remember to save your configuration if you choose to navigate away from any of the pipeline creation screens. Progress will be lost if configurations are not saved. Continue below to add application environment and pipeline model details.

Configuring a new pipeline - Application Details

Application Packaging Environment

1. Specify the environment URL from where the application is packaged in the **Application packaging environment** field and select or enter the Authentication profile to be used for connection to the application packaging environment.

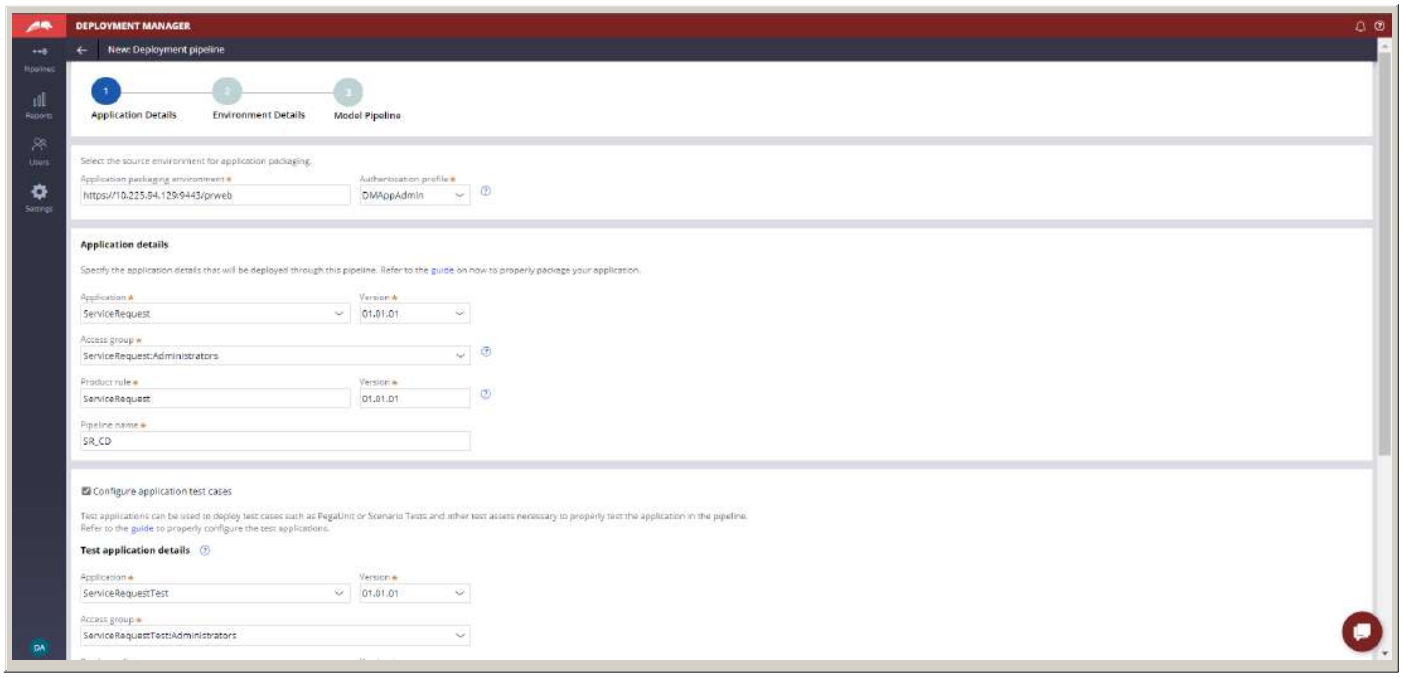


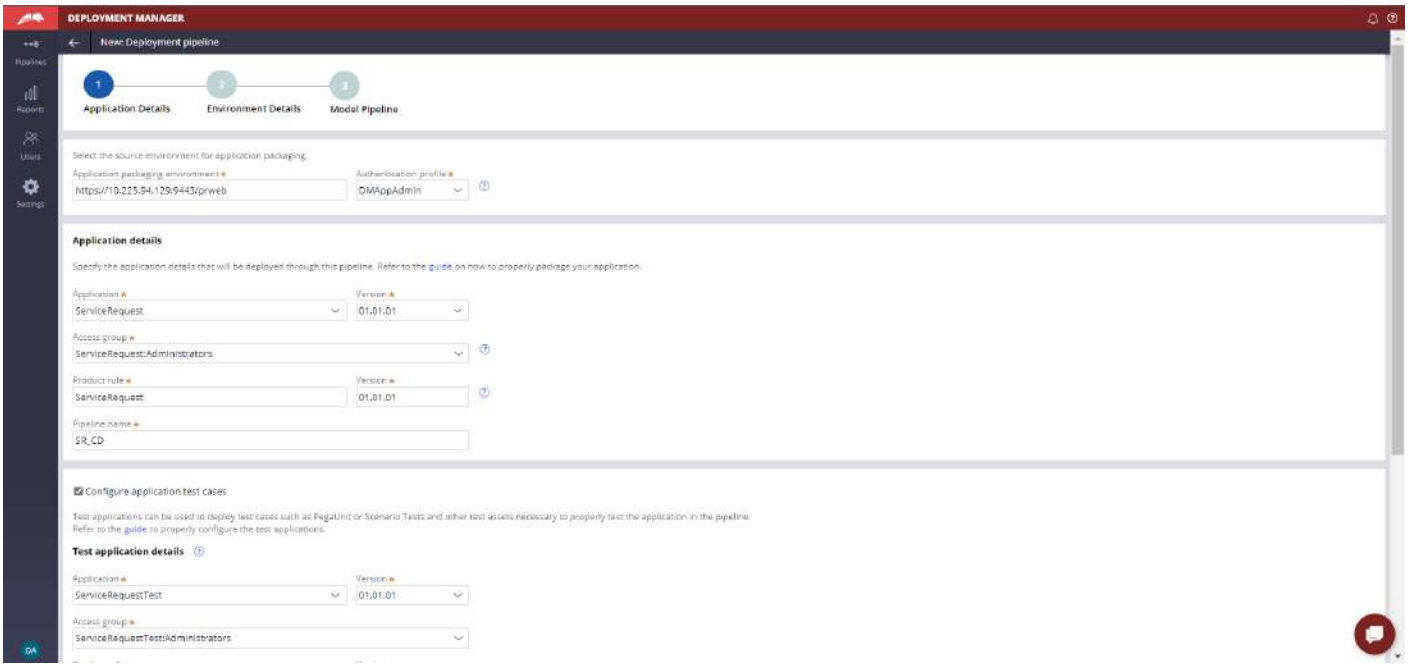


Application Details

Application details specify the application that this pipeline progresses through.

1. In the **Application field**, select the name of the application.
2. In the **Version** field, select the application version.
3. In the **Access group** field, select the access group for which pipeline tasks are run.
    - This access group must be present on all candidate systems and have at least the sysadmin4 role.
4. In the **Product rule** field, enter the name of the product rule that defines the contents of the application.
5. In the **Version field**, enter the product rule version.
6. In the **Pipeline name** field, enter a unique name for the pipeline.

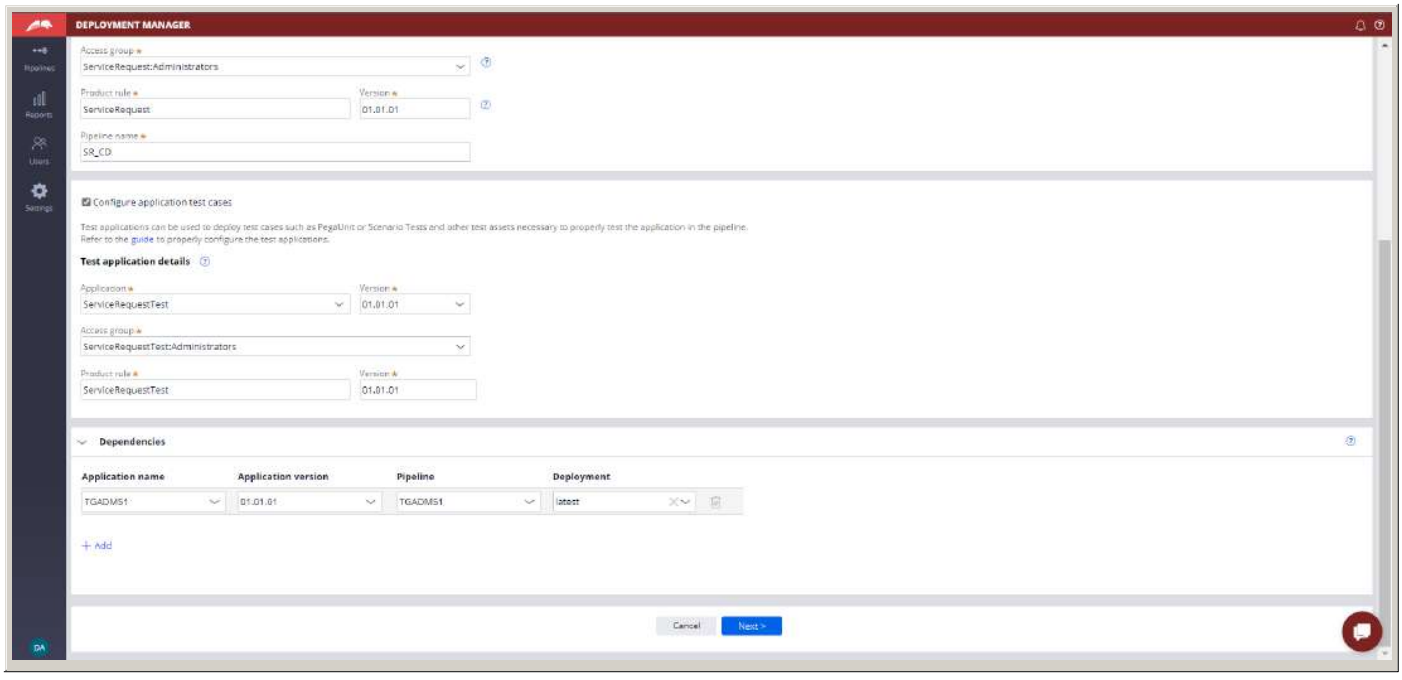


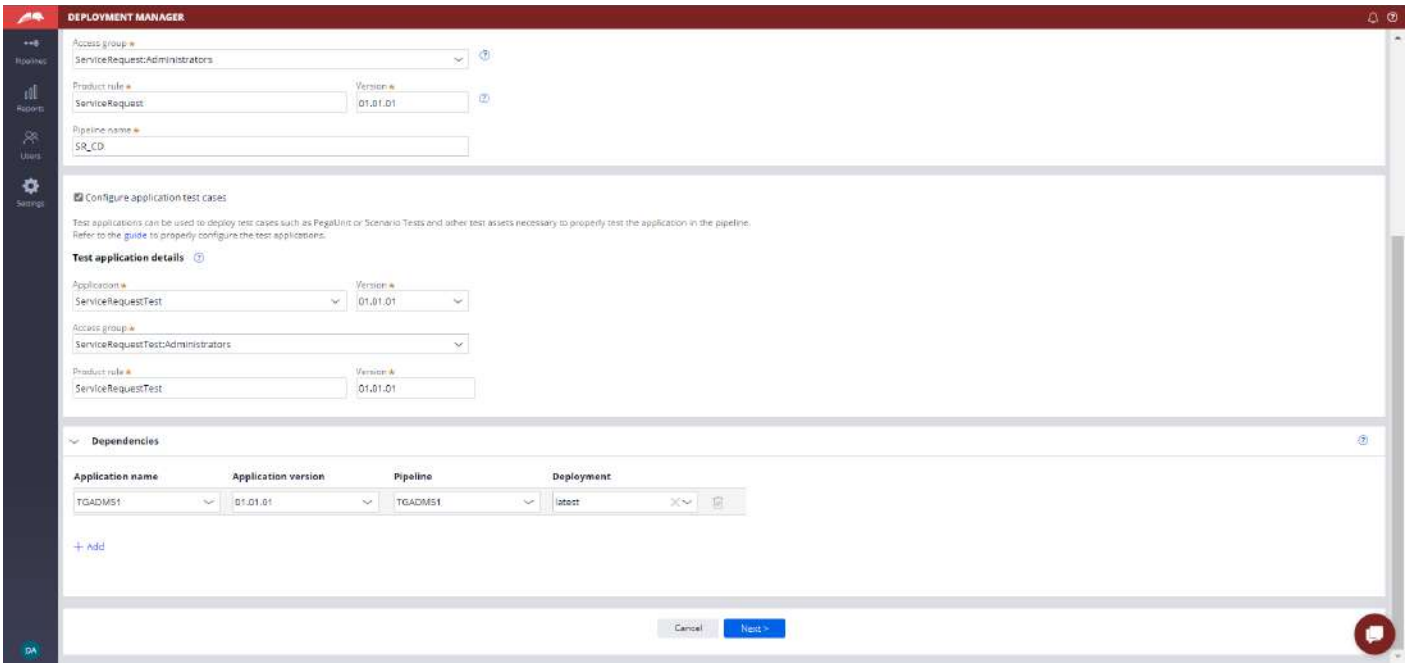


Deploy Application Test Cases

If you are using a separate product rule to manage test cases, in the Application test cases section select the **Configure application test cases** check box to deploy a test case. Then, complete the following steps to generate the artifact consisting of the test cases to run on a target environment.

1. In the **Application field**, select the name of the application.
2. In the **Version** field, select the application version.
3. In the **Access group** field, select the access group for which pipeline tasks are run.
    - This access group must be present on all candidate systems and have at least the sysadmin4 role.
4. In the **Product rule** field, enter the name of the product rule that defines the contents of the application.
5. In the **Version field**, enter the product rule version.

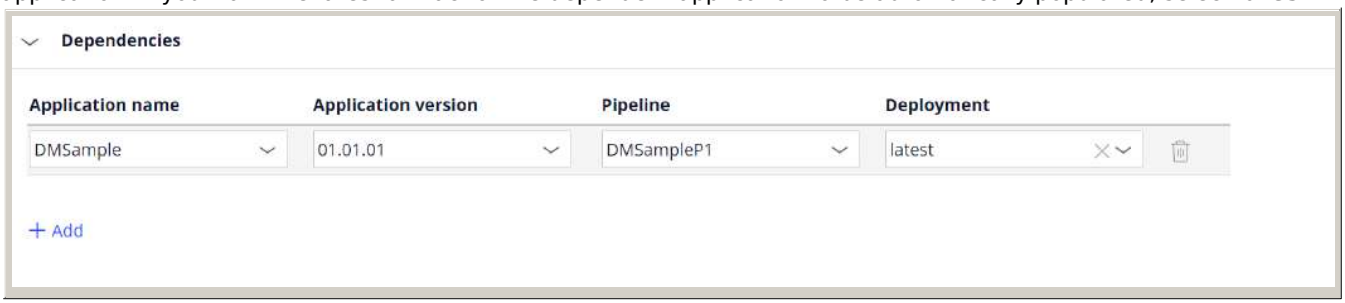




Dependencies

To configure dependent applications:

1. Click **Add** in the Dependencies section.
2. In the **Application** field, select the application name.
3. In the **Application version** field, select the application version.
4. In the **Pipeline** field, select the pipeline.
5. In the **Deployment** field, select the deployment that contains the production-ready artifact of the dependent application. If you want the latest artifact of the dependent application to be automatically populated, select **latest**.



Configuring a new pipeline - Environment Details

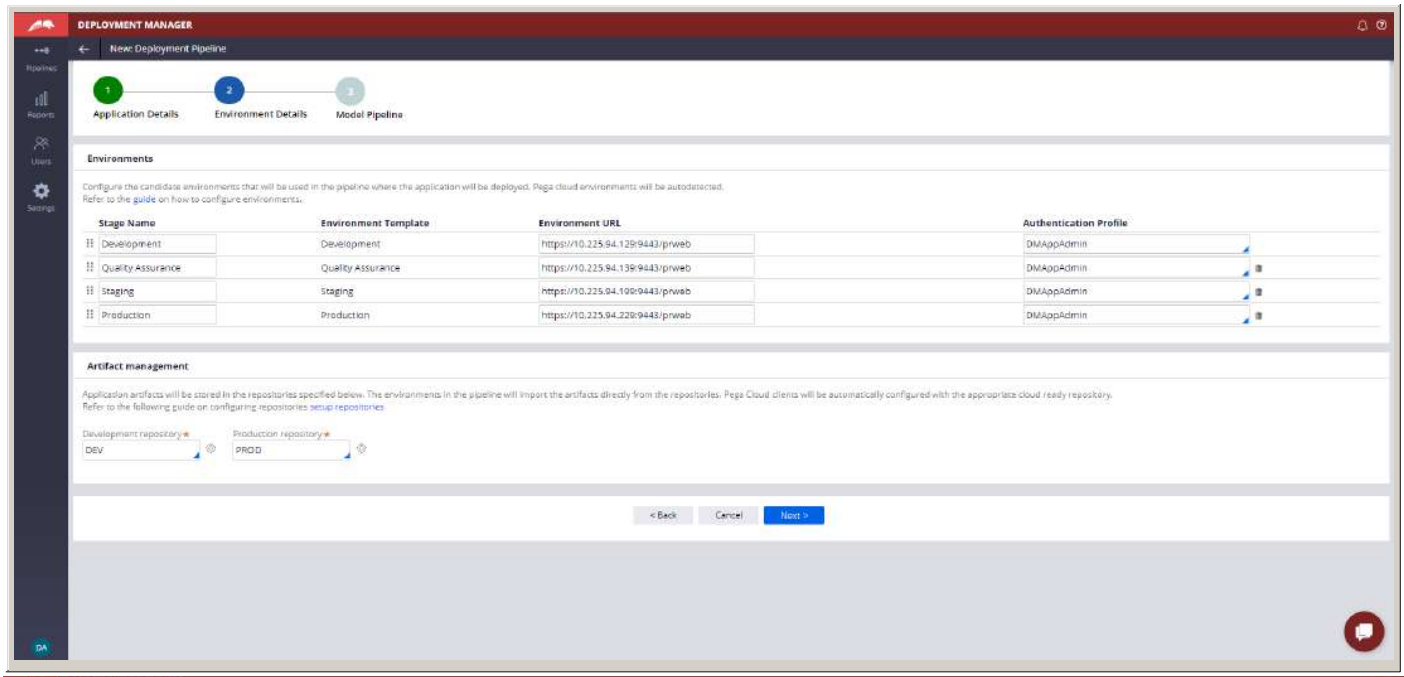
Environments

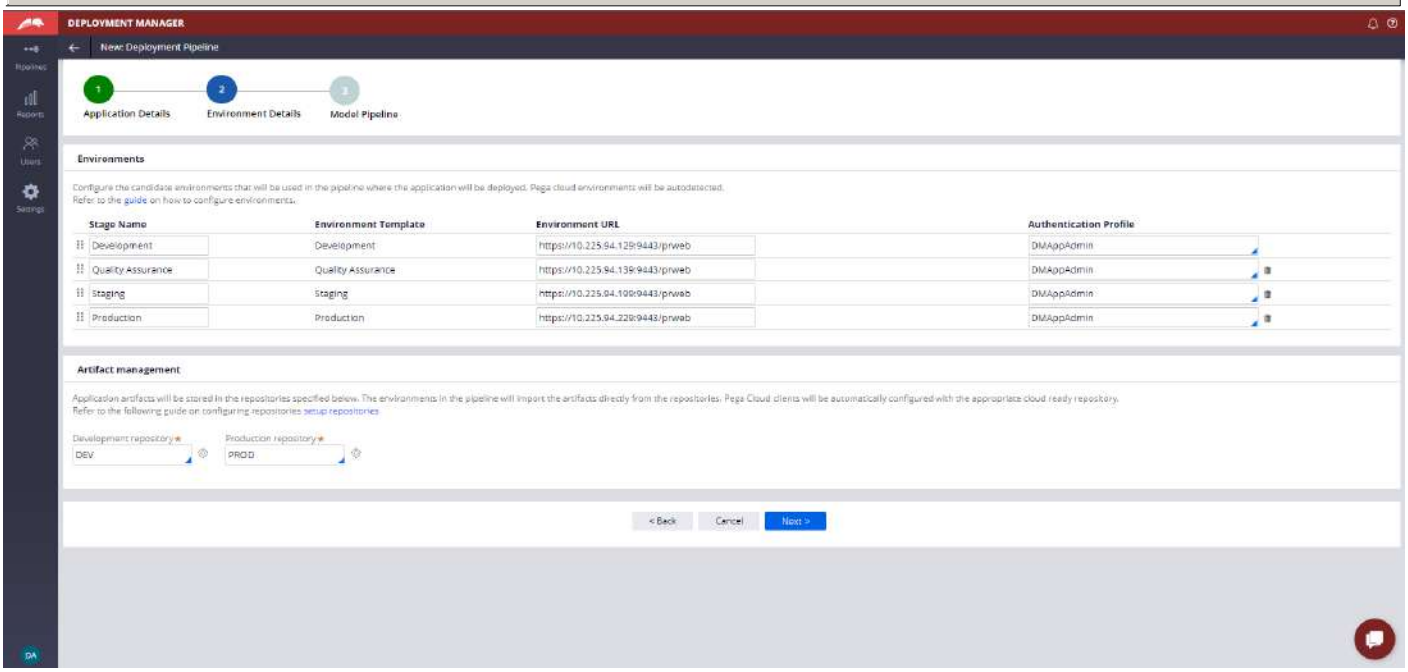
1. The default stages are populated based on the pipeline template created. See Pipeline templates for more information.
2. In the **Environment URL** field for the system, press the Down arrow key and select the URL of the system.
3. If you are using your own authentication profiles, in the **Authentication Profile** field for the system, press the Down arrow key and select the authentication profile that you want to communicate from the orchestration server to the system.
    - By default, the fields are populated with the DMAppAdmin authentication profile.

Note: The pipeline model screen will support the addition of more than four stages.

Repositories

1. In the Artifact management section, specify the development and production repositories through which the product rule that contains application contents moves through the pipeline.
2. In the **Development repository** field, select the development repository.
3. In the **Production repository** field, select the production repository.
4. Click **Next**.





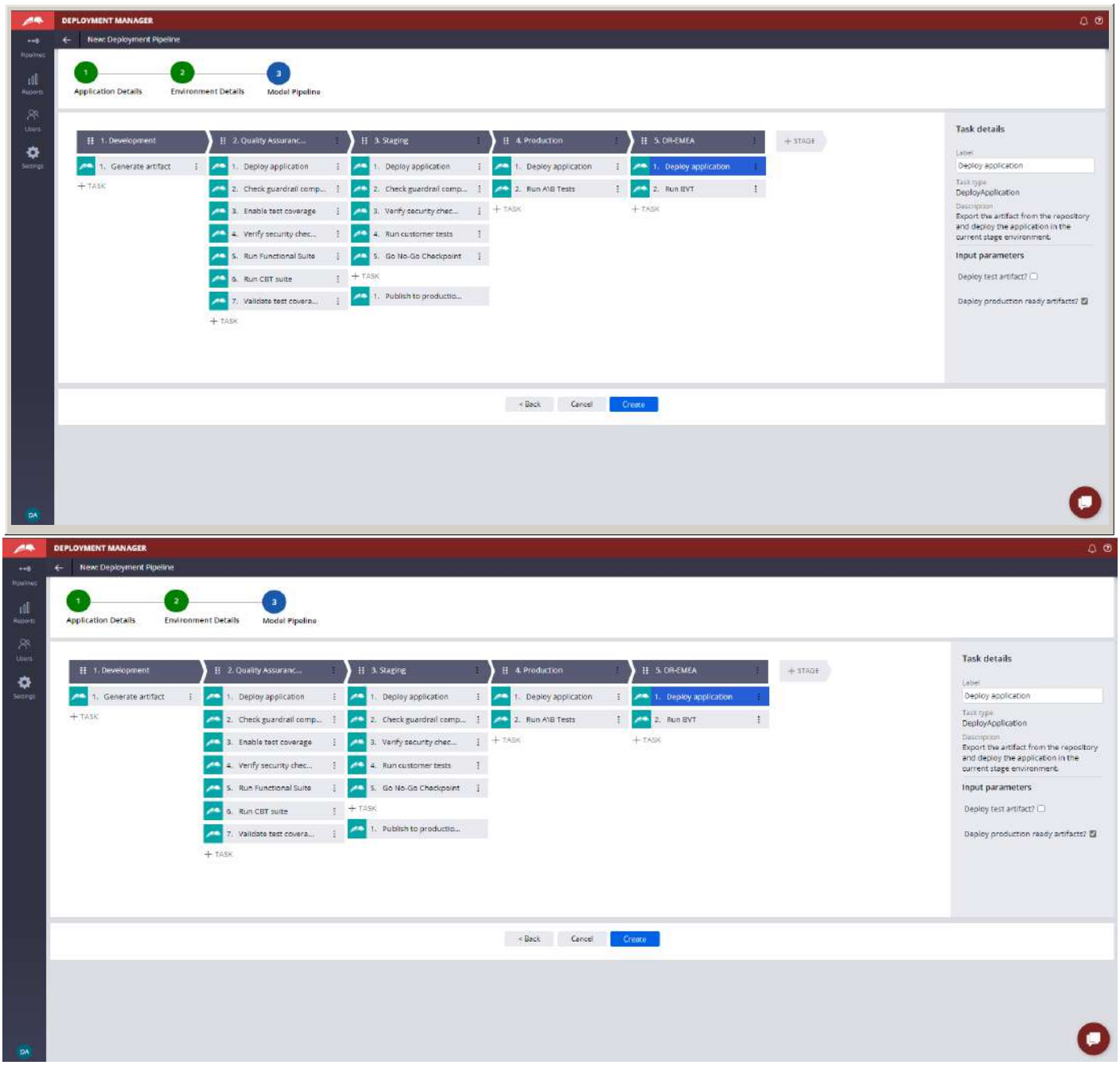
Configuring a new pipeline - Model Pipeline

From the model pipeline screen, you can tailor your stages and tasks to replicate the quality standard of the release process defined by your organization. You can modify the deployment process here by configuring tasks and modifying stages.

For more information on adding, editing, or deleting a stage, see .

1. Stages can be modified here by clicking the **More** icon on each stage.
2. To add a stage, click the **+Stage** button. You can apply an environment template to each new stage. See Environment templates.
3. To add a task to a stage, click the **+Task** button within a stage. The available tasks that you can add are specific to the associated stage. The available tasks that you can add are specific to the associated stage. For a list of task related information, see the Task catalog.
    1. The environment template assigned to a stage drives the availability of tasks. Mandatory tasks cannot be deleted.

4. Click **Create** to save the pipeline.



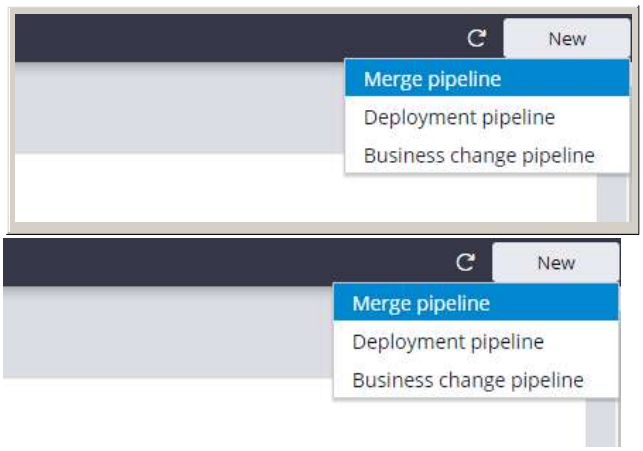
Creating a merge pipeline

Merge pipelines enable developers to submit rule branches for a merge after validating the changes against specific quality gates such as guardrails, branch review, and automated tests. For example, if you are using branches, you can start a build when a branch is merged. Pipelines are the first step in getting started with setting up a process for deploying a change through the release.

Add a new merge pipeline

To add a new pipeline, follow the steps below:

1. Click **Pipelines** and select **Application pipelines**.
2. Click **New** on the top right.
3. Select the **Merge pipeline** template. Pipeline templates drive the required information needed to complete the creation of a pipeline. For more information, see Pipeline templates.



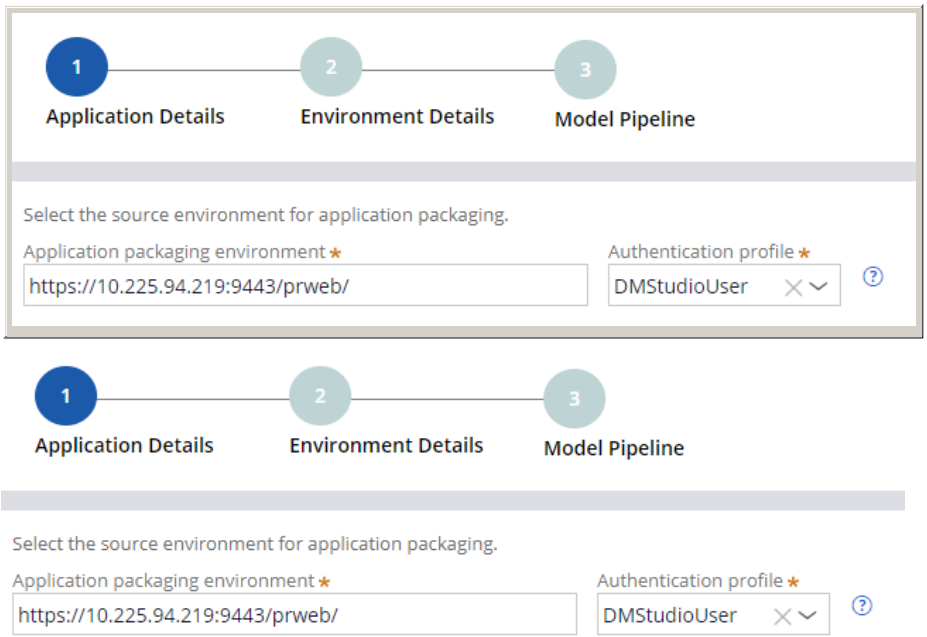
4. Click **Create** after configuring the following sections on Application details, environment, and process model.

Note: Remember to save your configuration if you choose to navigate away from any of the pipeline creation screens. Progress will be lost if configurations are not saved.

Configuring a new pipeline - Application Details

Application Packaging Environment

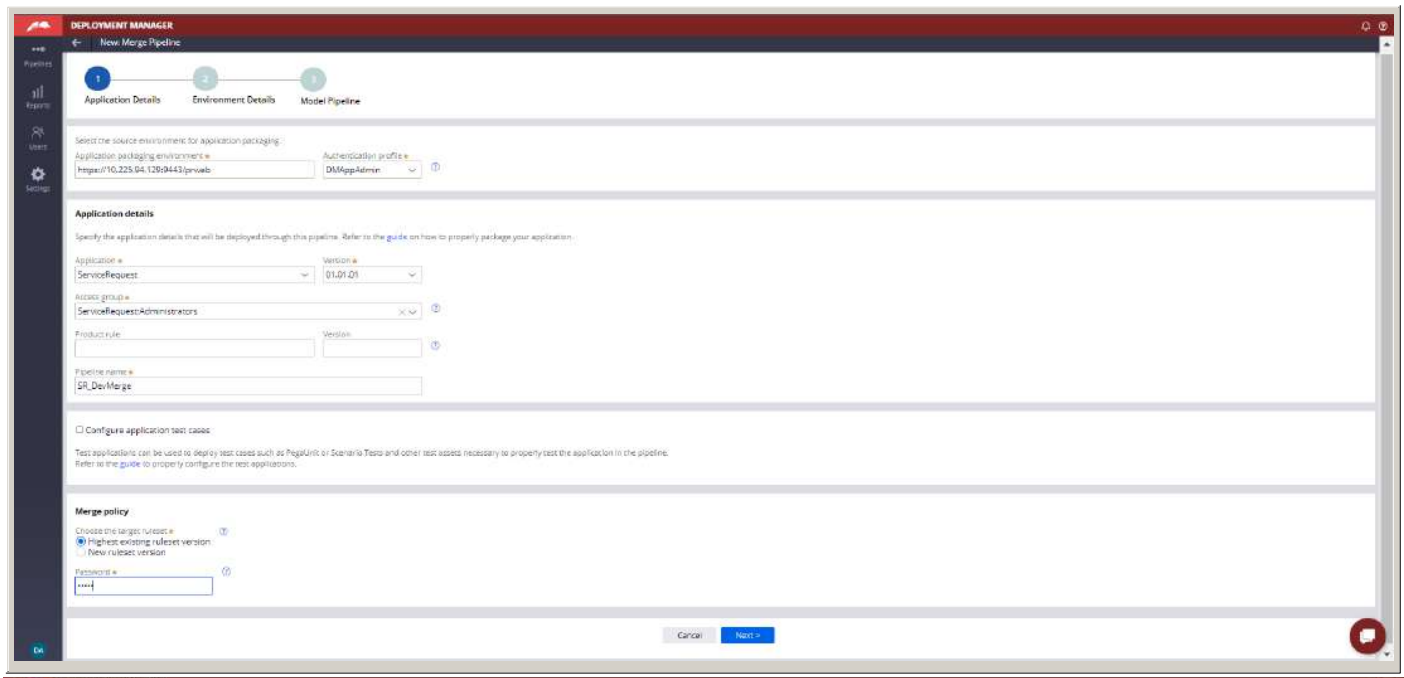
1. Specify the environment URL from where the rule branches are merged in the **Application packaging environment** field and select or enter the Authentication profile to be used for connection to the application packaging environment.

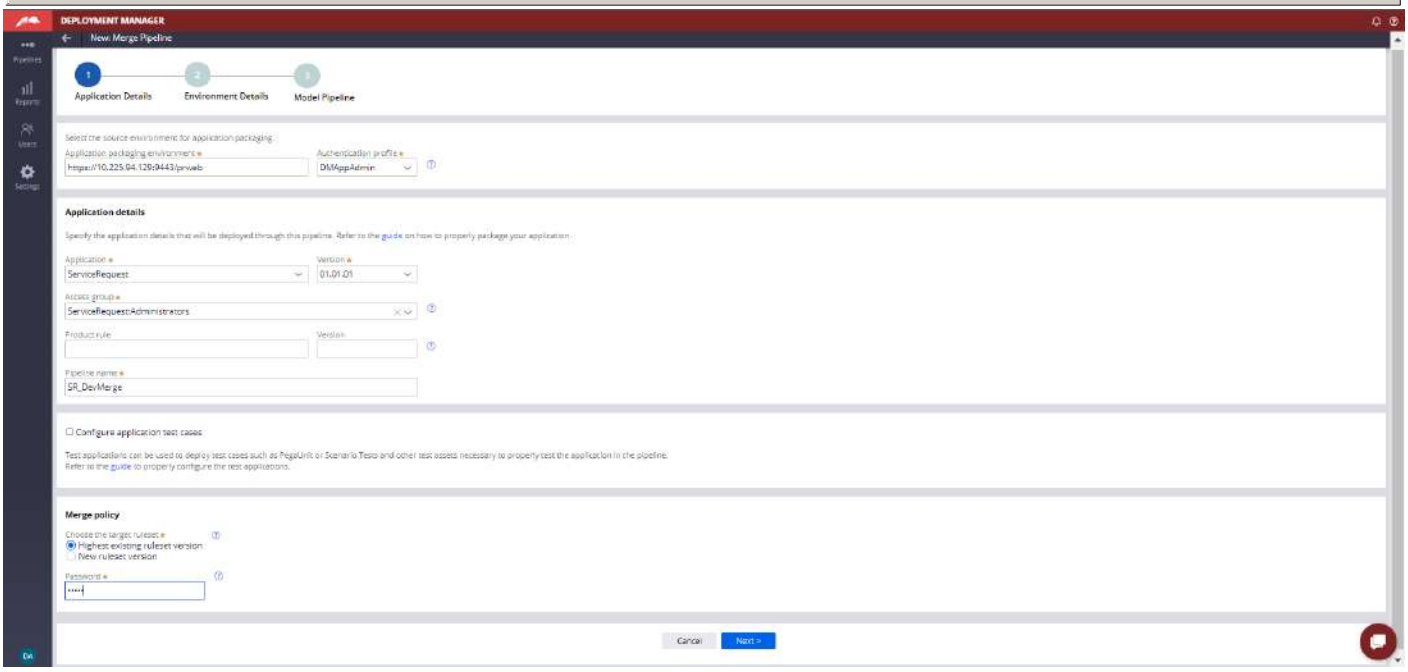


Application Details

Application details specify the application that this pipeline progresses through.

1. In the **Application field**, select the name of the application.
2. In the **Version** field, select the application version.
3. In the **Access group** field, select the access group for which pipeline tasks are run.
    - This access group must be present on all candidate systems and have at least the sysadmin4 role.
4. Skip the **Product rule** and **Version** field.
5. In the **Pipeline name** field, enter a unique name for the pipeline.





Merge Policy

To configure the target ruleset version used when merging branches:

1. In the **Choose the target ruleset**, field, select to use either the highest existing ruleset version, or input your own using the **New ruleset version** field.
2. Enter the password for the ruleset version.

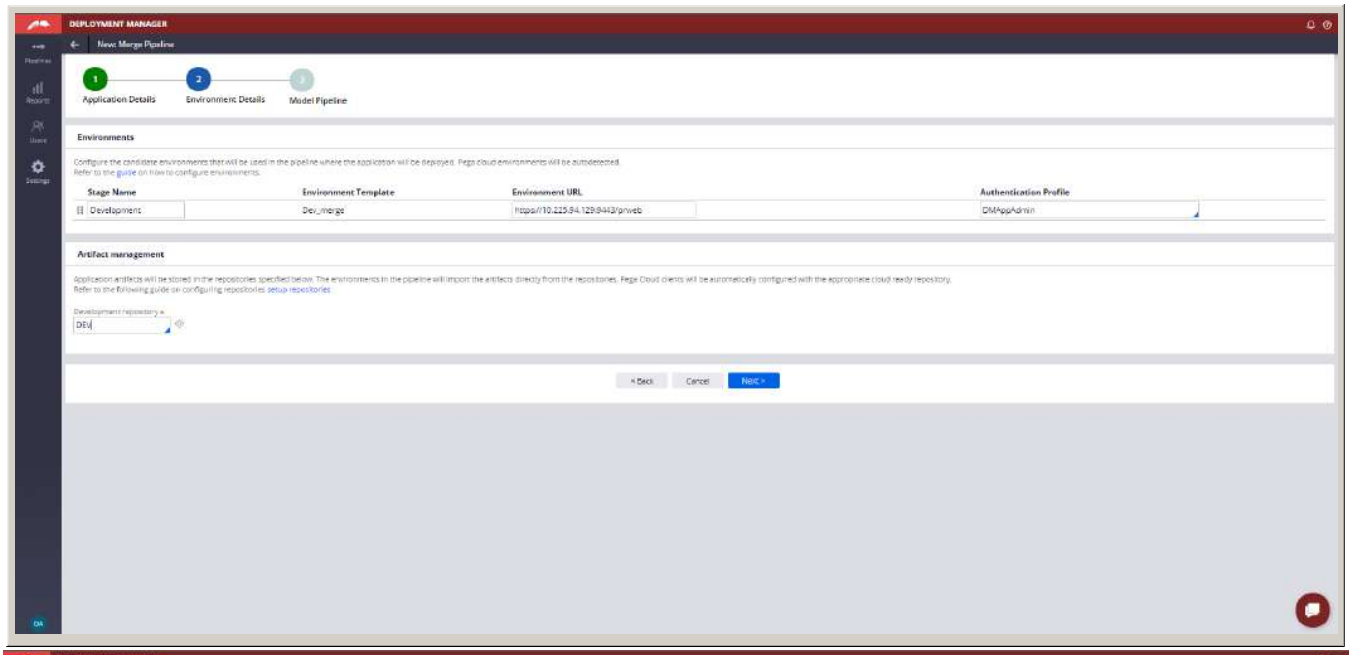
Configuring a new pipeline - Environment Details

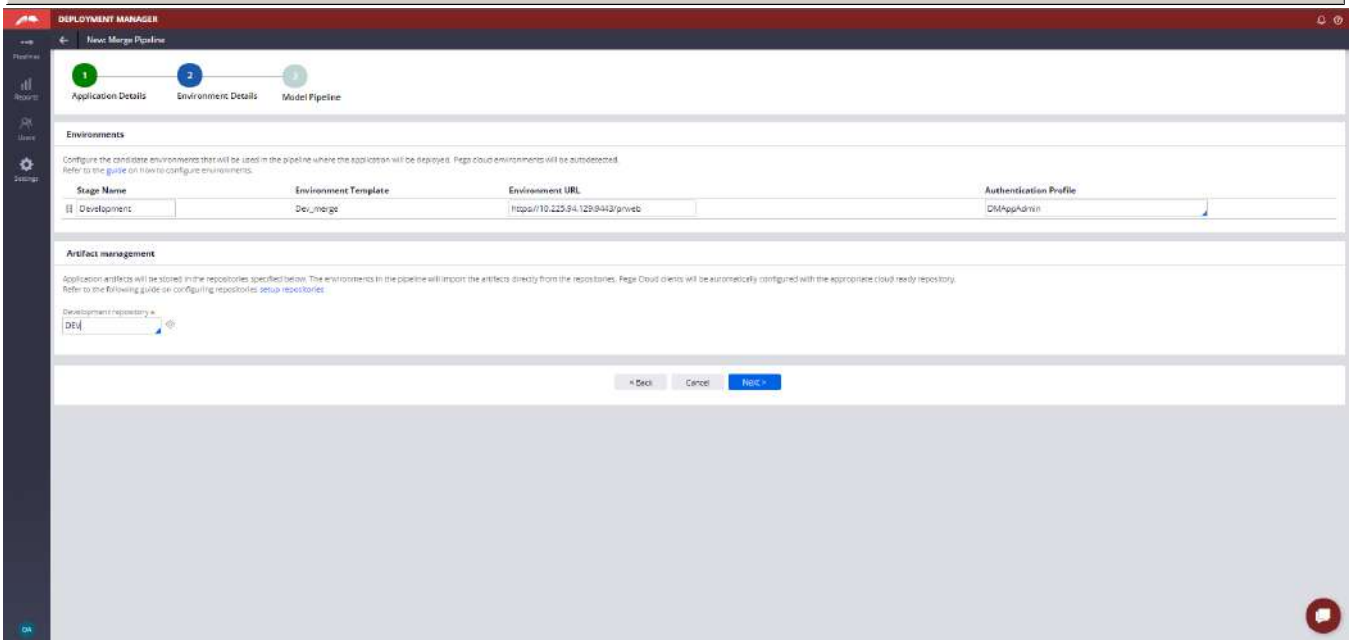
Environments

1. Merge pipelines support a single stage by default, Development.
2. Validate the **Environment URL**.
3. If you are using your own authentication profiles, in the **Authentication Profile** field for the system, press the Down arrow key and select the authentication profile that you want to communicate from the orchestration server to the system.
    - By default, the fields are populated with the DMAppAdmin authentication profile.

Repositories

1. In the Artifact management section, specify the development and production repositories through which the product rule that contains application contents moves through the pipeline.
2. In the **Development repository** field, select the development repository.
3. Click **Next**.





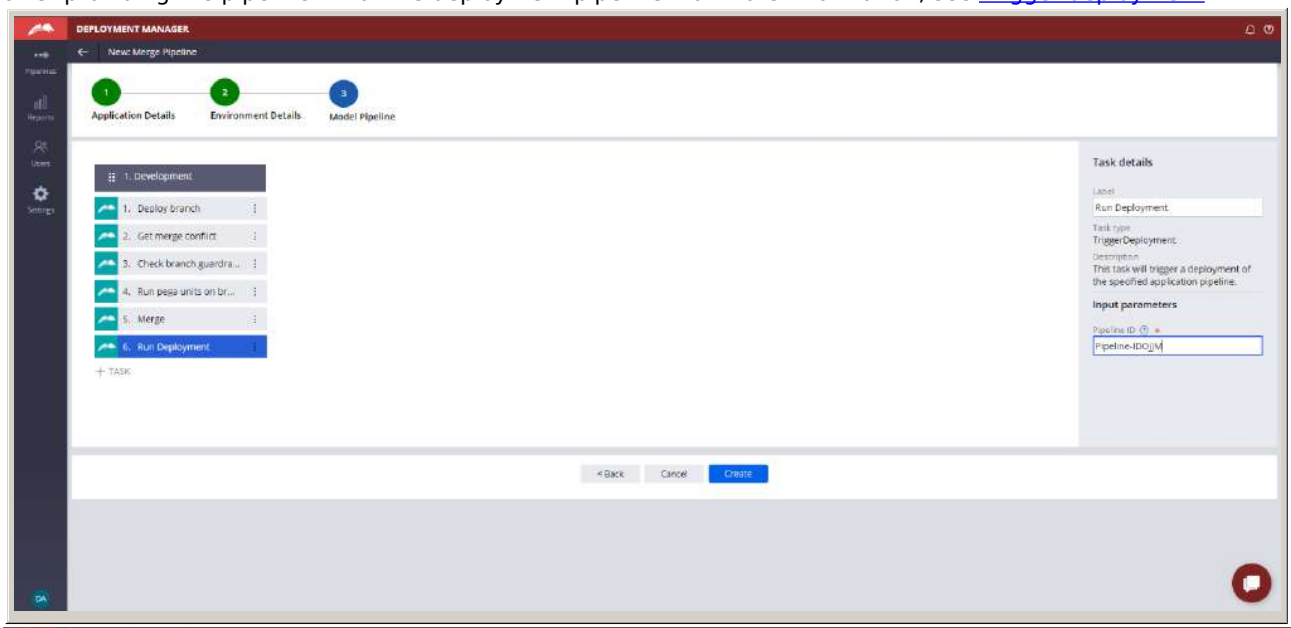
Configuring a new pipeline - Model Pipeline

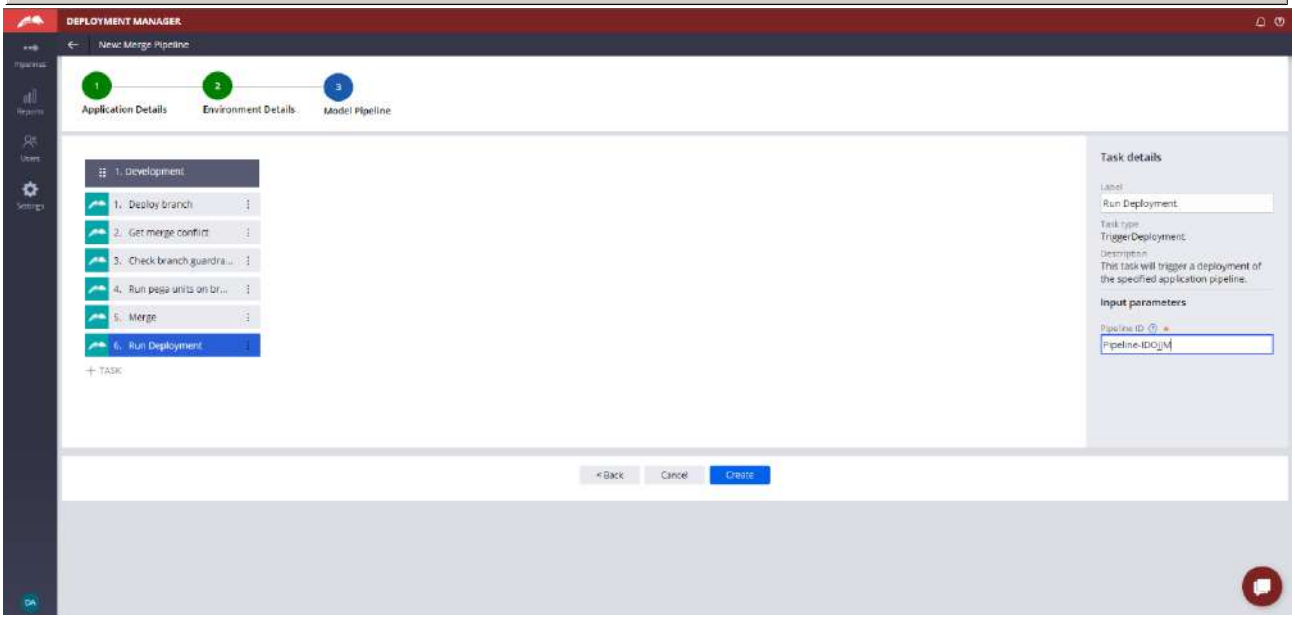
From the model pipeline screen, you can tailor your development stage by customizing the tasks.

1. Stages can be modified here by clicking the **More** icon on each stage.
2. You cannot add stages to a merge pipeline. For more information on the Development (Dev merge) stage, see

Environment templates.

3. To add a task to a stage, click the **+Task** button within a stage. The available tasks that you can add are specific to the associated stage. The available tasks that you can add are specific to the associated stage. For a list of task related information, see the Task catalog.
    1. The environment template assigned to a stage drives the availability of tasks. Mandatory tasks cannot be deleted.
    2. If you want to trigger a deployment pipeline to achieve continuous delivery, add a **Trigger Deployment** task after providing the pipeline ID for the deployment pipeline. For more information, see Trigger deployment.





Creating a business change pipeline

Business change pipelines are used to support everyday, business-as-usual changes to your application. This pipeline type allows users to respond to changing requirements by modifying and deploying application rules in a controlled manner. When you add a pipeline, you define all the stages and tasks that you want to do on each system. Pipelines are the first step in getting started with setting up a process for deploying a business change outside of an enterprise release.

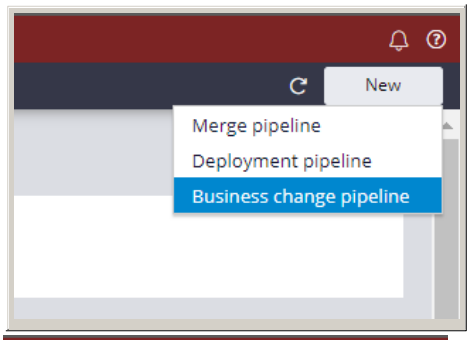
Note: Applications being managed by a business change pipeline cannot automatically increment a minor version higher than v99. You must manually roll over your application version using the New application task, should your application or ruleset versions encounter this scenario. **Note:** The application shall specify the major and minor version when configuring a business change pipeline.

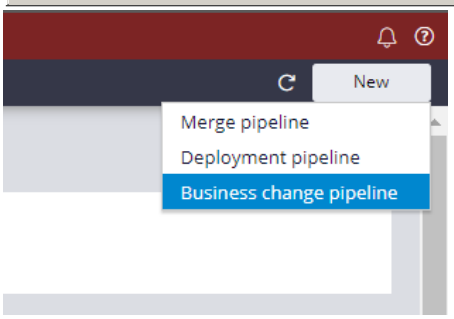
Add a new business change pipeline

To add a new pipeline, follow the steps below:

1. Click **Pipelines** and select **Application pipelines**.

2. Click **New** on the top right.
3. Select the business change pipeline template. Pipeline templates drive the required information needed to complete the creation of a pipeline. For more information, see Pipeline templates.



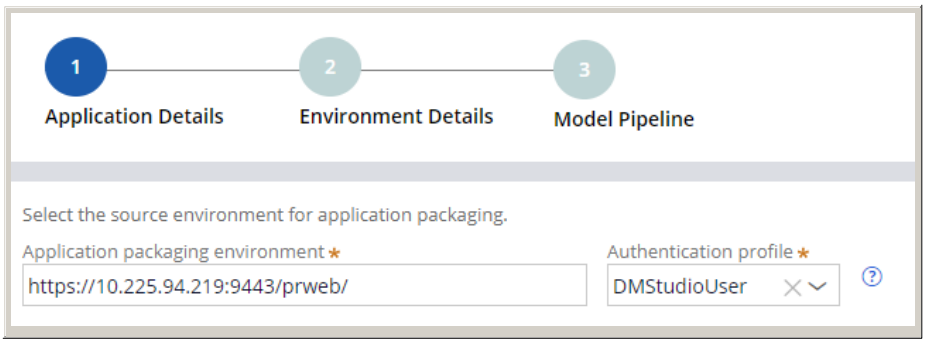


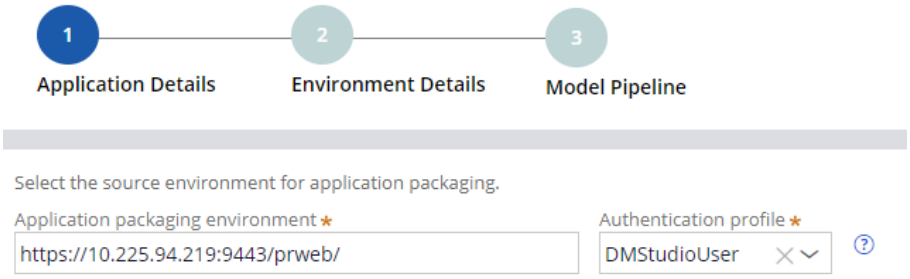
Note: Remember to save your configuration if you choose to navigate away from any of the pipeline creation screens. Progress will be lost if configurations are not saved.

Configuring a new pipeline - Application Details

Application Packaging Environment

1. Specify the environment URL from which the business changes will be merged and progressed in the **Application packaging environment** field and select or enter the Authentication profile to be used for connection to the application packaging environment.



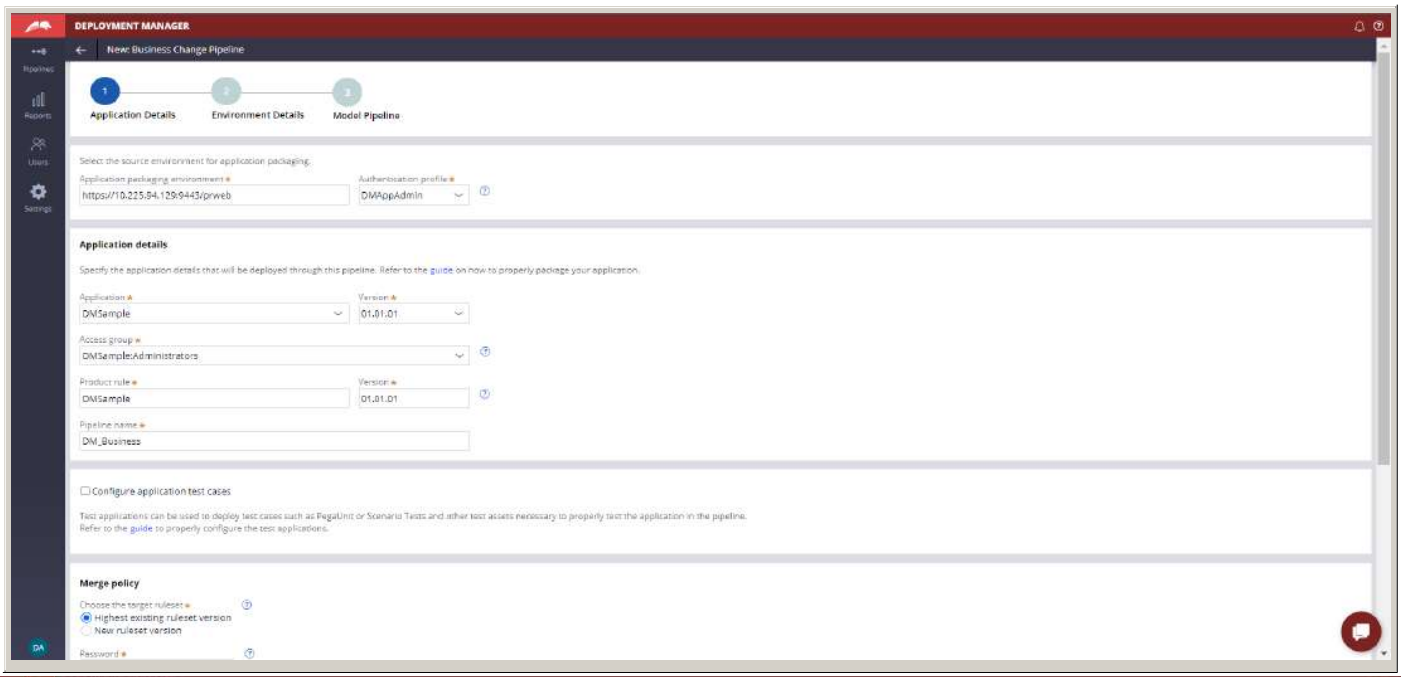


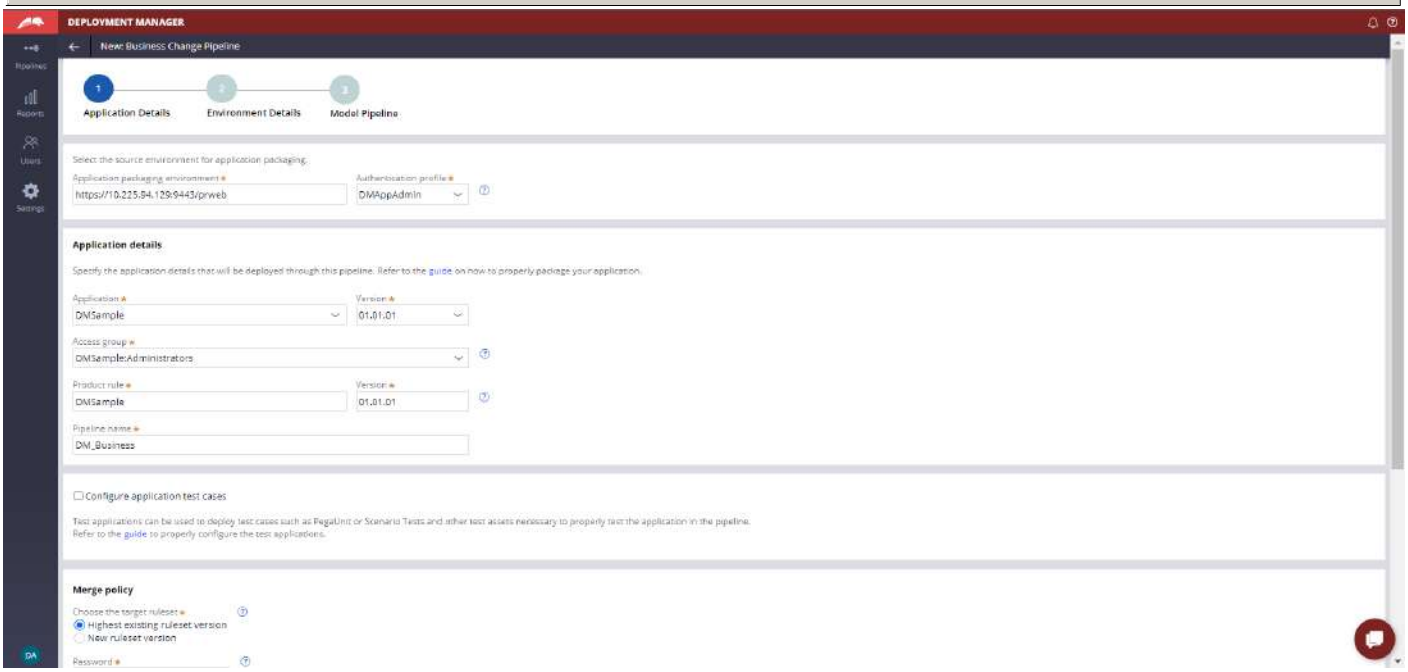
Application Details

Application details specify the application that this pipeline progresses through.

1. In the **Application field**, select the name of the application.
2. In the **Version** field, select the application version.
3. In the **Access group** field, select the access group for which pipeline tasks are run.
   - This access group must be present on all candidate systems and have at least the sysadmin4 role.
4. In the **Product rule** field, enter the name of the product rule that defines the contents of the application.

5. In the **Version field**, enter the product rule version.
6. In the **Pipeline name** field, enter a unique name for the pipeline.

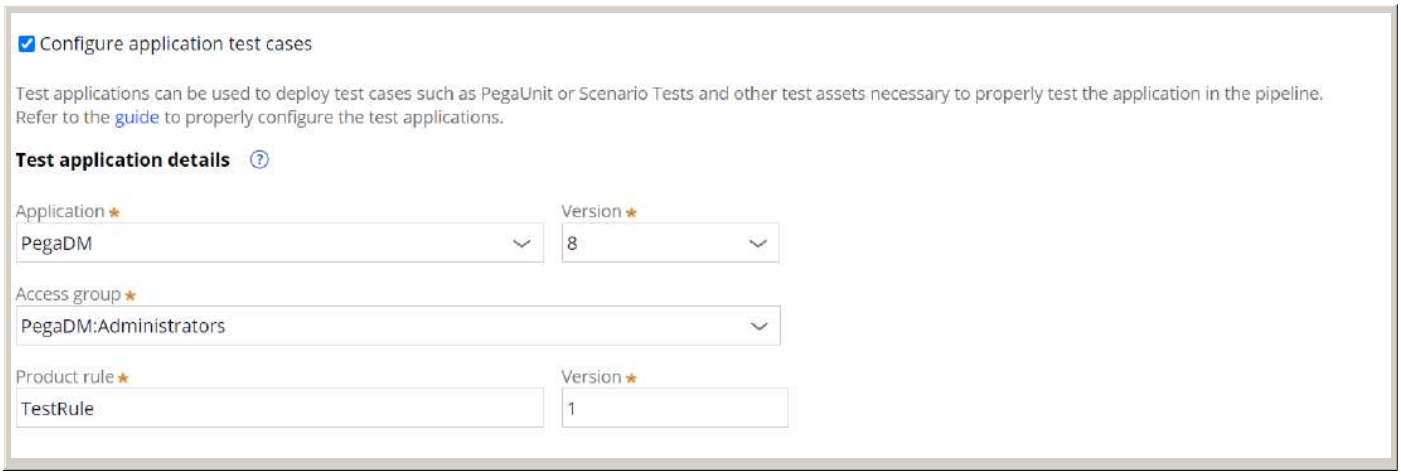




Deploy Application Test Cases

If you are using a separate product rule to manage test cases, in the Application test cases section select the **Configure application test cases** check box to deploy a test case. Then, complete the following steps to generate the artifact consisting of the test cases to run on a target environment.

1. In the **Application field**, select the name of the application.
2. In the **Version** field, select the application version.
3. In the **Access group** field, select the access group for which pipeline tasks are run.
    - This access group must be present on all candidate systems and have at least the sysadmin4 role.
4. In the **Product rule** field, enter the name of the product rule that defines the contents of the application.
5. In the **Version field**, enter the product rule version.



Configure application test cases

Test applications can be used to deploy test cases such as PegaUnit or Scenario Tests and other test assets necessary to properly test the application in the pipeline.
Refer to the guide to properly configure the test applications.

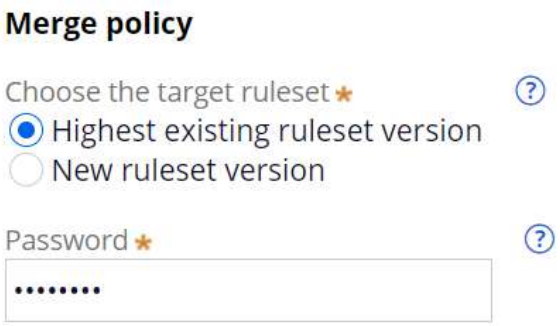
Test application details

Application *
PegaDM
Version *
8
Access group *
PegaDM:Administrators
Product rule *
TestRule
Version *
1

Merge Policy

To configure the target ruleset version used when merging branches:

1. In the **Choose the target ruleset**, field, select to use either the highest existing ruleset version, or input your own using the **New ruleset version** field.
2. Enter the password for the ruleset version.



Configuring a new pipeline - Environment Details

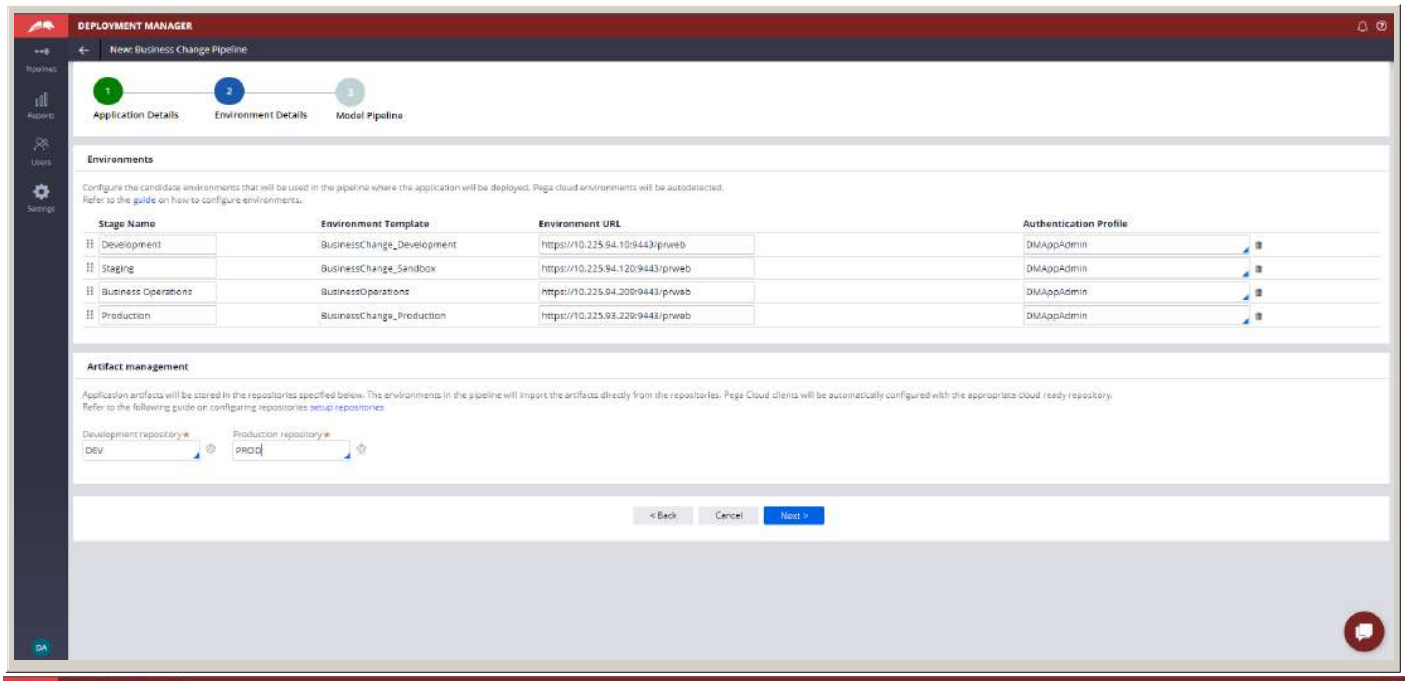
Environments

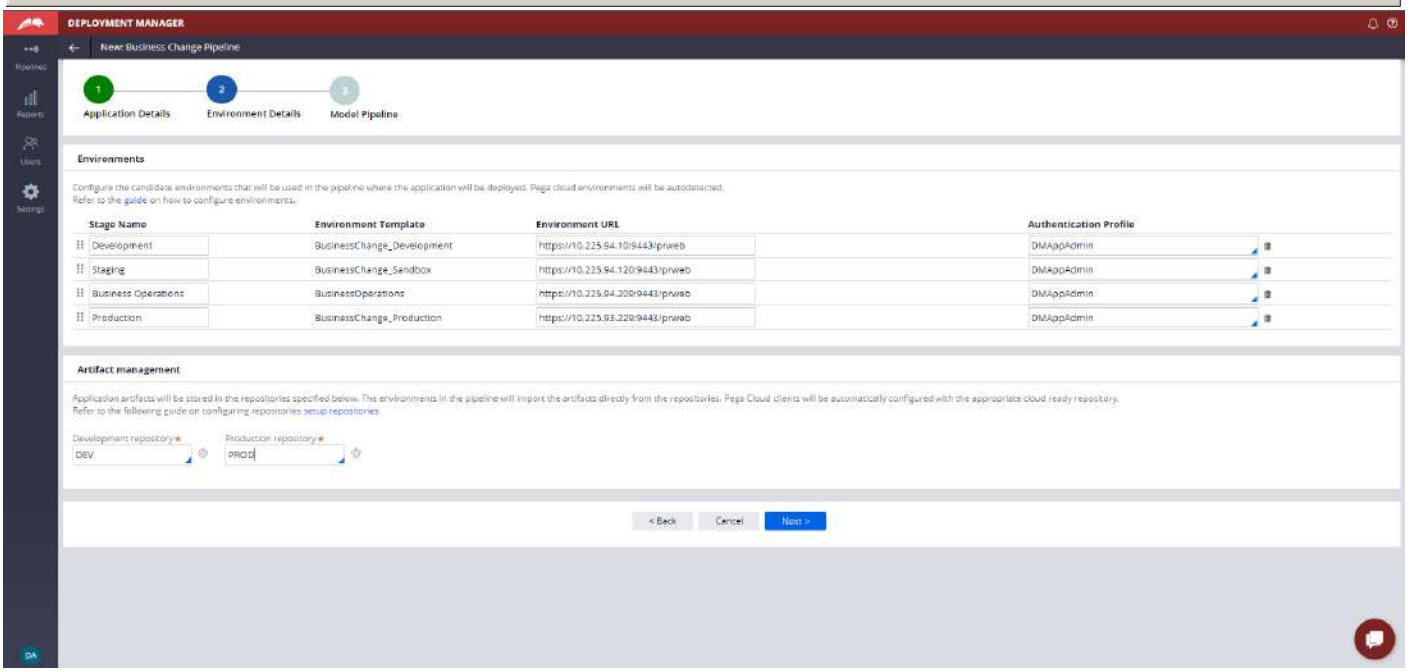
1. The default stages are populated based on the pipeline template created. See Pipeline templates for more information.
2. In the **Environment URL** field for the system, press the Down arrow key and select the URL of the system.
3. If you are using your own authentication profiles, in the **Authentication Profile** field for the system, press the Down arrow key and select the authentication profile that you want to communicate from the orchestration server to the system.
    - By default, the fields are populated with the DMAppAdmin authentication profile.

Note: The pipeline model screen will support the addition of more than four stages.

Repositories

1. In the Artifact management section, specify the development and production repositories through which the product rule that contains application contents moves through the pipeline.
2. In the **Development repository** field, select the development repository.
3. In the **Production repository** field, select the production repository.
4. Click **Next**.





Configuring a new pipeline - Model Pipeline

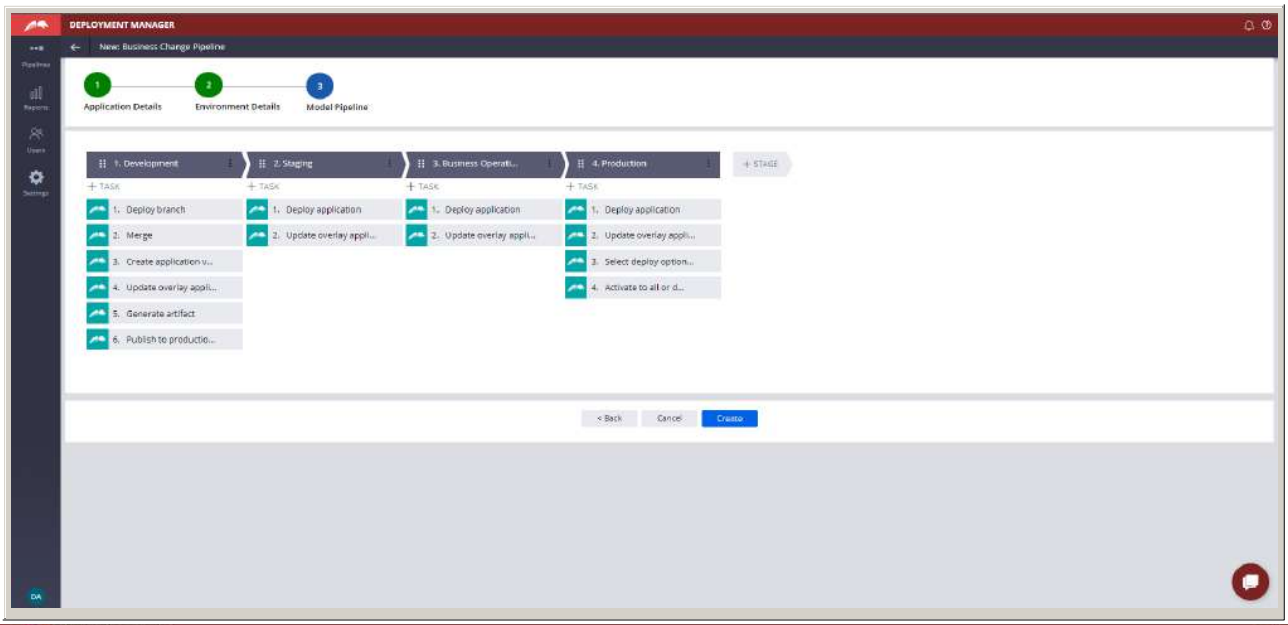
From the model pipeline screen, you can tailor your stages and tasks to replicate the quality standard of the release process defined by your organization. You can modify the deployment process here by configuring tasks and modifying stages.

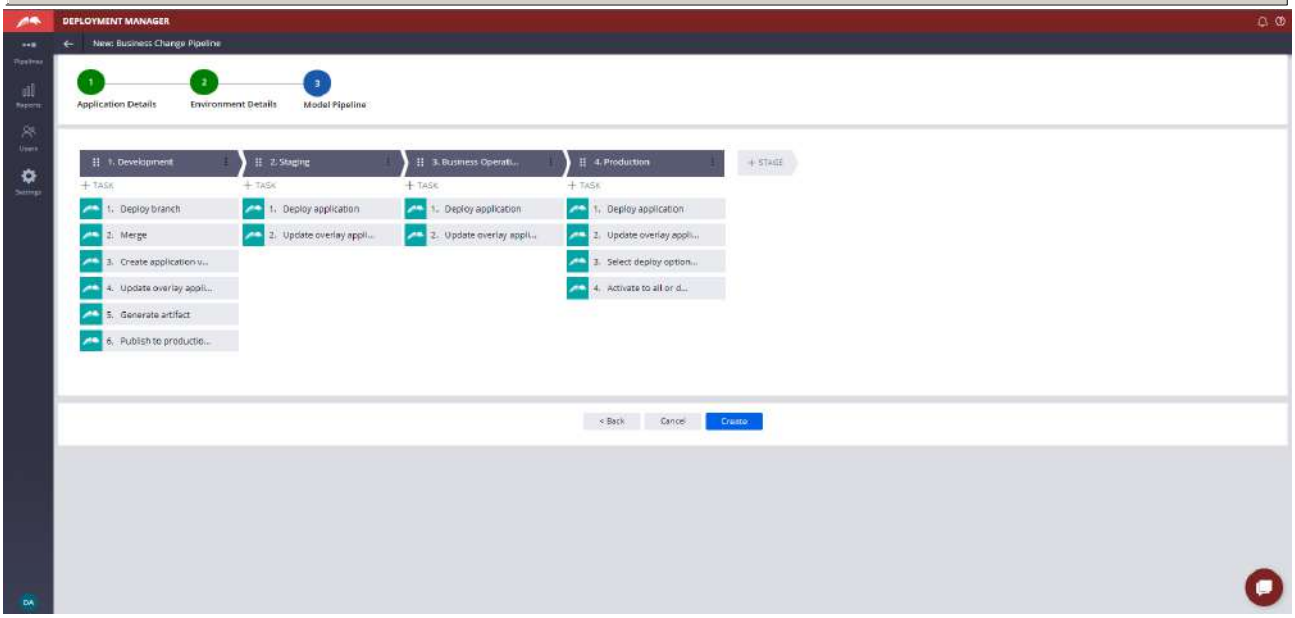
Business change pipelines offer a specific default use case, however this can be customized using the model pipeline screen.

For more information on adding, editing, or deleting a stage, see .

1. Stages can be modified here by clicking the **More** icon on each stage.
2. To add a stage, click the **+Stage** button. You can apply an environment template to each new stage. See Environment templates.

3. To add a task to a stage, click the **+Task** button within a stage. The available tasks that you can add are specific to the associated stage. The available tasks that you can add are specific to the associated stage. For a list of task related information, see the Task catalog.
    1. The environment template assigned to a stage drives the availability of tasks. Mandatory tasks cannot be deleted.



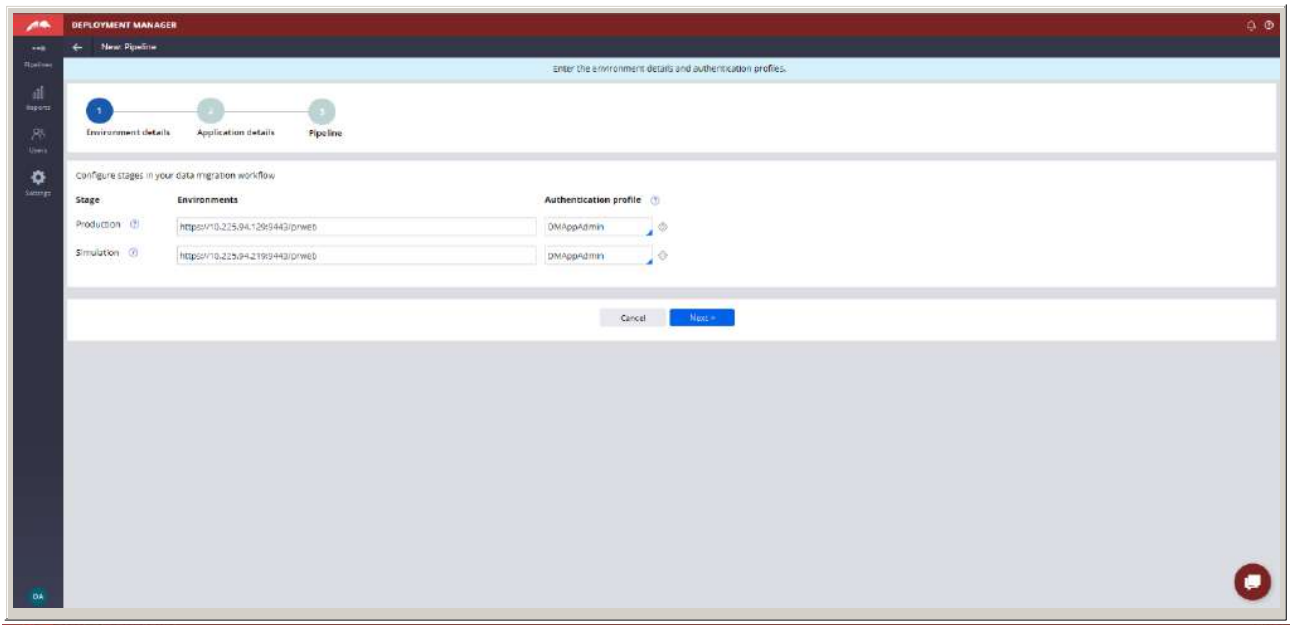


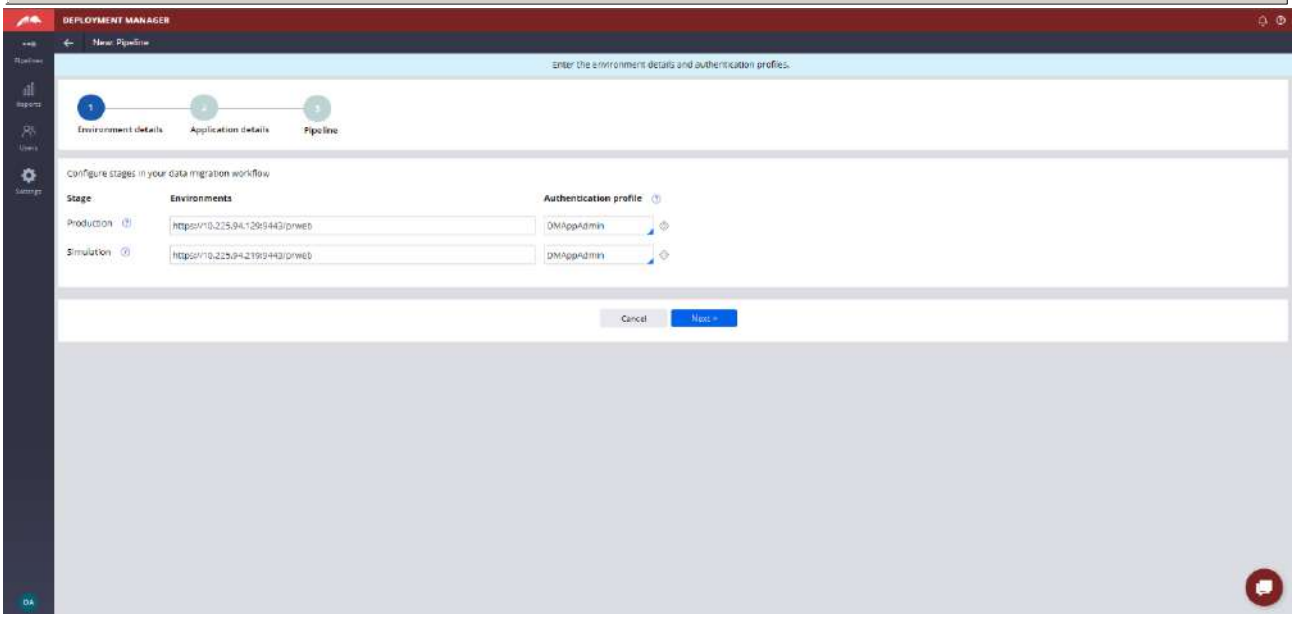
Creating a data migration pipeline

Create a pipeline by defining the production and simulation environments and the application details for the pipeline. By using a data migration pipeline, you can export and import simulation data automatically.

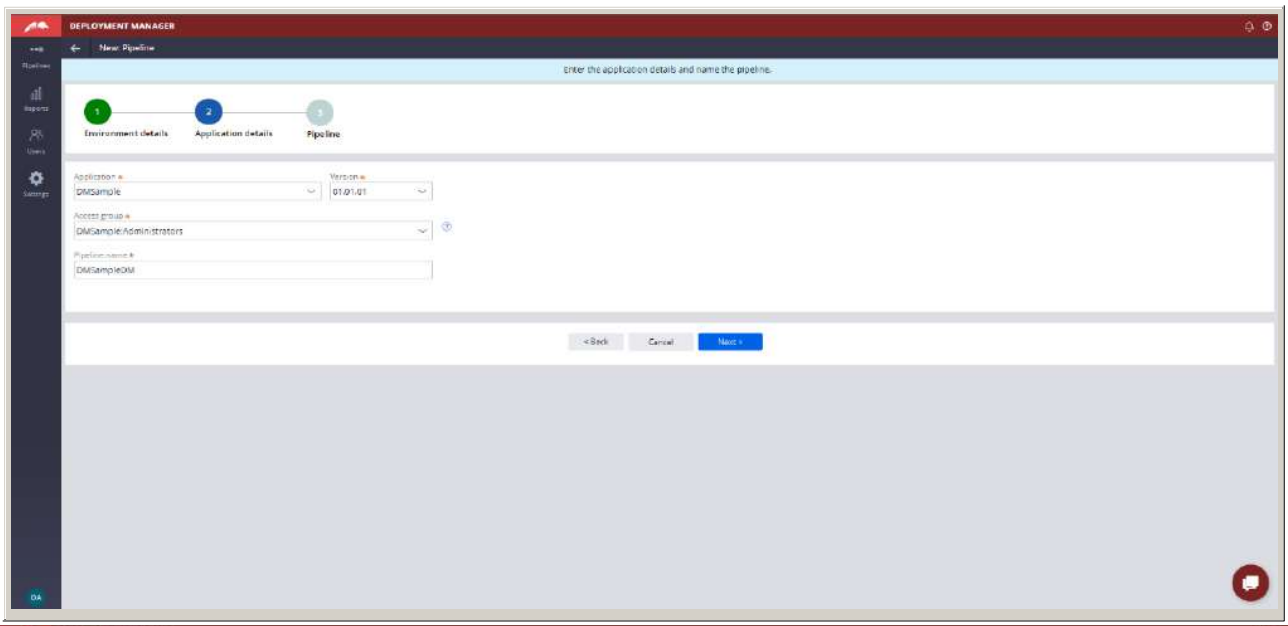
Do the following steps:

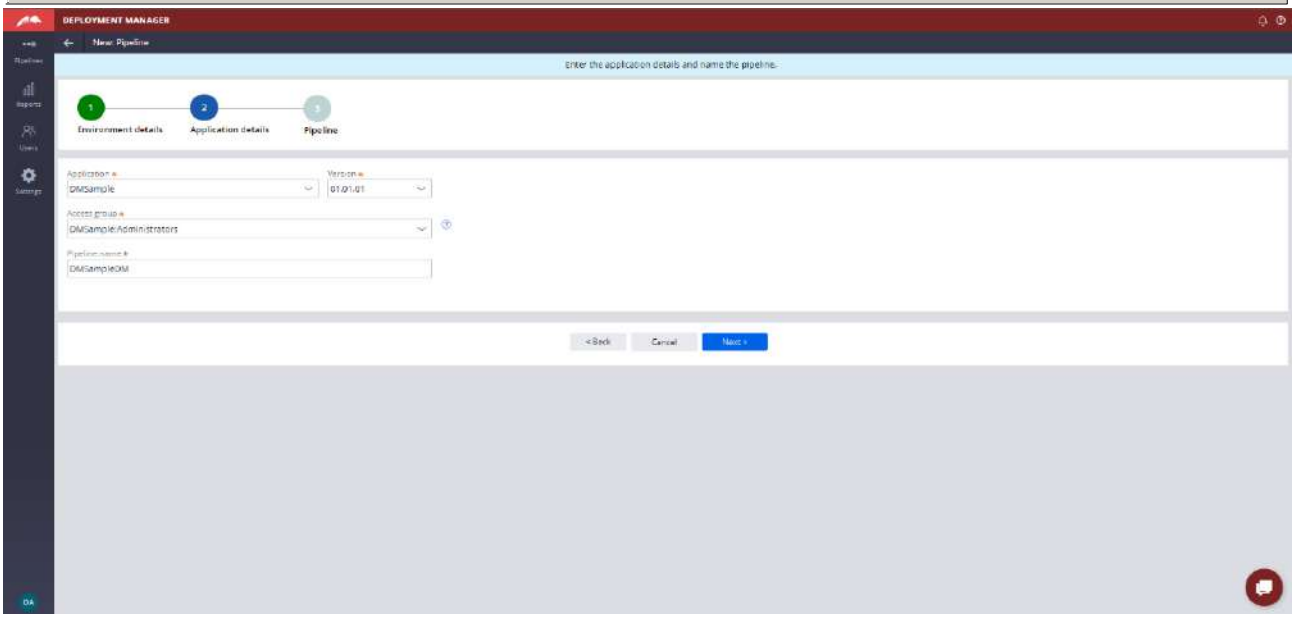
1. If the pipeline is not open, in the navigation pane of Deployment Manager, click **Pipelines Data migration pipelines** , and then click the name of the pipeline.
2. Click **New**.
3. On the **Environment Details** page, if you are using Deployment Manager on-premises, configure environment details.
   This information is automatically populated if you are using Deployment in Pega Cloud Services environments, but you can change it.
    a. In the **Environment** fields, enter the URLs of the production and simulation environments.
    b. If you are using your own authentication profiles, from the **Auth profile** lists, select the authentication profiles that you want the orchestration server to use to communicate with the production and simulation environments.
    c. Click **Next**.





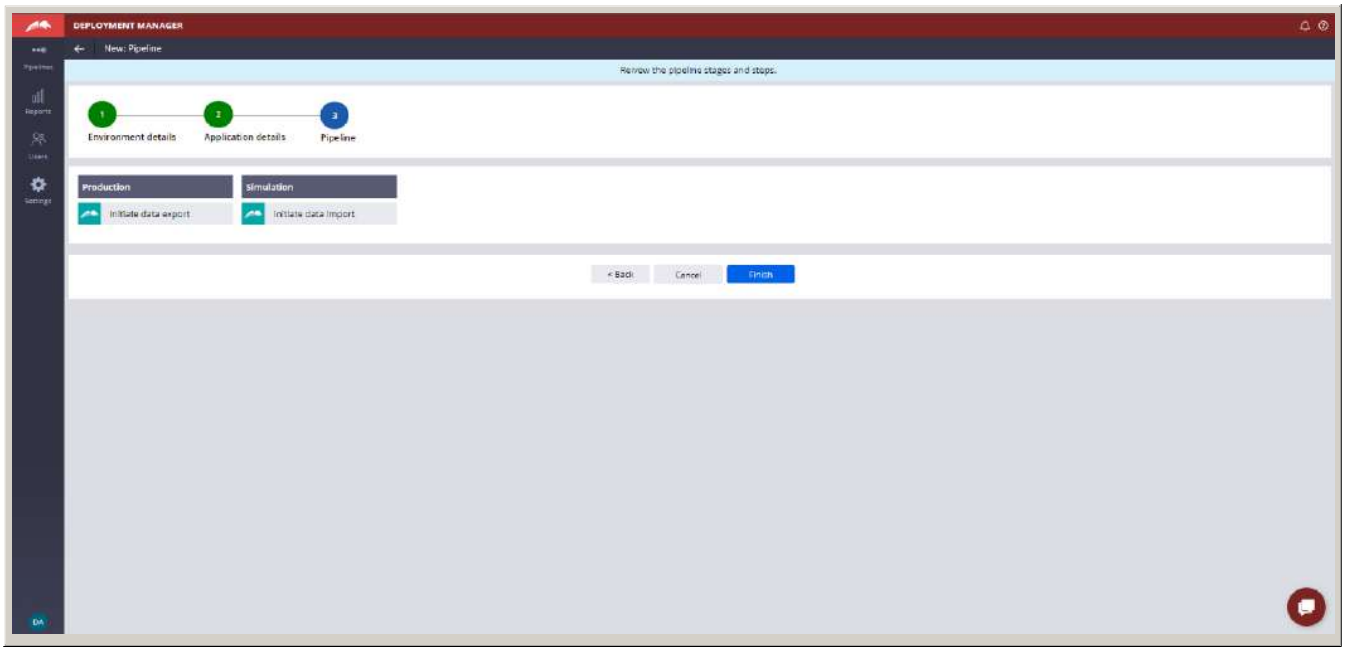
4. On the **Application details** page, specify the application information for which you are creating the pipeline.
    a. From the **Application** list, select the name of the application.
    b. From the **Version** list, select the application version.
    c. From the **Access group** list, select the access group for which you want to run pipeline tasks. This access group must be present on the production and simulation environments and have at least the *sysadmin4* role.
    d. In the **Name of the pipeline** field, enter the pipeline name.
    e. Click **Next**.

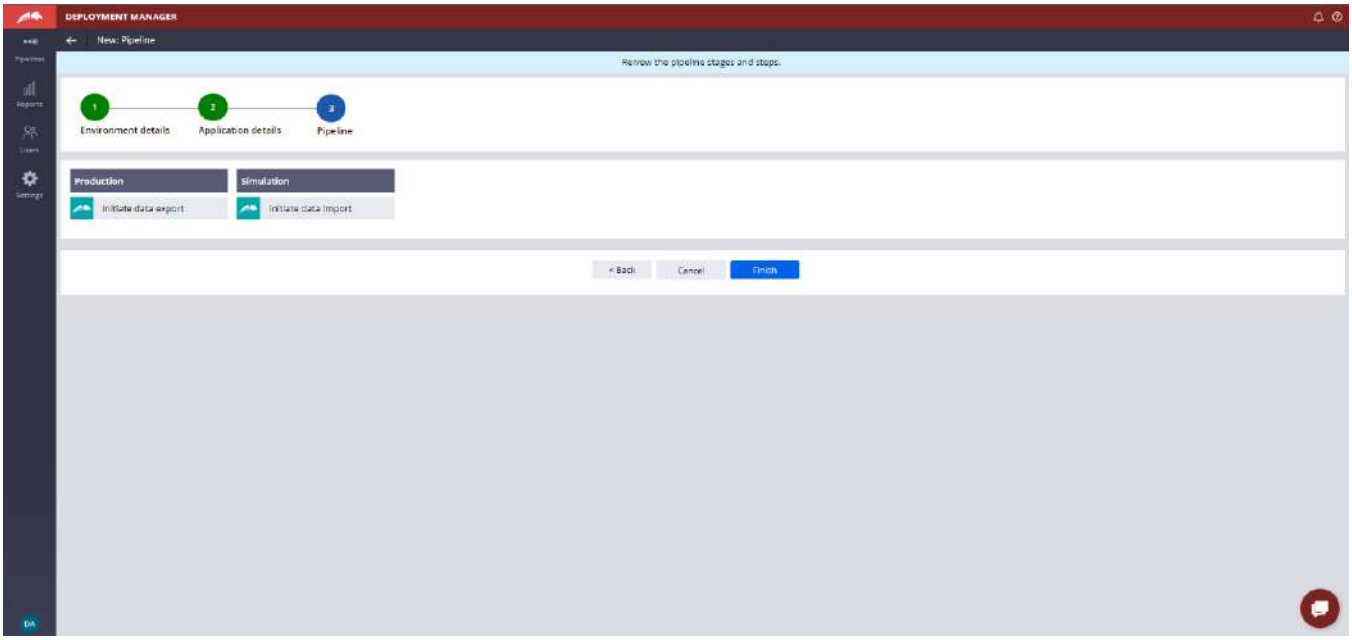




Result: The **Pipeline** page displays the stages and tasks, which you cannot delete, that are in the pipeline.

5. Click **Finish**.





Configuring an application pipeline in 5.1.x

When you add a pipeline, you specify merge criteria and configure stages and steps in the continuous delivery workflow. For example, you can specify that a branch must be peer-reviewed before it can be merged, and you can specify that Pega unit tests must be run after a branch is merged and is in the QA stage of the pipeline.

You can create multiple pipelines for one version of an application. For example, you can use multiple pipelines to use parallel development and hotfix lifecycles for your application.

Note: If you are using Deployment Manager for a release pipeline, the development or applicable stage responsible for artifact should not be at a higher version than the candidates where the artifact is planned for deployment. All candidate environments should have same versions of Pega platform.

Application pipeline compatibility

Candidate platform version	Orchestrator 4.8.4	Orchestrator 5.x
v8.1 - 8.5.1	Supported with PDF 4.8.4	Supported with PDF 4.8.4
v8.5.2+		Supported with PDF 5

- Modifying an application pipeline

  You can modify the details of your pipeline, such as configuring tasks, updating the repositories that the pipeline uses, and modifying the URLs of the systems in your environment. You cannot modify information if your pipeline is running.

- Pipeline templates

  When you create a pipeline, Deployment Manager prompts you to categorize your pipeline and apply one of three templates. These pipeline templates help you structure pipelines for their ultimate purpose, as well separate business needs for better organizational efficiency.

- Environment templates

  Environment templates are applied during stage creation and determine which tasks are added to a stage by default. Default stages and their associated environment templates are automatically added to your pipeline at the time of pipeline creation based on the type of pipeline created (merge, deployment, business change). You can modify the tasks for each stage on the Pipeline model screen.

- Task catalog

  Deployment Manager supports a range of tasks to support the different application deployment scenarios for Pega powered applications. Tasks relevant to a use case are structured by environment templates and pipeline templates.

- Creating and using custom tasks

  Deployment Manager supports a range of tasks to support the different application deployment scenarios for Pega powered applications out-of-the-box. Although comprehensive, the default task configurations cannot account for

every use case for every customer. The orchestrator-based custom task functionality allows you to create unique tasks specific to your business needs which can be included in pipelines. Any task not shipped with Deployment Manager is considered to be a custom task.

- Using multispeed deployments in 5.1.x

  Achieve greater flexibility over your pipeline workflow through multiple deployments that queue in the same stage of a pipeline. With this functionality, developers can regularly merge their changes for selective testing and promotion of deployments without blocking the pipeline. This approach introduces the concept of a transition between stages requiring user input to proceed.

- Archiving and activating pipelines

  If your role has the appropriate permissions, you can archive inactive pipelines so that they are not displayed on the Deployment Manager landing page.

- Disabling and enabling a pipeline

  If your role has the appropriate permissions, you can disable a pipeline on which errors continuously cause a deployment to fail. Disabling a pipeline prevents branch merging, but you can still view, edit, and stop deployments on a disabled pipeline.

- Managing artifacts generated by Deployment Manager

  You can view, download, and delete application packages in repositories that are on the orchestration server. If you are using Deployment Manager on Pega Cloud Services, application packages that you have deployed to cloud repositories are stored on Pega Cloud Services. To manage your cloud storage space, you can download and permanently delete the packages.

- Deleting a pipeline

  If your role has the appropriate permission, you can delete a pipeline. When you delete a pipeline, its associated application packages are not removed from the repositories that the pipeline is configured to use.

Modifying an application pipeline

You can modify the details of your pipeline, such as configuring tasks, updating the repositories that the pipeline uses, and modifying the URLs of the systems in your environment. You cannot modify information if your pipeline is running.

- Modifying application details

  You can modify application details, such as the product rule that defines the content of the application that moves through the pipeline.

- Modifying URLs and authentication profiles in 5.1.x

  You can modify the URLs of your development and candidate systems and the authentication profiles that are used to communicate between those systems and the orchestration server.

- Modifying repositories

  You can modify the development and production repositories through which the product rule that contains application contents moves through the pipeline. All the generated artifacts are archived in the Development repository, and all the production-ready artifacts are archived in the Production repository.

- Modifying stages and tasks in the pipeline

  You can modify the stages and the tasks that are performed in each stage of the pipeline. For example, you can skip a stage or add tasks such as Pega unit testing to be done on the QA stage.

- Modifying merge options for branches in 5.1.x

  If you are using branches in your application, specify options for merging branches into the base application.

Modifying application details

You can modify application details, such as the product rule that defines the content of the application that moves through the pipeline.

Do the following steps:

1. If the pipeline is not open, in the navigation pane of Deployment Manager, click **Pipelines** and then click the name of the pipeline.
2. Click **Actions Edit pipeline** .
3. Click **Application details**.

4. Specify the environment URL in the **Application packaging environment** field, and select or enter the authentication profile for connecting to the application packaging environment.
5. In the **Application** field, press the Down arrow key and select the name of the application.
6. In the **Version** field, press the Down arrow key and select the application version.
7. In the **Access group** field, press the Down arrow key and select the access group for which pipeline tasks are run.
    - This access group must be present on all candidate systems and have at least the sysadmin4 role.
8. In the **Product rule** field, enter the product rule that defines the contents of the application. If you are using a separate product rule to manage test cases, complete the following steps in the Application test cases section.
    a. Select **Deploy test applications** check box to deploy test cases.
    b. In the **Test application** field, enter the name of the test application.
    c. In the **Version** field, enter the version of the test case product rule.
    d. In the **Access group** field, enter the access group for which test cases are run.
    e. In the **Product rule** field, enter the name of the test case product rule.
9. In the **Version** field, enter the product rule version.
10. If the application depends on other applications, add those application in the **Dependencies** section.
    a. To configure dependent applications, click **Dependencies**.
    b. Click **Add**.
    c. In the **Application name** field, press the Down arrow key and select the application name.
    d. In the **Application version** field, press the Down arrow key and select the application version.
    e. In the **Pipeline** field, press the Down arrow key and select the pipeline.
    f. In the **Deployment** field, press the Down arrow key and select the deployment that contains the production-ready artifact of the dependent application.
    g. If you want the latest artifact of the depended application to be automatically populated, select **latest**.
    For more information, see Listing product dependencies.
11. Click **Save.**
12. Run diagnostics to verify that your pipeline is configured correctly.
    For more information, see Diagnosing a pipeline in 5.1.x.

Modifying URLs and authentication profiles in 5.1.x

You can modify the URLs of your development and candidate systems and the authentication profiles that are used to communicate between those systems and the orchestration server.

Do the following steps:

1. If the pipeline is not open, in the navigation pane of Deployment Manager, click **Pipelines** and then click the name of the pipeline.
2. Click **Actions Edit pipeline** .
3. In the pipeline model window, click the stage for which you want to modify the URL or authentication profile. Under stage details, in the **Environment URL** field for the system, enter the URL.
    - VPN and firewalls that add the orchestrator to a list of trusted IP ranges must support two-way communication.
    - Environments with URL redirects are not supported, and the URL used for the environment should not have redirection enabled.
4. In the **Authentication Profile** field for the system, press the Down arrow key and select the authentication profile that you want to communicate from the orchestration server to the system.
5. Click **Save**.
6. Run diagnostics to verify that your pipeline is configured correctly.
    For more information, see Diagnosing a pipeline in 5.1.x.

Modifying repositories

You can modify the development and production repositories through which the product rule that contains application contents moves through the pipeline. All the generated artifacts are archived in the development repository, and all the production-ready artifacts are archived in the production repository.

You do not need to configure repositories if you are using Pega Cloud Services; you can use different repositories other than the default ones that are provided.

1. If the pipeline is not open, in the navigation pane of Deployment Manager, click **Pipelines** and then click the name of the pipeline.
2. Click **Actions Edit pipeline** .
3. Click **Artifact management**.
4. If you are looking to modify the current or default repositories, complete the following tasks:
    a. In the **Application repository** section, in the **Development repository** field, press the Down arrow key and select the development repository
    b. In the **Production repository** field, press the Down arrow key and select the production repository.
5. Click **Save**.
6. Run diagnostics to verify that your pipeline is configured correctly.
    For more information, see Diagnosing a pipeline.

Modifying stages and tasks in the pipeline

You can modify the stages and the tasks that are performed in each stage of the pipeline. For example, you can skip a stage or add tasks such as Pega unit testing to be done on the QA stage.

Stages

Version 5 of Deployment Manager supports more than four stages in a pipeline. If you are modifying an existing pipeline, you can perform the following steps for the desired results.

Add a new stage

1. If the pipeline you wish to modify is not open, then click on **Pipeline Application Pipeline.**
2. Click the pipeline name.
3. From the **Actions** menu, click **Edit pipeline**. Ensure there are no active deployments in progress or awaiting action.
4. Click **Pipeline model**.
5. On this page, click the **+Stage** icon on the right of the flow model.
6. Add a name to the stage.
7. On the stage details section, add the mandatory parameters. For more information on the Environment template, see Environment templates.
8. To enable multispeed deployment, select **Wait for user action** under the section labeled **When all tasks in this stage are complete**. For more information on multispeed deployment, see Using multispeed deployments in 5.1.x.
9. Click **Save**.

Delete a stage

1. If the pipeline you wish to modify is not open, then click on **Pipeline Application Pipeline.**
2. Click the pipeline name.
3. From the **Actions** menu, click **Edit pipeline**. Ensure there are no active deployments in progress or awaiting action.
4. Click **Pipeline model**.
5. On this page, click the stage you wish to delete.
6. Click the **More** icon on the stage header.
7. Click **Delete stage**.
8. Click **Save**.

Tasks

Deployment Manager supports a range of tasks to support the different application deployment scenarios for Pega powered applications. Tasks relevant to a use case are structured by the following types:

- Environment template: When the environment template on a stage is configured to anything other than **custom**, recommended tasks are populated for the environment template. A list of available tasks for the stage provides all tasks that can be added as necessary. When the environment template for a stage is custom, no recommendation or restriction applies to a stage. Users can choose any tasks from the tasks available in the system.
- Pipeline template: Pipeline templates drive the range of tasks available though the environment type. Adding a task to a stage requires you to decide the step on which you want to add a stage and then follow the following steps:

    1. In the navigation pane of Deployment Manager, click **Pipelines Application pipelines** , and then click the name of the pipeline.
    2. Select the pipeline name. Click **Actions Edit pipeline** .
    3. On the pipeline model screen, identify the position where you need to add the task.
    4. Click the **more** icon on the task and click **Add task below**.
    5. From the task menu, select the task and click **Select**.
    6. Update the required task details.
    7. Click **Save**.

For more information on available tasks, refer to the Task catalog.

See below for a few different scenarios on using tasks to achieve different outcomes.

Scenario: Mandate a Guardrail compliance score of 100, with branch review before performing a merge.

1. If the Merge pipeline is not open, in the navigation pane of Deployment Manager, click **Pipelines Application pipelines** , and then click the name of the pipeline. You can also filter on the pipeline template to narrow the search.
2. Click **Pipeline model**.
3. To specify that a branch must meet a compliance score of 100 before it can be merged:
    1. In the Development stage, click **Check branch guardrail**.
    2. In the **Input parameters**, add the desired score of 100 in the **Threshold guardrail score.**
    3. Click **Save**.

    For more information about compliance scores, see Check guardrail compliance.
4. To specify that a branch must be reviewed before it can be merged:
    1. In the Development stage, click **More** on **Merge task**.
    2. Click **Add Task Above**.
    3. From the task list, select **Check branch review status**.
    4. Click **Select.**

5. Click **Save**.

For more information about branch reviews, see Branch reviews.

Scenario: Start a deployment after a merge is performed.

1. If the Merge pipeline is not open, in the navigation pane of Deployment Manager, click **Pipelines Application pipelines** , and then click the name of the pipeline. You can also filter on the pipeline template to narrow the search.
2. Click **Pipeline model**.
3. To start a deployment automatically when a branch is merged:
    1. In the Development stage, click **More** on the Merge task.
    2. Select **TriggerDeployment** from the available task options.
    3. Click **Select**.
    4. In the task details panel, provide the **Deployment pipeline ID**.
    5. Click **Save**.

Modifying merge options for branches in 5.1.x

If you are using branches in your application, specify options for merging branches into the base application.

Do the following steps:

1. If the pipeline is not open, in the navigation pane of Deployment Manager, click **Pipelines** and then click the name of the pipeline.
2. Click **Actions Edit pipeline** .
3. Click **Merge policy**.
4. If you are not using branches, click the **No** radio button, and then go to step 6.
5. If you are using branches, do the following actions:
    a. Click **Yes**.
    b. Do one of the following actions:
    - To merge branches into the highest existing ruleset in the application, click **Highest existing ruleset**.
    - To merge branches into a new ruleset, click **New ruleset**.
    a. In the **Password** field, enter the password that locks the rulesets on the development system.
6. Click **Save**.
7. Run diagnostics to verify that your pipeline is configured correctly.
   For more information, see Diagnosing a pipeline in 5.1.x.

Pipeline templates

When you create a pipeline, Deployment Manager prompts you to categorize your pipeline and apply one of three templates. These pipeline templates help you structure pipelines for their ultimate purpose, as well separate business needs for better organizational efficiency.

Learn more about the different pipeline templates and their intended purposes.

Merge pipeline

Use a merge pipeline to merge any application changes to the system of record (SOR) as part of the continuous integration (C/I) DevOps methodology. Merge pipelines apply specific quality checks and guardrails to ensure that any changes that you want to merge have no negative impact on the SOR. If all merge criteria are successfully met, deployment can be triggered directly from the merge pipeline by adding a **Trigger deployment** task.

Deployment pipeline

Deployment pipelines generate artifacts from the SOR and deliver them through the various stages of the workflow. This pipeline template aligns with the continuous delivery (CI/CD) DevOps methodology, and deploys predictable, high-quality releases without using third-party tools.

Business change pipeline

A business change pipeline is a new pipeline type to support everyday, business-as-usual changes to your application. This pipeline template merges updates and generates a new application that exposes the latest changes. With this functionality, you can test the application with a small subset of users before activation, as well as roll back in the event of errors. Business change pipelines facilitate changing requirements by modifying and deploying application rules in a controlled manner.**Note:** Business change pipelines in Deployment Manager 5.1 are supported only on Pega Platform version 8.5.1 and later.

Upgrade pipeline

Upgrade pipelines automate and simplify near-zero-downtime upgrades to promote the latest application functionality to your production environment. By using Deployment Manager as a standard DevOps approach during upgrades, you can assess the effect of the latest Pega Platform capabilities and plan your new feature adoption strategy across your

environments after you upgrade to the latest version of Pega Infinity.

Environment templates

Environment templates are applied during stage creation and determine which tasks are added to a stage by default. Default stages and their associated environment templates are automatically added to your pipeline at the time of pipeline creation based on the type of pipeline created (merge, deployment, business change). You can modify the tasks for each stage on the **Pipeline model** screen.

See the tables below for the various environment templates associated with each pipeline, and the task list available in each stage.

*Denotes a mandatory task that cannot be deleted

Deployment pipeline

Deployment pipeline

Environment type	Default Tasks
Development	• Generate artifact*
Production	• Deploy application*
QA	• Deploy application* • Check guardrail compliance • Verify security checklist
Staging	• Deploy application* • Check guardrail compliance • Verify security checklist • Publish to production*
Custom	No default tasks populated, all tasks available. For a list of all available tasks, see Task catalog.

Merge pipeline

Merge pipeline

Environment type	Default Tasks
Dev_merge	• Get merge conflict* • Merge*
Custom	No default tasks populated, all tasks available. For a list of all available tasks, see Task catalog.

Business change pipeline

Business change pipeline

Environment type	Default Tasks
Business operations	• Update overlay application • Deploy application
BusinessChange_Development	• Deploy branch • Generate artifact • Merge • Update overlay application

Environment type	Default Tasks
BusinessChange_Production	• Deploy application • Update overlay application
BusinessChange_Sandbox	• Deploy application • Update overlay application
Custom	No default tasks populated, all tasks available. For a list of all available tasks, see Task catalog.

Upgrade pipeline

Upgrade pipeline

Environment type	Default Tasks
Staging Clone	• Generate upgrade artifact
Ephemeral	• Enable DMAppAdmin, Test operators and set passwords • Deploy upgrade fixes • Publish to production • Go approval
Production	• Confirm upgrade readiness • Deploy upgrade fixes
Staging	• Confirm upgrade readiness • Deploy upgrade fixes
Quality Assurance	• Confirm upgrade readiness • Deploy upgrade fixes
Development	• Confirm upgrade readiness • Deploy upgrade fixes

Task catalog

Deployment Manager supports a range of tasks to support the different application deployment scenarios for Pega powered applications. Tasks relevant to a use case are structured by environment templates and pipeline templates.

See below for the two different task structures:

- Environment template: When the environment template on a stage is configured to anything other than **custom**, recommended tasks are populated for the environment template. A list of available tasks for the stage provides all tasks that can be added as necessary. When the environment template for a stage is custom, no recommendation or restriction applies to a stage. Users can choose any tasks from the tasks available in the system.
- Pipeline template: Pipeline templates drive the range of tasks available though the environment type. Adding a task to a stage requires you to decide the step on which you want to add a stage and then follow the following steps:
  1. In the navigation pane of Deployment Manager, click **Pipelines Application pipelines** , and then click the name of the pipeline.
  2. Select the pipeline name. Click **Actions Edit pipeline** .
  3. On the pipeline model screen, identify the position where you need to add the task.
  4. Click the **more** icon on the task and click **Add task below**.
  5. From the task menu, select the task and click **Select**.
  6. Update the required task details.
  7. Click **Save**.

- Generate artifact

Exports the artifact as defined in the product rule and publishes to the development repository.

- Deploy application

  Exports the artifact from the repository and deploys the application in the current stage environment.

- Check guardrail compliance

  This task returns an error if guardrail compliance for this application is less than the compliance score. As a best practice, Pega recommends a compliance score of 97 for highly performing applications.

- Check branch guardrail

  This task returns an error if guardrail compliance for the branch associated with the deployment is less than the compliance score. As a best practice, Pega recommends a compliance score of 97 for highly performing applications.

- Verify security checklist

  Prepares the application for a secure deployment.

- Publish to production

  Promotes the application artifacts to production repository.

- Get merge conflict

  Before a merge action is performed, this provides details on the conflicts from the changes in the specified branch identified by branch ID (input parameter).

- Merge

  This task merges the changes residing in the branch identified by branch ID (input parameter) to the SOR (System of Record) where the development changes are committed.

- Run pega units on branch

  Executes PegaUnit tests for the pipeline application. This task returns an error if the pass percentage is less than 100%.

- Deploy branch

  Exports the branch artifact from the repository and deploys the branch into the current stage environment.

- Check branch review status

  Ensures the review on the specified branch is closed.

- Perform manual step

  Pauses the deployment and assigns a manual task to the user or adds an approval requirement from a stakeholder for a critical action.

- Update application on access group

  Update access groups with new application version.

- Trigger deployment

  Triggers a deployment of the specified application pipeline.

- Run Jenkins task

  This task is executed on the Jenkins environment. Please ensure that it is properly configured on your Jenkins environment.

- Validate test coverage

  Stops the test coverage session at the application level and updates the quality metrics for the application. It throws an error either if the test coverage percentage is less than the defined test coverage percentage, or if a test coverage session is not in progress for this application.

- Enable test coverage

  This task starts an application-level test coverage session that captures which rules are covered and uncovered while the test gates are running. It provides an intermediate input option named "Restart Test Coverage" if the test coverage session is already in progress for this application. Clicking on this option will stop the active coverage session on candidate application and starts a new one.

- Run pega unit tests

  Executes PegaUnit tests for the pipeline application or the application given by the access group. If the test suite ID is configured, tests in that test suite are executed. Otherwise, all PegaUnit tests for the application are executed. This task returns an error if the pass percentage is less than 100%.

- Refresh application quality

  Regenerates the application quality metrics.

- Update overlay application

  This task is only used in Business change pipelines, and it works in conjunction with the NewApplication task. It updates the built-on of the pipeline's application to be the one that was created by the NewApplication task

- Run pega scenario (sauce labs)

  Runs the application Pega scenario tests on Sauce Labs Testing tool. If the test suite ID is configured, this executes the tests in that test suite. Otherwise, this executes all scenario tests for the application. This task fails if the test pass rate is less than 100%.

- Run pega scenario (standalone selenium)

  Runs the application Pega scenario tests in a standalone-selenium provider. If the test suite ID is configured, this executes the tests in that test suite. Otherwise, this executes all scenario tests for the application. This task fails if the test pass rate is less than 100%.

- Run pega scenario (CBT)

  Runs the application Pega scenario tests on a cross-browser testing tool. If the test suite ID is configured, this executes the tests in that test suite. Otherwise, this executes all scenario tests for the application. This task fails if the test pass rate is less than 100%.

- Run pega scenario (browser stack)

  Runs the application Pega scenario tests in Browser Stack. If the test suite ID is configured, this executes the tests in that test suite. Otherwise, this executes all scenario tests for the application. This task fails if the test pass rate is less than 100%.

- New application

  Creates a new application version.

Generate artifact

Exports the artifact as defined in the product rule and publishes to the development repository.

This task is available in Development and Business change pipelines.

Parameters

Input parameters

There are no input parameters for this task type.

Output parameters

Output parameters

Parameter	Type	Description
Application Archive ID	Text	The archive resource ID for the pipeline application.
Test Application Archive ID	Text	The archive resource ID for the test application configured in the pipeline .

Deploy application

Exports the artifact from the repository and deploys the application in the current stage environment.

This task is available in Development and Business change pipelines.

Parameters

Input parameters

Input parameters

Parameter	Type	Description
Deploy test artifact?	True/False	Select to deploy the test application configured in the pipeline
Deploy production ready artifacts?	True/False	Select to deploy the production ready artifacts
Default action on aged update?	Text	Sets the default method for handling aged updates. If 'override' is specified, any aged update reported during deployment refreshes with the rules in artifact. If 'skip' is specified, the aged updates are not refreshed with the rules in artifact. Blank or any other value defaults to a manual interaction if a deployment encounters aged updates.

Output parameters

Output parameters

Parameter	Type	Description
Restore point	Text	Restore point created as part of the deploy task
Application Records Moved Count	Integer	Records moved for the application
Application New Instances	JSON	New instances deployed for the application
Application Updated Instances	JSON	List of instances updated part of deploy task
Application Draft Flows	JSON	List of draft flows in the RAP
Test Application Records Moved Count	Integer	Records moved for the test application
Test Application New Instances	JSON	New instances deployed for the test application
Test Application Updated Instances	JSON	List of instances updated part of deploy task
Application aged updates	JSON	Returns a list of aged updates identified during the deployment on a target candidate system

Check guardrail compliance

This task returns an error if guardrail compliance for this application is less than the compliance score. As a best practice, Pega recommends a compliance score of 97 for highly performing applications.

This task is available in Development and Business change pipelines.

Parameters

Input parameters

Input parameters

Parameter	Type	Description
Threshold Guardrail Score	Decimal	This score will assess if the application is guardrail compliant

Output parameters

Output parameters

Parameter	Type
Actual Guardrail Score	Text
Compliant Rules Percentage	Decimal
Total Rules	Decimal
Rules With Warnings	Integer
Rules With Unjustified Warnings	Integer
Total Warnings	Integer
Severe Warnings	Integer
Moderate Warnings	Integer
Informational Warnings	Integer

Check branch guardrail

This task returns an error if guardrail compliance for the branch associated with the deployment is less than the compliance score. As a best practice, Pega recommends a compliance score of 97 for highly performing applications.

This task is available in Merge pipelines.

Parameters

Input parameters

Input parameters

Parameter	Type	Description
Threshold Guardrail Score	Decimal	This score will assess if the application is guardrail compliant

Output parameters

Output parameters

Parameter	Type
Actual Guardrail Score	Text
Total Rules	Decimal

Parameter	Type
Rules With Test Support Count	Decimal
Rules With Warnings	Integer
Rules With Unjustified Warnings	Integer
Total Test Case Count	Integer
Failed Test Cases Count	Integer
Passed Test Cases Count	Integer
Test Coverage Percent	Decimal

Verify security checklist

Prepares the application for a secure deployment.

This task is available in Development and Business change pipelines.

Parameters

There are no parameters available for this task type.

Publish to production

Promotes the application artifacts to production repository.

Parameters

Input parameters

Input parameters

Parameter	Type	Description
Include test artifact?	True/False	Select to include the test artifacts in production repository

Output parameters

Output parameters

Parameter	Type	Description
Application Archive ID	Text	The archive resource ID for the pipeline application
Test Application Archive ID	Text	The archive resource ID for the test application configured in the pipeline

Get merge conflict

Before a merge action is performed, this provides details on the conflicts from the changes in the specified branch identified by branch ID (input parameter).

This task is only available in merge pipelines.

Parameters

Input parameters

Input parameters

Parameter	Type	Description
Branch ID	Text	Identifier of the branch being merged

Output parameters

Thee ae no output parameters available for this task.

Merge

This task merges the changes residing in the branch identified by branch ID (input parameter) to the SOR (System of Record) where the development changes are committed.

This task is available in Merge and Business change pipelines.

Parameters

Input parameters

Input parameters

Parameter	Type	Description
Branch ID	Text	Identifier of the branch being merged

Output parameters

There are no output parameters for this task type.

Run pega units on branch

Executes PegaUnit tests for the pipeline application. This task returns an error if the pass percentage is less than 100%.

This task is available in Merge pipelines.

Parameters

Input parameters

There are no input parameters available for this task.

Output parameters

Output parameters

Parameter	Type	Description
Failed Test Cases	Text	Test cases failed during Pega Units task execution
Pega Units xUnit Response	Text	PegaUnits execution result in xUnit format

Deploy branch

Exports the branch artifact from the repository and deploys the branch into the current stage environment.

This task is available in Merge and Business change pipelines.

Parameters

Input parameters

There are no input parameters available for this task.

Output parameters

Output parameters

Parameter	Type	Description
RecordsMovedCount	Text	Count of records imported as part of branch deploy
TotalRecordsCount	Decimal	Count of records included in the branch
NewInstances	Decimal	New instances deployed as part of branch deployment
UpdatedInstances	Integer	Existing instance updated as part of branch deploy

Check branch review status

Ensures the review on the specified branch is closed.

This task is available in Merge pipelines.

Parameters

Input parameters

Input parameters

Parameter	Type	Description
Branch ID	Text	Identifier of the branch being merged

Output parameters

There are no input parameters available for this task.

Perform manual step

Pauses the deployment and assigns a manual task to the user or adds an approval requirement from a stakeholder for a critical action.

This task is available in Merge, Development and Business change pipelines.

Parameters

Input parameters

Input parameters

Parameter	Type	Description
Assign To	Text	This is used to assign the deployment for approval to the given operator/Email

Output parameters

There are no output parameters available for this task.

Update application on access group

Update access groups with new application version.

This task is available in Merge, Development and Business change pipelines.

Parameters

Input parameters

There are no output parameters available for this task.

Output parameters

There are no output parameters available for this task.

Trigger deployment

Triggers a deployment of the specified application pipeline.

This task is available in Development and Business change pipelines.

Parameters

Input parameters

Input parameters

Parameter	Type	Description
Pipeline ID	Text	ID of the pipeline to be triggered.

Output parameters

Output parameters

Parameter	Type	Description
Triggered Deployment ID	Text	ID of the deployment triggered.

Run Jenkins task

This task is executed on the Jenkins environment. Please ensure that it is properly configured on your Jenkins environment.

This task is available in Development and Business change pipelines.

Parameters

Input parameters

Input parameters

Parameter	Type	Description
Jenkins server URL	Text	ID of the pipeline to be triggered.
Authentication Profile	Text	Authentication profile
Job Name	Text	Jenkins job name
Token	Text	Jenkins token
Parameters	Text	Jenkins parameter

Output parameters

Output parameters

Parameter	Type	Description
Build Number	Text	Build number

Validate test coverage

Stops the test coverage session at the application level and updates the quality metrics for the application. It throws an error either if the test coverage percentage is less than the defined test coverage percentage, or if a test coverage session is not in progress for this application.

This task is available in Development and Business change pipelines.

Parameters

Input parameters

Input parameters

Parameter	Type	Description
Threshold Coverage Percentage	Decimal	On the Test coverage landing page, you can generate user-level and application-level test coverage reports by running test coverage reports to identify how many executable rules in your application are covered by tests. Application-level reports contain results from multiple coverage sessions that are performed by multiple users.

Output parameters

There are no output parameters available for this task.

Enable test coverage

This task starts an application-level test coverage session that captures which rules are covered and uncovered while the test gates are running. It provides an intermediate input option named "Restart Test Coverage" if the test coverage session is already in progress for this application. Clicking on this option will stop the active coverage session on candidate application and starts a new one.

This task is available in Development and Business change pipelines.

Parameters

Input parameters

Input parameters

Parameter	Type	Description
Start new coverage session	True/false	Select to abort any running coverage sessions and start a new one every time.

Output parameters

There are no output parameters available for this task.

Run pega unit tests

Executes PegaUnit tests for the pipeline application or the application given by the access group. If the test suite ID is configured, tests in that test suite are executed. Otherwise, all PegaUnit tests for the application are executed. This task returns an error if the pass percentage is less than 100%.

This task is available in Development and Business change pipelines.

Parameters

Input parameters

Input parameters

Parameter	Type	Description
Test Suite ID	Text	pxInsName of Test Suite to be executed
Access Group	Text	Access Group to be used for executing Pega Units

Output parameters

Output parameters

Parameter	Type	Description
Failed Test Cases	Text	pxInsName of Test Suite to be executed
Pega Units xUnit Response	Text	PegaUnits execution result in xUnit format

Refresh application quality

Regenerates the application quality metrics.

This task is available in Development and Business change pipelines.

Parameters

Input parameters

There are no input parameters available for this task.

Output parameters

There are no output parameters available for this task.

Update overlay application

This task is only used in Business change pipelines, and it works in conjunction with the NewApplication task. It updates the built-on of the pipeline's application to be the one that was created by the NewApplication task

This task is available in Business change pipelines.

Parameters

Input parameters

There are no output parameters available for this task.

Output parameters

There are no output parameters available for this task.

Run pega scenario (sauce labs)

Runs the application Pega scenario tests on Sauce Labs Testing tool. If the test suite ID is configured, this executes the tests in that test suite. Otherwise, this executes all scenario tests for the application. This task fails if the test pass rate is less than 100%.

This task is available in Development and Business change pipelines.

Parameters

Input parameters

Input parameters

Parameter	Type
User name	Text
Password	Password
Test Suite ID	Text
Provider auth name	Text
Provider auth key	Password
Browser	Text
Browser Version	Text
Platform	Text
Screen Resolution	Text

Output parameters

There are no output parameters available for this task.

Run pega scenario (standalone selenium)

Runs the application Pega scenario tests in a standalone-selenium provider. If the test suite ID is configured, this executes the tests in that test suite. Otherwise, this executes all scenario tests for the application. This task fails if the test pass rate is less than 100%.

This task is available in Development and Business change pipelines.

Parameters

Input parameters

Input parameters

Parameter	Type
User name	Text
Password	Password
Test Suite ID	Text
Provider URL	Text
Browser	Text

Parameter	Type
Browser Version	Text
Platform	Text
Screen Resolution	Text

Output parameters

There are no output parameters available for this task.

Run pega scenario (CBT)

Runs the application Pega scenario tests on a cross-browser testing tool. If the test suite ID is configured, this executes the tests in that test suite. Otherwise, this executes all scenario tests for the application. This task fails if the test pass rate is less than 100%.

This task is available in Development and Business change pipelines.

Parameters

Input parameters

Input parameters

Parameter	Type
User name	Text
Password	Password
Test Suite ID	Text
Provider auth name	Text
Provider auth key	Password
Browser	Text
Browser Version	Text
Platform	Text
Screen Resolution	Text

Output parameters

There are no output parameters available for this task.

Run pega scenario (browser stack)

Runs the application Pega scenario tests in Browser Stack. If the test suite ID is configured, this executes the tests in that test suite. Otherwise, this executes all scenario tests for the application. This task fails if the test pass rate is less than 100%.

This task is available in Development and Business change pipelines.

Parameters

Input parameters

Input parameters

Parameter	Type
Username	Text
Password	Password
Test Suite ID	Text
Provider auth name	Text
Provider auth key	Password
Browser	Text
Browser Version	Text
Platform	Text
Screen Resolution	Text

Output parameters

There are no output parameters available for this task.

New application

Creates a new application version.

This task is available in business change pipelines.

Parameters

Input parameters

Input parameters

Parameter	Type	Description
Application name	Text	Name of the application

Output parameters

Output parameters

Parameter	Type	Description
New application version	Text	New application version
Current application version	Text	Current application version
Application name	Text	Name of the application

Creating and using custom tasks

Deployment Manager supports a range of tasks to support the different application deployment scenarios for Pega powered applications out-of-the-box. Although comprehensive, the default task configurations cannot account for every use case for every customer. The orchestrator-based custom task functionality allows you to create unique tasks specific to your business needs which can be included in pipelines. Any task not shipped with Deployment Manager is considered to be a custom task.

Note: A ruleset is required to associate a task definition with an implementation, which should be added to the application. Custom tasks are likely to be shared by multiple teams and reused where applicable, therefore it is recommended to create a dedicated component for each task.

Common use cases for custom tasks are:

- Updating project management systems with deployment results
- Customized notifications
- Additional automated testing capabilities

At this time, task executions are only supported on the orchestrator environment. It is possible to create tasks that interact with candidate environments through service APIs, but this should be done with caution as direct communication between the orchestrator and the candidate environments will be restricted in the future.

Tip: For assistance in authoring a task component in Deployment Manager 5.1 or 5.2, import the PDM Task Definition found here.

Create a task definition

A task definition establishes the characteristics of a task including its name, description, and parameters. The task definition determines how the task is presented in the Deployment Manager portal and how it should be configured while working with a pipeline model.

Define a task by creating an instance of the PegaDevOps-Data-Task class and associate the definition record to the task ruleset to ensure it is packaged alongside the implementation. Refer to the following table for the required properties for the task instance. **Note:** Properties marked with an asterisk (*) are mandatory.

Task definition properties

Property name	Data type	Purpose
TaskType*	Text	Alphanumeric identifier for the task type (no spaces).
Purpose	Text	Purpose of the task.
TaskLabel*	Text	Enter task label by default which appears in pipeline model.
TaskCategory	Text	Enter the task category like Package & Deploy, Quality Metrics, Versioning,
Description	Text	Explain in detail why this task is requires? what is processes? Etc.
Input parameters	Page List of class Embed-PegaDevOps-Parameter	This list contains input parameter required for task processing.
Output parameters	Page List of class Embed-PegaDevOps-Parameter	This list contains expected out parameters post task processing.
Supported environment types*	Page List of class Embed-PegaDevOps-Parameter	This list contains task supported environment templates. Pick values from PegaDevOps-Data-EnvironmentTemplate class

Implement the task runtime

The task runtime executes in the background when it has been queued up by an active deployment. The Deployment Manager agent handles polling active deployments and delegates to the appropriate task runtime when required.

To implement the task runtime, follow these steps and associate them to a ruleset included in the system runtime context:

1. Create an abstract class that inherits from the class Embed-PegaDevOps-TaskRuntime-.
   For example, a custom task that integrates with Bamboo named Embed-PegaDevOps-TaskRuntime-Bamboo.

2. Override the SetTaskInformation data transform in the newly created class. This data transform sets the .taskType property value to the TaskType from the task definition.
3. Override the RunTask activity. Pass the following required parameters to this activity. Use this activity to implement the task runtime.

Parameter	Data type
TaskPage	Page of class PegaDevOps-Int-Task
Orchestrator	String

Useful utilities for implementing task runtimes

The implementation contained within the RunTask activity is completely open-ended, however there are many utilities that hep make the process of implementing a task easier.

- If a task is successful, call the *ResolveTaskSuccess* activity and pass the following parameters:
  - **Task ID** - Value is available in *TaskPage* under the *taskID* property.
  - **Orchestrator URL** - Available as an input parameter to the *RunTask* activity.
- If a task fails, call the *ResolveTaskWithError* activity and pass the following parameters:
  - **Task ID** - Value is available in *TaskPage* under the *taskID* property.
  - **Orchestrator URL** - Available as an input parameter to the *RunTask* activity.
  - **Error message** - String with error message.

- To perform CRUD actions against a Deployment Manager resource:
  - Create a page of an integration class, for instance, PegaDevOps-Int-Deployment.
  - Call one of the action API activities on that page:
    - ResourceCreateInt
    - ResourceUpdateInt
    - ResourceDeleteInt
- If any REST API needs to be called, use the *D_ExecuteRestConnector* data page and pass the parameters below:
  - **MethodName** - Type of method used by the REST API (Post, Put, Delete, Get, etc.).
  - **AuthenticationProfile** - Authentication profile to authenticate before service processing.
  - **ResourcePath** - Service resource path.
  - **RequestJSON** - Request JSON.

Note: Task implementations should only use rules present in the PegaDeploymentManagerIntegrations and PegaDevopsShared rulesets.

Debugging tasks

Use the following steps to run and trace the implementation of your custom tasks in context:

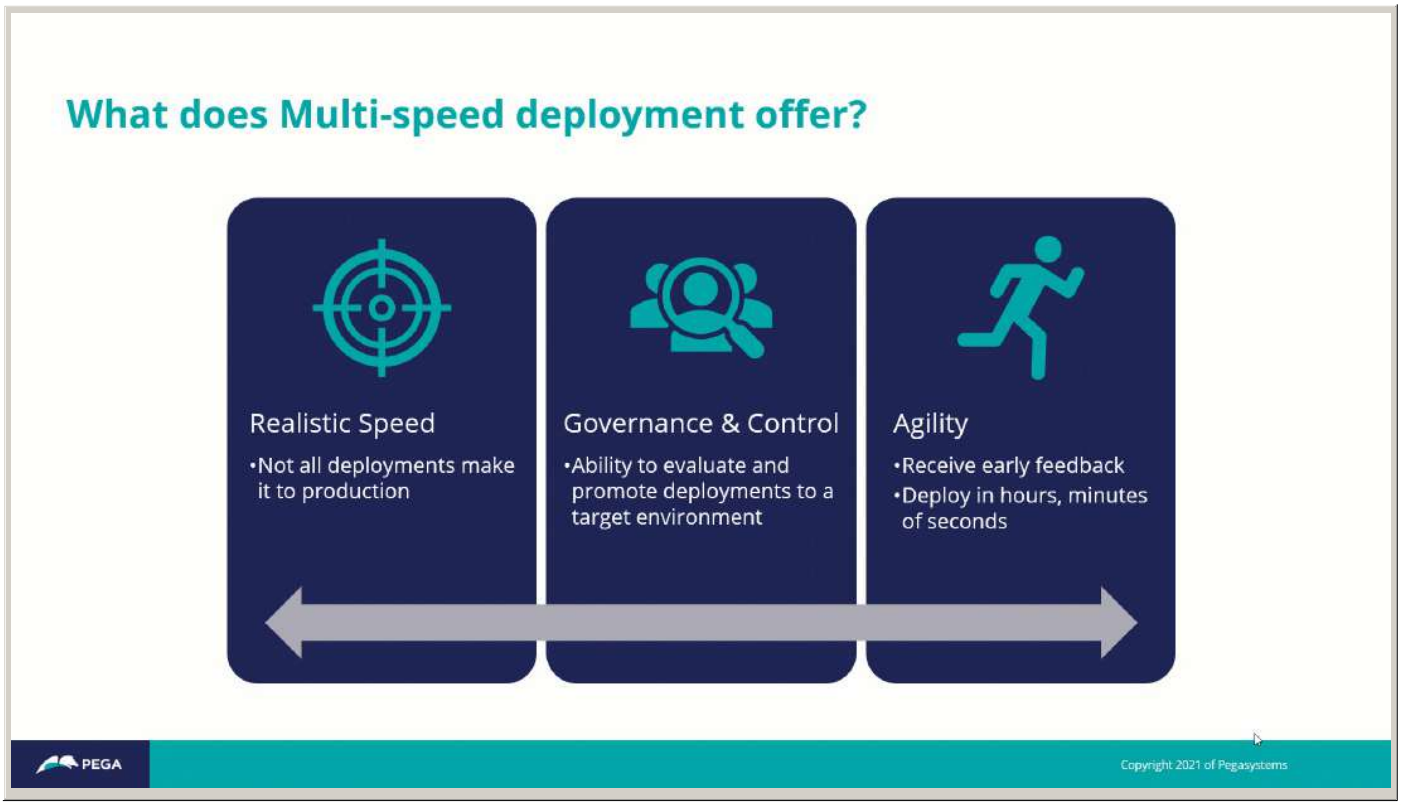
1. Temporarily disable DeploymentManagerAgentScheduler job scheduler from Admin Studio.
2. Add the custom task to a pipeline, preferably as one of the first tasks.
3. Start a deployment and progress to the custom task.
4. Open and track the RunDeploymentManagerAgent activity.
5. Run the RunDeploymentManagerAgent activity.

If the task runs successfully when run manually, but encounters an issue when executed by the job scheduler:

1. Trace the job scheduler named DeploymentManagerAgentScheduler from Admin Studio.
2. To trace task updates posted to Deployment Manager, enable tracer for the service DeploymentManager/v1/tasks.

Using multispeed deployments in 5.1.x

Achieve greater flexibility over your pipeline workflow through multiple deployments that queue in the same stage of a pipeline. With this functionality, developers can regularly merge their changes for selective testing and promotion of deployments without blocking the pipeline. This approach introduces the concept of a transition between stages requiring user input to proceed.



What does Multi-speed deployment offer?
Realistic Speed
•Not all deployments make it to production
Governance & Control
•Ability to evaluate and promote deployments to a target environment
Agility
•Receive early feedback
•Deploy in hours, minutes of seconds
PEGA
Copyright 2021 of Pegasystems

In the example above, there is a manual transition configured when going from Quality Assurance (QA) to Staging. New deployments queue chronologically, waiting for promotion to the next stage. The user with the appropriate privilege is able to select and promote a specific deployment. In this example, Deployment #3 is selected for promotion. This automatically resolves the previous deployments, Deployment #1 and Deployment #2 (superseded). Deployment #3 is promoted to Staging and the pipeline resumes running through all of the tasks configured in the QA stage.

Note: Superseded deployments are resolved and are not be available to promote to the next phase of the pipeline. This process is intended to work on the assumption that newer deployments in a pipeline are cumulative of all the changes in previous deployments.

Configuring a stage for multispeed deployment

In multispeed deployments, the pipeline queues multiple deployments after the completion of a stage, which you can use to selectively promote deployments to the next stage. This functionality relies on the underlying pipeline model, with available configuration from Deployment Manager. When all tasks within a stage are complete, the stage progresses forward by using a configuration called **Transition**. You can configure the transition setting to automatically move to the next stage, or wait for a user action.

1. From the Deployment Manager portal, click **Pipelines Application pipelines** .
2. Select the pipeline that you want to configure for multispeed deployment.
3. On the **Actions** menu, select **Edit pipeline**.
4. Click **Pipeline model**.
5. Click the heading for the stage that requires manual transition, where you want deployments to queue after stage completion until selected for promotion.
6. In the stage details section, in the **When all tasks in this stage are complete** field, select the **Wait for user action** switch.
7. Click **Save**.

Promoting pending deployments

The pipeline modeling landing page provides a dashboard of all pending deployments within your application pipeline. From here, deployments can be selectively promoted for progression through the workflow. **Note:** Deployments queue chronologically within the stage of an application pipeline. If the most recent deployment is selected for promotion, all previously queued deployments will be superseded. Any deployments that you add later are still waiting for promotion.

1. Click **Pipelines Application pipelines** .
2. Open the pipeline that contains the deployment that you want to promote.

   The overview section shows a dashed line between stages that indicate that a manual transition is required. The **Deployment History** section shows this deployment with a Waiting status.

3. In the overview section, choose the action that you want to run on the deployment.

Multiple pipelines for an application

If you are maintaining multiple pipelines across an application for a release process and leverage the **deploy with an existing artifact** feature, Deployment Manager 5.1 offers multispeed deployment. Simply reconfigure your pipeline with a manual transition between stages to utilize this feature. Now, you do not need to carry forward deployments with an existing artifact, as the pipeline will do this for you.

Customers migrating from 4.x versions can now leverage multispeed deployments if maintaining multiple pipelines targeting different stages for an application.

Archiving and activating pipelines

If your role has the appropriate permissions, you can archive inactive pipelines so that they are not displayed on the Deployment Manager landing page.

To archive or activate a pipeline, do the following steps:

1. In the navigation pane of Deployment Manager click **Pipelines Application Pipelines** .
2. To archive a pipeline, select the pipeline name and perform the following steps:
   a. From the **Actions** menu on the pipeline you want to archive, click **Archive**.
   b. In the **Archive pipeline** dialog box, click **Archive**.
3. To activate an archived pipeline, perform the following steps:
   a. Click **Pipelines Application Pipelines** .
   b. Click **Archived** to open the toggle menu.
   c. Click **Activate** to activate the archived pipeline.

- Setting up users and roles

Disabling and enabling a pipeline

If your role has the appropriate permissions, you can disable a pipeline on which errors continuously cause a deployment to fail. Disabling a pipeline prevents branch merging, but you can still view, edit, and stop deployments on a disabled pipeline.

To disable and enable a pipeline, perform the following steps:

1. In the navigation pane of Deployment Manager click **Pipelines Application pipelines** .
2. To disable a pipeline, perform the following steps:
   a. Select the pipeline you want to disable.
   b. From the **Actions** menu, click **Disable** for the pipeline that you want to disable.
3. To enable a disabled pipeline, perform one of the following steps:

- From the **Actions** menu on the disabled pipeline, click **Enable**.
- From the pipeline landing page, navigate to the disable pipeline and click **Enable**.

Managing artifacts generated by Deployment Manager

You can view, download, and delete application packages in repositories that are on the orchestration server. If you are using Deployment Manager on Pega Cloud Services, application packages that you have deployed to cloud repositories are stored on Pega Cloud Services. To manage your cloud storage space, you can download and permanently delete the packages.

If you are using a separate product rule to manage a test application, the name of the product rule is the same as that of the product rule with _Tests appended to it.

Do the following steps:

1. If the pipeline is not open, in the navigation pane of Deployment Manager, click **Pipelines** and then click the name of the pipeline.
2. Click the pipeline for which you want to download or delete packages.
3. Click **Actions Browse artifacts** .
4. Click either **Development Repository** or **Production Repository**.
5. To download a package, click the package, and then save it to the appropriate location.

Deleting a pipeline

If your role has the appropriate permission, you can delete a pipeline. When you delete a pipeline, its associated application packages are not removed from the repositories that the pipeline is configured to use.

To delete a pipeline, do the following steps:

1. If the pipeline is not open, in the navigation pane of Deployment Manager, click **Pipelines** and then click the name of the pipeline.
2. Click the **Actions** menu, and then click **Delete** for the pipeline that you want to delete.
3. In the **Delete pipeline** confirmation dialogue box, click **Delete** to confirm.

Interacting with application pipelines

- Running deployments in Deployment Manager

  You can start deployments in a number of ways. For example, you can start a deployment manually if you are not using branches, by submitting a branch into the Merge Branches wizard, or by publishing application changes in App Studio to create a patch version of your application. Your user role determines if you can start a deployment.

- Pausing and resuming deployments

  When you pause a deployment, the pipeline completes the task that it is running, and stops the deployment at the next step. Your user role determines if you can pause a deployment.

- Stopping a deployment

  If your role has the appropriate permissions, you can a deployment to prevent it from moving through the pipeline.

- Rolling back a deployment

  A rollback can be performed on any deployment that encounters an error or fails for any reason. The rollback feature relies on restore points, which are automatically generated every time an import happens. Any change implemented after the import but before the next restore point will be lost when the rollback action is triggered.

- Accepting or rejecting a manual step

  If a manual step is configured on a stage, the deployment pauses when it reaches the step, and you can either complete it or reject it if your role has the appropriate permissions. For example, if a user was assigned a task and completed it, you can complete the task in the pipeline to continue the deployment. Deployment Manager also sends you an email when there is a manual step in the pipeline. You can complete or reject a step either within the pipeline or through email.

- Managing aged updates

  An aged update is a rule or data instance in an application package that is older than an instance that is on a system to which you want to deploy the application package. By being able to import aged updates, skip the import, or manually deploy your application changes, you now have more flexibility in determining the rules that you want in your application and how you want to deploy them.

- Managing a deployment that has errors

If a deployment has errors, the pipeline stops processing on it. You can perform actions such as rolling back the deployment or skipping the step on which the error occurred.

- Understanding schema changes in application packages in 5.1.x

  If an application package that is to be deployed on candidate systems contains schema changes, the Pega Platform orchestration server checks the candidate system to verify that you have the required privileges to deploy the schema changes. One of the following results occurs:

- Deploying revisions with Deployment Manager

  As a best practice, you should use Deployment Manager 5.1 or newer to automate your business change management process, run CI/CD pipelines to merge a revision package to the development system of record (SOR), and subsequently deploy that package through to your production environment by using a deployment pipeline.

Running deployments in Deployment Manager

You can start deployments in a number of ways. For example, you can start a deployment manually if you are not using branches, by submitting a branch into the Merge Branches wizard, or by publishing application changes in App Studio to create a patch version of your application. Your user role determines if you can start a deployment.

- Manually starting a deployment in Deployment Manager

  You can start a deployment manually if you are not using branches and are working directly in rulesets. You can also start a deployment manually if you do not want deployments to start automatically when branches are merged.

- Starting a deployment as you merge branches from the development environment in 5.1.x

  In either a branch-based or distributed, branch-based environment, you can immediately start a deployment by submitting a branch into a pipeline in the Merge Branches wizard. The wizard displays the merge status of branches so that you do not need to open Deployment Manager to view it.

Manually starting a deployment in Deployment Manager

You can start a deployment manually if you are not using branches and are working directly in rulesets. You can also start a deployment manually if you do not want deployments to start automatically when branches are merged.

To start a deployment manually, do the following steps:

1. In the navigation pane, click **Pipelines Application pipelines** , and then click **Start deployment** for the pipeline that you want to start.
2. In the **Start deployment** dialog box, provide a description of the deployment. You can use this field to tag a deployment with a version note, such as alpha, beta, pilot, or RC. Click **Submit** to start the deployment.

Starting a deployment as you merge branches from the development environment in 5.1.x

In either a branch-based or distributed, branch-based environment, you can immediately start a deployment by submitting a branch into a pipeline in the Merge Branches wizard. The wizard displays the merge status of branches so that you do not need to open Deployment Manager to view it.

If you are using a separate product rule for a test application, after you start a deployment either by using the Merge Branches wizard, the branches of both the target and test applications are merged in the pipeline.

You can submit a branch to your application and start the continuous integration portion of the pipeline when the following criteria is met:

- You have created a pipeline for your application in Deployment Manager.
- You are merging a single branch.
- The OrchestratorURL dynamic system setting, which defines the URL of orchestration server, is configured on the system.
- All the rulesets in your branch belong to a single application that is associated with your pipeline. Therefore, your branch cannot contain rulesets that belong to different application layers.

- Configuring settings before using the Merge Branches wizard

  You can start a branch merge, which triggers a deployment, by using the Merge Branches wizard. You must configure certain settings before you can submit a branch to your application.

- Submitting a branch into an application by using the Merge Branches wizard

  You can start a branch merge, which triggers a deployment, by submitting a branch into an application in the Merge Branches wizard. By using the wizard to start merges, you can start a deployment without additional configuration.

Configuring settings before using the Merge Branches wizard

You can start a branch merge, which triggers a deployment, by using the Merge Branches wizard. You must configure certain settings before you can submit a branch to your application.

Before you begin: Before starting a branch merge, do the following tasks.

1. Check all rules into their base rulesets before you merge them.
2. Check if there are any potential conflicts to address before merging branches. For more information, see Viewing branch quality and branch contents.
3. As a best practice, lock a branch after development is complete so that no more changes can be made. For more information, see Locking a branch.

Submitting a branch into an application by using the Merge Branches wizard

You can start a branch merge, which triggers a deployment, by submitting a branch into an application in the Merge Branches wizard. By using the wizard to start merges, you can start a deployment without additional configuration.

To submit a branch into an application by using the Merge Branches wizard, perform the following steps:

1. In the navigation pane of Dev Studio,, click **App**, and then click **Branches**.
2. Right-click the branch and click **Merge**.
3. Click **Proceed**. **Result:** The wizard displays a message in the following scenarios:
   - If there are no pipelines that are configured for your application or there are no branches in the target application.
   - If the value for the OrchestratorURL dynamic system setting is not valid.
4. Click **Switch to standard merge** to switch to the Merge Branches wizard that you can use to merge branches into target rulesets. For more information, see Merging branches into target rulesets .
5. In the **Application pipelines** section, from the **Pipeline** list, select the application for which the pipeline is configured into which you want to merge branches.
6. In the **Merge Description** field, enter information that you want to capture about the merge.

   This information appears when you view deployment details.

7. In the **Associated User stories/bugs** field, press the Down arrow key, and then select the Agile Workbench user story or bug that you want to associate with this branch merge.
8. Click **Merge**.

Result:

The system queues the branch for merging, generates a case ID for the merge, and runs the continuous integration criteria that you specified.

If there are errors, and the merge is not successful, an email is sent to the operator ID of the release manager that is specified on the orchestration server.

The branch is stored in the development repository and, after the merge is completed, Deployment Manager deletes the branch from the development system. By storing branches in the development repository, Deployment Manager keeps a history, which you can view, of the branches in a centralized location.

If your development system is appropriately configured, you can rebase your development application to obtain the most recently committed rulesets after you merge your branches. For more information, see Rebasing rules to obtain latest versions.

Pausing and resuming deployments

When you pause a deployment, the pipeline completes the task that it is running, and stops the deployment at the next step. Your user role determines if you can pause a deployment.

Note: Deployments can be paused or resumed directly from the pipeline landing page. If you wish to navigate directly to a pipeline, follow the steps below. To pause a deployment:

1. If the pipeline is not open, in the navigation pane of Deployment Manager, click **Pipelines** and then click the name of the pipeline.
2. Select the stage where the deployment is active and click the **More** icon.
3. Click **Pause**.
4. To resume a deployment, navigate to the stage where the deployment is paused and click the **More** icon.
5. Click **Resume** to continue the deployment.

Stopping a deployment

If your role has the appropriate permissions, you can a deployment to prevent it from moving through the pipeline.

1. If the pipeline is not open, in the navigation pane of Deployment Manager, click **Pipelines** and then click the name of the pipeline.
2. Select the stage where the deployment is active.
3. Click the **More** icon, and then click **Abort**.

Rolling back a deployment

A rollback can be performed on any deployment that encounters an error or fails for any reason. The rollback feature relies on restore points, which are automatically generated every time an import happens. Any change implemented after the import but before the next restore point will be lost when the rollback action is triggered.

Note: For Pega versions 8.3 and earlier the rollback execution impacts the entire system and is not scoped to the application. Be careful about using restore when there are multiple independent applications on the system.

For Pega versions 8.4 and later with PegaDevOpsFoundation 4.8 and later, this will be an application-level rollback with no impact on other applications. Refer to the Application-level rollback document for more information.

To perform a rollback, see Managing a deployment that has errors.

Rolling back a successful deployment

If there is a situation where a successful deployment must be rolled back, perform the following steps:

1. Insert a manual step as the final task in the stage where rollback is required.
2. Assign the manual step to a user who will make the decision to approve/reject the deployment.
3. To rollback, reject the deployment via email or directly in Deployment Manager.

Accepting or rejecting a manual step

If a manual step is configured on a stage, the deployment pauses when it reaches the step, and you can either complete it or reject it if your role has the appropriate permissions. For example, if a user was assigned a task and completed it, you can complete the task in the pipeline to continue the deployment. Deployment Manager also sends you an email when there is a manual step in the pipeline. You can complete or reject a step either within the pipeline or through email.

Deployment Manager also generates a manual step if there are schema changes in the application package that the release manager must apply. For more information, see Understanding schema changes in application packages in 5.1.x. To complete or reject a manual step within the deployment, do the following steps:

1. To complete or reject a manual step from within an email, click either **Accept** or **Reject**.
2. To complete or reject a manual step in the pipeline,
    a. If the pipeline is not open, in the navigation pane, click **Pipelines Application pipelines** , and then click the name of the pipeline.
    b. Accept or reject the step by doing one of the following actions:
    - To resolve the task so that the deployment continues through the pipeline, click **Complete**.
    - To reject the task so that the deployment does not proceed, click **Reject**.

Managing aged updates

An aged update is a rule or data instance in an application package that is older than an instance that is on a system to which you want to deploy the application package. By being able to import aged updates, skip the import, or manually deploy your application changes, you now have more flexibility in determining the rules that you want in your application and how you want to deploy them.

If your role has the appropriate permissions, you can manage aged updates in a number of ways, such as importing them, skipping the import, or manually deploying applications. Managing aged updates gives you more flexibility in how you deploy application changes. Do the following steps:

1. If the pipeline is not open, in the navigation pane of Deployment Manager, click **Pipelines** and then click the name of the pipeline.
2. Click **View aged updates** to view a list of the rules and data instances, which are in the application package, that are older than the instances that are on the system.
3. Click the **More** icon and do one of the following actions:
    - To import the older rule and data instances that are in the application package into the system, which overwrites the newer versions that are on the system, click **Override aged updates**.
    - To skip the import, click **Skip aged updates**.
    - To manually deploy the package from the Import wizard on the system, click **Deploy manually and resume**. Deployment Manager does not run the Deploy step on the stage.

Managing a deployment that has errors

If a deployment has errors, the pipeline stops processing on it. You can perform actions such as rolling back the deployment or skipping the step on which the error occurred.

Do the following steps:

1. If the pipeline is not open, in the navigation pane of Deployment Manager, click **Pipelines** and then click the name of the pipeline.
2. Click the **More** icon, and then do one of the following actions:
    - To resume running the pipeline from the task, click **Retry**.
    - To roll back to an earlier deployment, click **Rollback**.
        - Pega Platform 8.4 supports application-level rollback. To leverage this functionality, candidate and orchestrator environments must be on Deployment Manager 4.8 or above. For older versions of Deployment Manager (4.7.x and below) or users on Pega Platform 8.3 or below, rollback defaults to the system-level.

          For more information on rollback, see Rolling back a deployment.
    - To stop running the deployment, click **Abort**.
    - To skip the failed task and resume deployment from next available task, click **Skip**. You must have the Skip on Failure privilege associated with your role to use this feature.

Understanding schema changes in application packages in 5.1.x

If an application package that is to be deployed on candidate systems contains schema changes, the Pega Platform orchestration server checks the candidate system to verify that you have the required privileges to deploy the schema changes. One of the following results occurs:

- If you have the appropriate privileges, schema changes are automatically applied to the candidate system, the application package is deployed to the candidate system, and the pipeline continues.
- If you do not have the appropriate privileges, Deployment Manager generates an SQL file that lists the schema changes and sends it to your email address. It also creates a manual step, pausing the pipeline, so that you can apply the schema changes. After you complete the step, the pipeline continues. For more information about completing a step, see Completing or rejecting a manual step.

You can also configure settings to automatically deploy schema changes so that you do not have to manually apply them if you do not have the required privileges. For more information, see Configuring settings to automatically deploy schema changes.

Your user role must have the appropriate permissions so that you can manage schema changes.

Configuring settings to automatically apply schema changes

You can configure settings to automatically deploy schema changes that are in an application package that is to be deployed on candidate systems. Configure these settings so that you do not have to apply schema changes if you do not have the privileges to deploy them.

Do the following steps:

1. On the candidate system, in Pega Platform, set the *AutoDBSchemaChanges* dynamic system setting to true to enable schema changes at the system level.
    a. In Dev Studio, search for AutoDBSchemaChanges.
    b. In the dialog box that appears for the search results, click **AutoDBSchemaChanges**.
    c. On the **Settings** tab, in the **Value** field, enter true.
    d. Click **Save**.
2. Add the *SchemaImport* privilege to your access role to enable schema changes at the user level. For more information, see Specifying privileges for an Access of Role to Object rule.

Result: These settings are applied sequentially. If the *AutoDBSchemaChanges* dynamic system setting is set to false, you cannot deploy schema changes, even if you have the *SchemaImport* privilege.

For more information about the *database/AutoDBSchemaChanges* dynamic system setting, see Importing rules and data by using a direct connection to the database.

Schema changes are also attached to the deployment report for the pipeline.

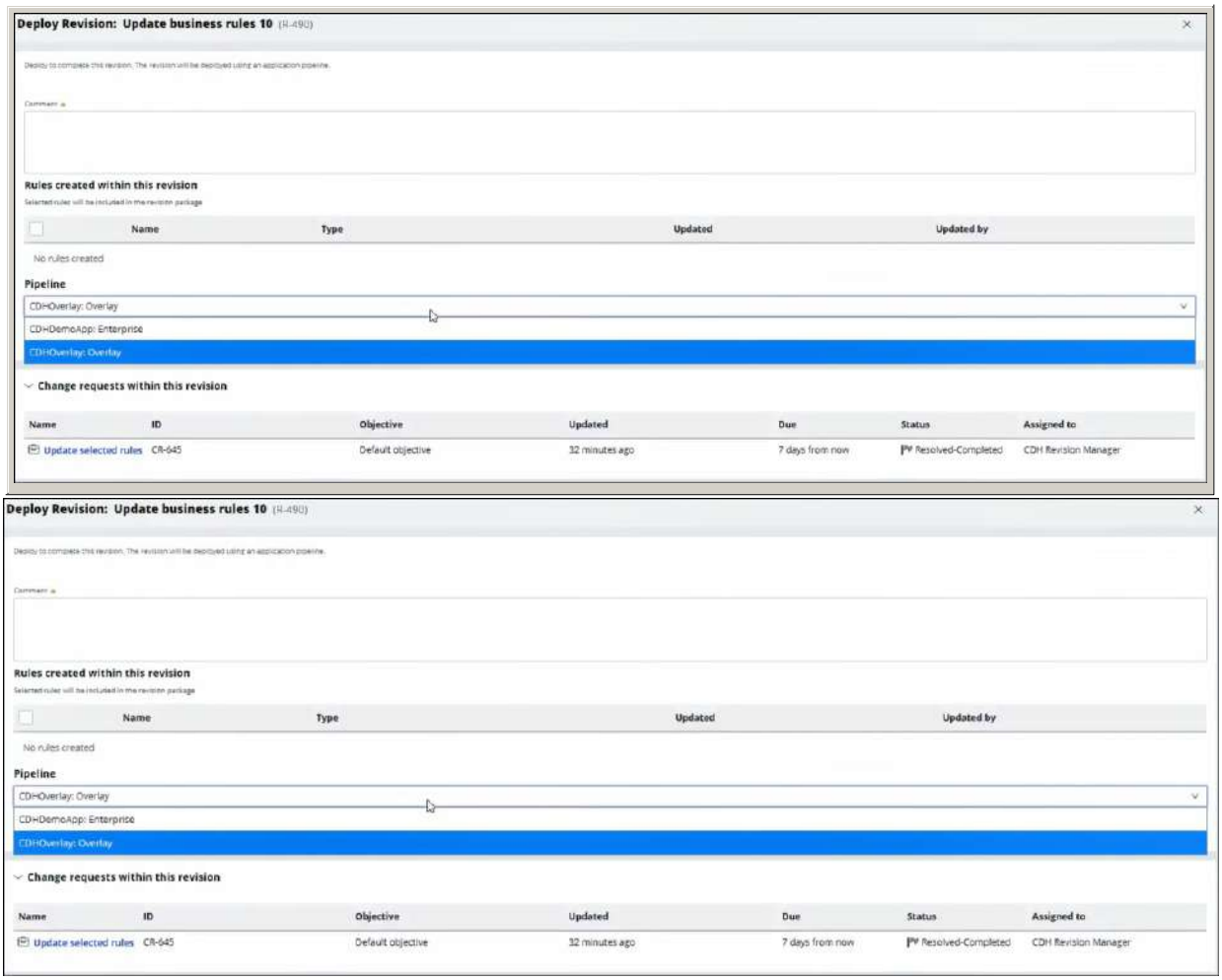
Deploying revisions with Deployment Manager

As a best practice, you should use Deployment Manager 5.1 or newer to automate your business change management process, run CI/CD pipelines to merge a revision package to the development system of record (SOR), and subsequently deploy that package through to your production environment by using a deployment pipeline.

Before you begin: Configure Deployment Manager. For more information, see Using Deployment Manager for model-driven DevOps.**Important:** Your Deployment Manager version must be 5.1 or newer. Previous versions of Deployment Manager do not support working with revision management packages. After a revision manager submits a change request

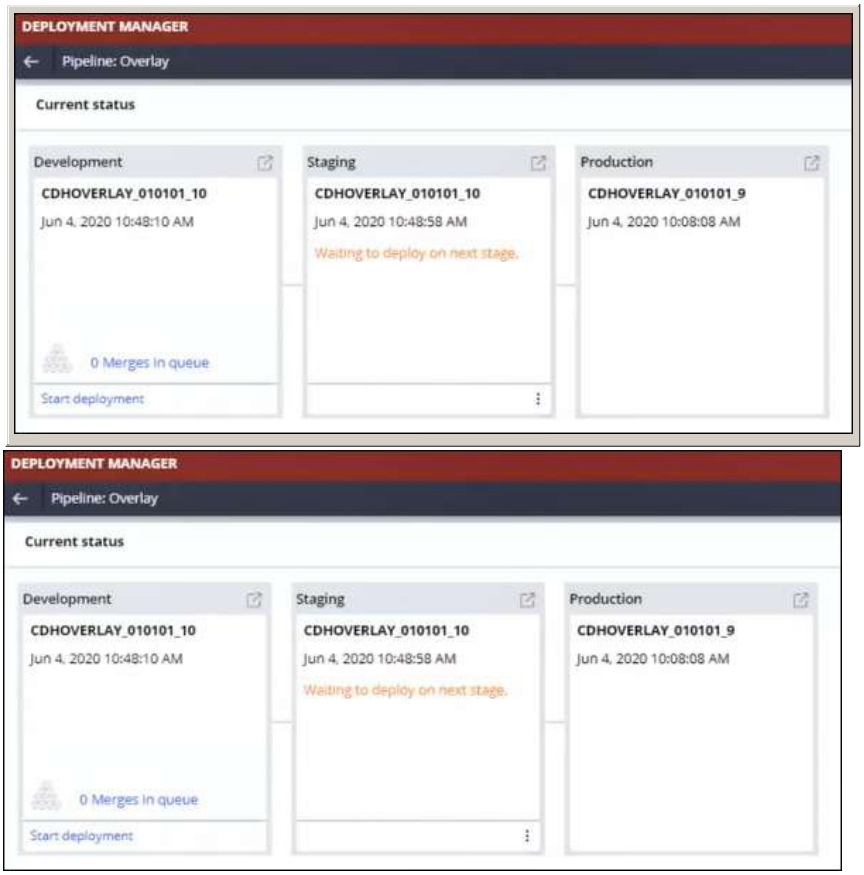
in the business operations environment and a strategy designer implements the required changes, the new revision must be deployed to production. To automatize the deployment process, use Deployment Manager pipelines. You can merge a revision to a deployment pipeline directly from the Pega Customer Decision Hub portal.

1. Log in to the Pega Customer Decision Hub portal as an operator with the *RevisionManager* access role.
2. In the left navigation panel, click **Revision Management**.
3. On the Revision Management landing page, click **Revisions**.
4. Click the revision that you want to deploy and review its contents.
5. Click **Deploy revision**.
6. In the **Pipelines** section, select a Deployment Manager pipeline to merge the package to the development system of record, as shown in the following figure:



7. As a best practice, in the **Comment** field, enter a short description of the changes included in the revision, for example, Introduced a new business issue.
8. Click **Deploy**. **Result:** Deployment Manager queues the revision to merge it into a pipeline. After a successful merge, the revision deploys automatically to all environments in the pipeline. You can monitor the status of the deployment process in the Pega Customer Decision Hub portal, as shown in the following figure:

9. **Optional:** Depending on the configuration of your deployment pipeline, you may need to manually approve the revision for deployment to your production environment. If required, click the **Deployment Manager portal** link and approve the revision in Deployment Manager, as in the following figure:



10. Click **Complete revision** to resolve the revision.

Configuring data migration pipelines

Data migration tests provide you with significant insight into how the changes that you make to decision logic affect the results of your strategies. To ensure that your simulations are reliable enough to help you make important business decisions, you can deploy a sample of your production data to a dedicated data migration test environment.

When you use Deployment Manager 5.1.x in data migration pipelines, you automate exporting data from the production environment and into the simulation environment. Data migration pipelines also require the following:

- Pega Platform 8.3.x or higher
- Decision management
- Pega Marketing

- Modifying a pipeline

  You can change the URLs of your production and simulation environments. You can also change the application information for which you are creating the pipeline.

- Diagnosing a pipeline

  You can diagnose your pipeline to verify its configuration. For example, you can verify that the orchestration system can connect to the production and simulation environments.

- Scheduling a pipeline by creating a job scheduler rule

  You can schedule a data migration pipeline to run during a specified period of time by creating and running a job scheduler. The job scheduler runs a Deployment Manager activity (pzScheduleDataSyncPipeline) on the specified pipeline, based on your configuration, such as weekly or monthly.

Modifying a pipeline

You can change the URLs of your production and simulation environments. You can also change the application information for which you are creating the pipeline.

1. If the pipeline is not open, in the navigation pane of Deployment Manager, click **Pipelines Data migration pipelines** , and then click the name of the pipeline.
2. Click **Action Settings** .
3. Modify environment details by doing the following:
   a. Click **Environment Details**.
   b. In the **Environment** fields, enter the URLs of the production and simulation environments.
4. To change the application information for which you are creating the pipeline, click **Application details**.
   a. From the **Version** list, select the application version.
   b. From the **Access group** list, select the access group for which you want to run pipeline tasks.
   This access group must be present on the production and simulation environments and have at least the *sysadmin4* role.
5. Click **Save**.

Diagnosing a pipeline

You can diagnose your pipeline to verify its configuration. For example, you can verify that the orchestration system can connect to the production and simulation environments.

1. If the pipeline is not open, in the navigation pane of Deployment Manager, click **Pipelines Data migration pipelines** , and then click the name of the pipeline.
2. Click **Actions Diagnose pipeline** .
3. In the **Diagnostics** window, review the errors, if any.

Scheduling a pipeline by creating a job scheduler rule

You can schedule a data migration pipeline to run during a specified period of time by creating and running a job scheduler. The job scheduler runs a Deployment Manager activity (*pzScheduleDataSyncPipeline*) on the specified pipeline, based on your configuration, such as weekly or monthly.

For more information about job scheduler rules, see Job Scheduler rules.

Do the following steps:

1. On the orchestration server, in the navigation panel of Dev Studio, click **Records SysAdmin Job Scheduler** .
2. On the **Create Job Scheduler** rule form, enter the label of the scheduler and select the ruleset into which to save the job scheduler.
3. Click **Create and open**.
4. On the **Edit Job Scheduler** rule form, on the **Definition** tab, from the **Runs on** list, configure the job scheduler to run on all or one nodes:
   - To run the job scheduler on all nodes in a cluster, click **All associated nodes**.
   - To run the job scheduler on only one node in a cluster, click **Any one associated node**.
5. From the **Schedule** list, select how often you want to start the job scheduler, and then specify the options for it.
6. Select the context for the activity resolution.
   - If you want to resolve the *pzScheduleDataSyncPipeline* activity in the context of Deployment Manager, go to step 7.
   - If you want to resolve the activity in the context that is specified in the System Runtime Context, go to step 8.
7. To resolve the *pzScheduleDataSyncPipeline* activity in the context of Deployment Manager:
   a. From the **Context** list, select **Specify access group**.
   b. In the **Access group** field, press the Down arrow key and select the access group that can access Deployment Manager.
   c. Go to step 9.
8. to resolve the activity in the context that is specified in the System Runtime Context:
   a. From the **Context** list, select **Use System Runtime Context**.
   b. Update the access group of the batch requestor type access group with the access group that can access Deployment Manager; in the header of Dev Studio, clicking **Configure System General** .
   c. On the **System:General** page, on the **Requestors** tab, click the **BATCH** requestor type.
   d. On the **Edit Requestor Type** rule form, on the **Definition** tab, in the **Access Group Name** field, press the Down arrow key and select the access group that can access Deployment Manager.
   e. Click **Save**.
9. On the **Job Schedule** rule form, in the **Class** field, press the Down arrow key and select **Pega-Pipeline-DataSync**.
10. In the **Activity** field, press the Down arrow key and select **pzScheduleDataSyncPipeline**.

11. Click the **Parameters** link that appears below the **Activity** field.
12. In the **Activity Parameters** dialog box, in the **Parameter value** field for the **PipelineName** parameter, enter the data migration pipeline that the job scheduler runs.
13. In the **Parameter** value field for the **ApplicationName** parameter, enter the application that the data migration pipeline is running.
14. Click **Submit**.
15. Save the Job schedule rule form.

Interacting with data migration pipelines

Create and run data migration pipelines in Deployment Manager 5.1.x to automatically export simulation data from a production environment into a simulation environment in which you can test simulation data. You can also use Deployment Manager to monitor and obtain information about your simulations, for example, by running diagnostics to ensure that your environment configurations are correct and by and viewing reports that display key performance indicators (KPIs).

- Starting a pipeline manually

  If you do not run a data migration pipeline based on a job scheduler, you can run it manually in Deployment Manager.

- Pausing a pipeline

  Pause a pipeline to stop processing the data migration. When you pause a data migration, the pipeline completes the current task and stops the data migration.

- Stopping a pipeline

  Stop a pipeline to stop data migrations from being exported and imported.

- Stopping and resuming a pipeline that has errors

  If a data migration has errors, the pipeline stops processing on it, and you can either resume or stop running the pipeline.

- Deleting a pipeline

  When you delete a pipeline, its associated application packages are not deleted from the pipeline repositories.

- Viewing data migration logs

  View the logs for a data migration to see the completion status of operations, for example, when a data migration moves to a new stage. You can change the logging level to control the events are displayed in the log. For example, you can change logging levels of your deployment from INFO to DEBUG for troubleshooting purposes. For more information, see Logging Level Settings tool.

- Viewing a report for a specific data migration

  You can view a report for a specific data migration to gain more visibility into data migrations on a pipeline.

- Viewing reports for all data migrations in a pipeline

  Reports provide a variety of information about all the data migrations in your pipeline so that you can gain more visibility into data migration processing. For example, you can view the average time taken to complete data migrations.

Starting a pipeline manually

If you do not run a data migration pipeline based on a job scheduler, you can run it manually in Deployment Manager.

1. Do one of the following actions:
   - If the pipeline for which you want to run a data migration is open, click **Start data migration**.
   - If the pipeline is not open, click **Pipelines Data migration pipelines** , and then click **Start data migration**.
2. In the **Start data migration** dialog box, click **Yes**.

Pausing a pipeline

Pause a pipeline to stop processing the data migration. When you pause a data migration, the pipeline completes the current task and stops the data migration.

1. If the pipeline is not open, in the navigation pane of Deployment Manager, click **Pipelines Data migration pipelines** , and then click the name of the pipeline.
2. Click **Pause**.

Stopping a pipeline

Stop a pipeline to stop data migrations from being exported and imported.

1. If the pipeline is not open, in the navigation pane of Deployment Manager, click **Pipelines** **Data migration pipelines** , and then click the name of the pipeline.
2. Click the **More** icon, and then click **Abort**.

Stopping and resuming a pipeline that has errors

If a data migration has errors, the pipeline stops processing on it, and you can either resume or stop running the pipeline.

1. If the pipeline is not open, in the navigation pane of Deployment Manager, click **Pipelines** **Data migration pipelines** , and then click the name of the pipeline.
2. Click the **More** icon, and then do one of the following:
    - To resume running the pipeline from the task, click **Start data migration pipeline**.
    - To stop running the pipeline, click **Abort**.

Deleting a pipeline

When you delete a pipeline, its associated application packages are not deleted from the pipeline repositories.

1. If the pipeline is not open, in the navigation pane of Deployment Manager, click **Pipelines** **Data migration pipelines** , and then click the name of the pipeline.
2. Click the **Delete** icon for the pipeline that you want to delete.
3. Click **Submit**.

Viewing data migration logs

View the logs for a data migration to see the completion status of operations, for example, when a data migration moves to a new stage. You can change the logging level to control the events are displayed in the log. For example, you can change logging levels of your deployment from INFO to DEBUG for troubleshooting purposes. For more information, see Logging Level Settings tool.

1. If the pipeline is not open, in the navigation pane of Deployment Manager, click **Pipelines** **Data migration pipelines** , and then click the name of the pipeline.
2. Do one of the following actions:
    - To view the log for the current data migration, click the **More** icon, and then click **View logs**.
    - To view the log for a previous data migration, expand the **Deployment History** pane and click **Logs** for the appropriate deployment.

Viewing a report for a specific data migration

You can view a report for a specific data migration to gain more visibility into data migrations on a pipeline.

1. If the pipeline is not open, in the navigation pane of Deployment Manager, click **Pipelines** **Data migration pipelines** , and then click the name of the pipeline.
2. Perform one of the following actions:
    - To view the report for the current deployment, click the **More** icon, and then click **View report**.
    - To view the report for a previous deployment, expand the **Deployment History** pane and click **Reports** for the appropriate deployment.

Viewing reports for all data migrations in a pipeline

Reports provide a variety of information about all the data migrations in your pipeline so that you can gain more visibility into data migration processing. For example, you can view the average time taken to complete data migrations.

You can view the following key performance indicators (KPI):

- Data migration success – Percentage of successfully completed data migrations
- Data migration frequency – Frequency of new deployments to production
- Data migration speed – Average time taken to complete data migrations
- Start frequency – Frequency at which new data migrations are triggered
- Failure rate – Average number of failures per data migration

To view reports, do the following tasks:

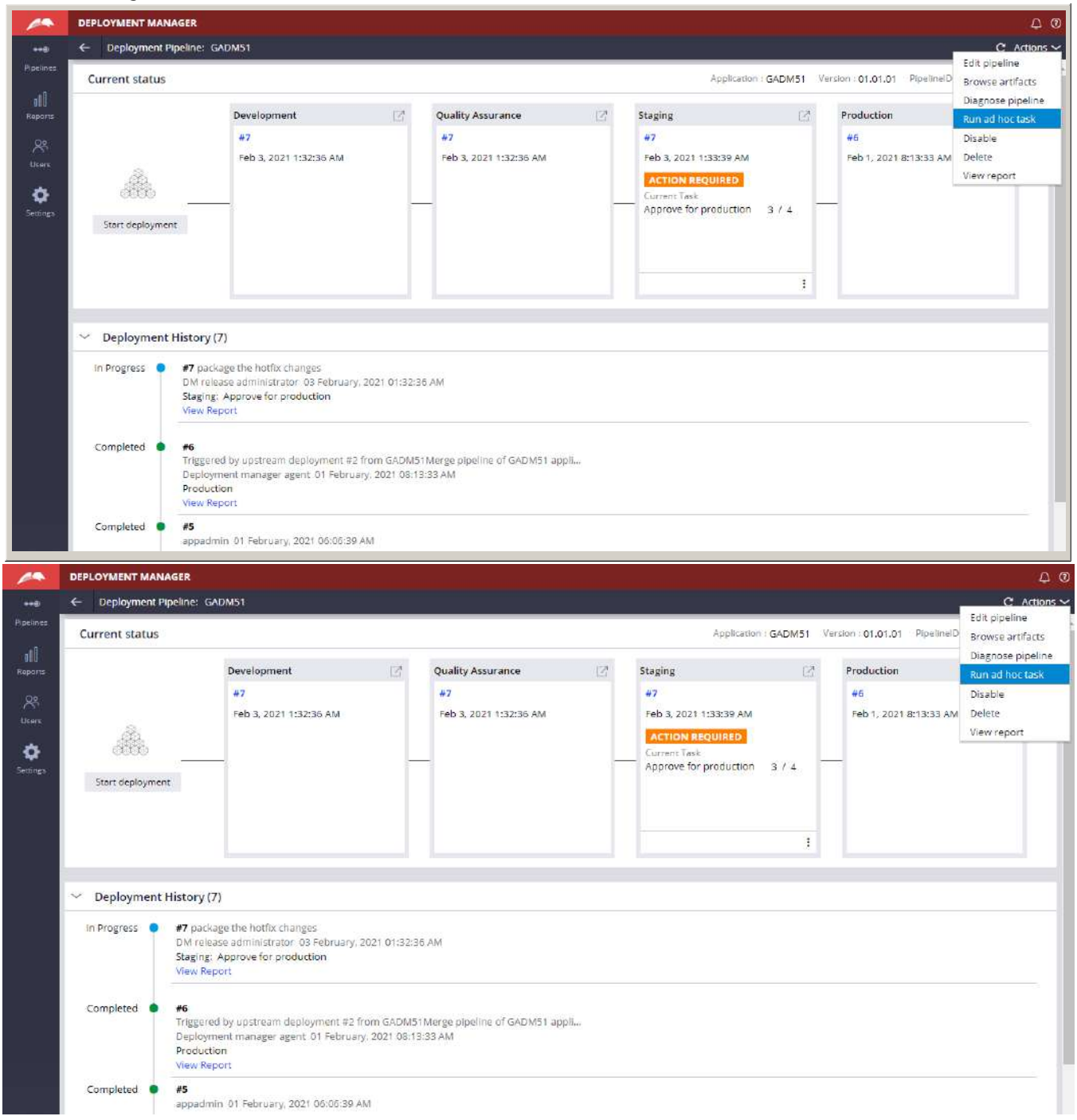
1. Do one of the following actions:
    - If the pipeline is open, click **Actions >View report**.
    - If a pipeline is not open, in the navigation pane, click **Reports**. Next, in the **Pipeline** field, press the Down arrow key and select the name of the pipeline for which to view the report.
2. From the list that appears in the top right of the **Reports** page, select whether you want to view reports for all deployments, the last 20 deployments, or the last 50 deployments.

Performing ad hoc tasks

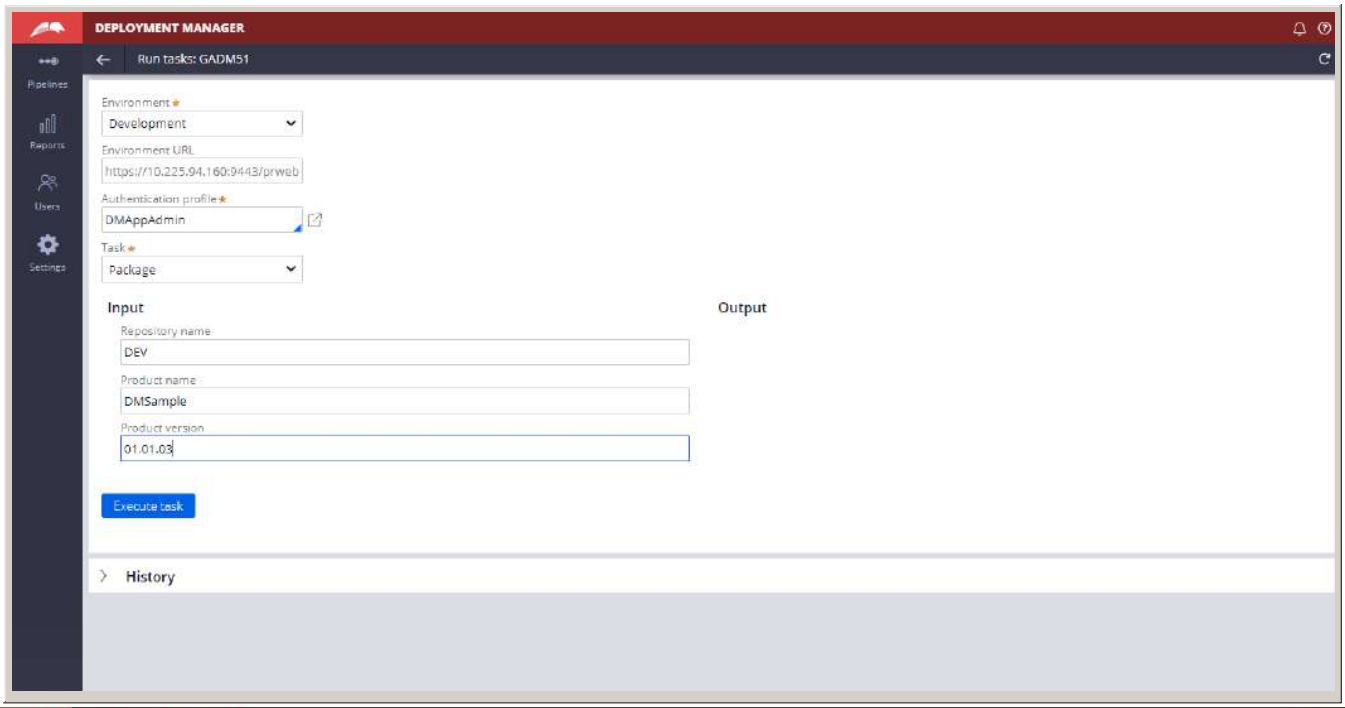
By using ad hoc tasks, you can independently perform package and deploy actions outside of the context of a deployment, which is useful for exporting artifacts from the production environment and importing them into a lower environment, or for packaging and deploying artifacts if a Deployment Manager pipeline is not available.

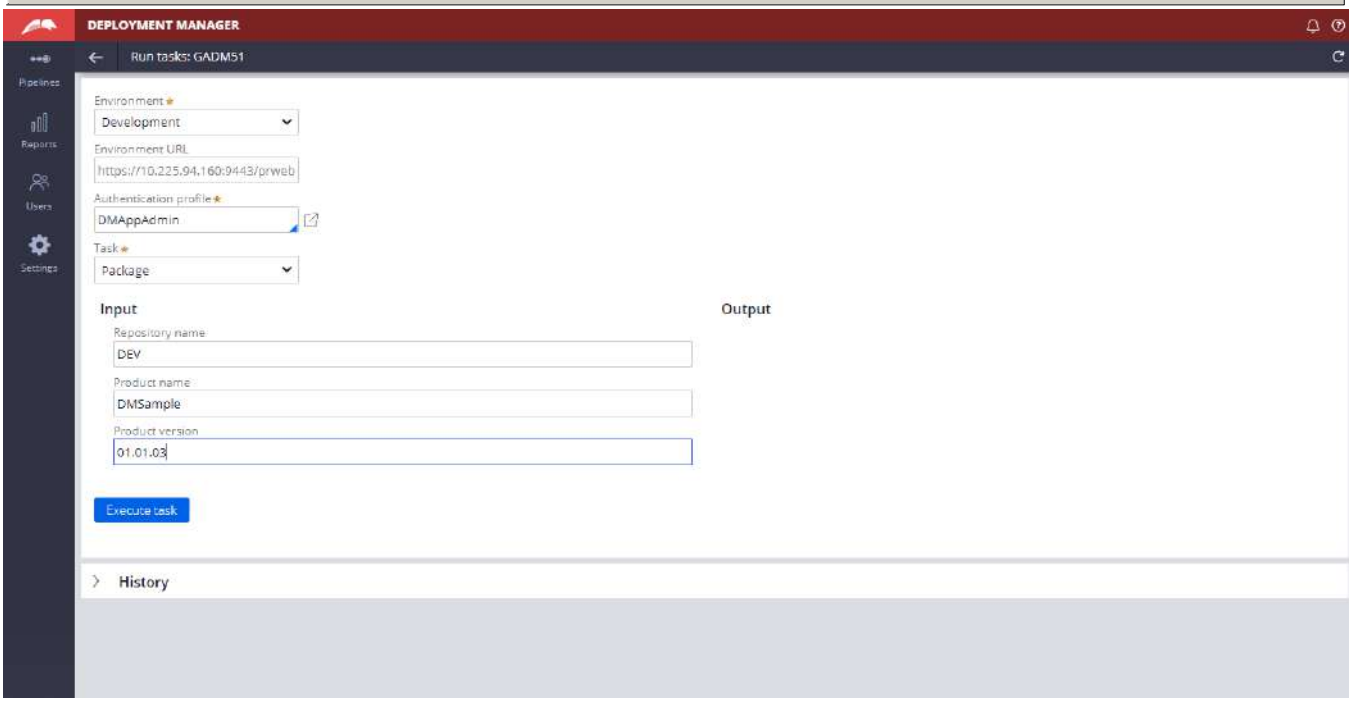
Note: Ad hoc tasks are restricted to Super Admin users within Deployment Manager, or any user that has access to the **Run ad hoc task** privilege in Deployment Manager. If import conflicts are encountered when using ad hoc tasks, the conflict will always be overwritten.

1. If a pipeline is not open, in the navigation pane of Deployment Manager, click **Pipelines Application pipelines** , and then click the name of a pipeline.
2. From the navigation banner, click **Actions Run ad hoc tasks** .

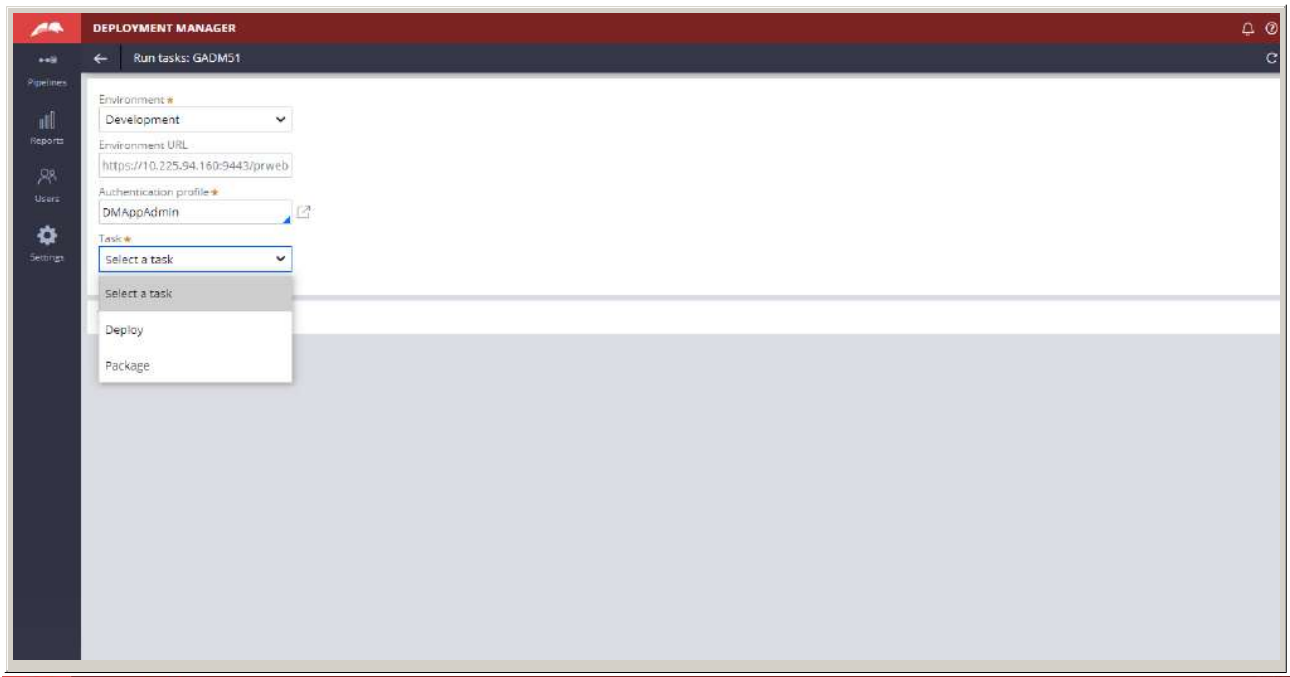


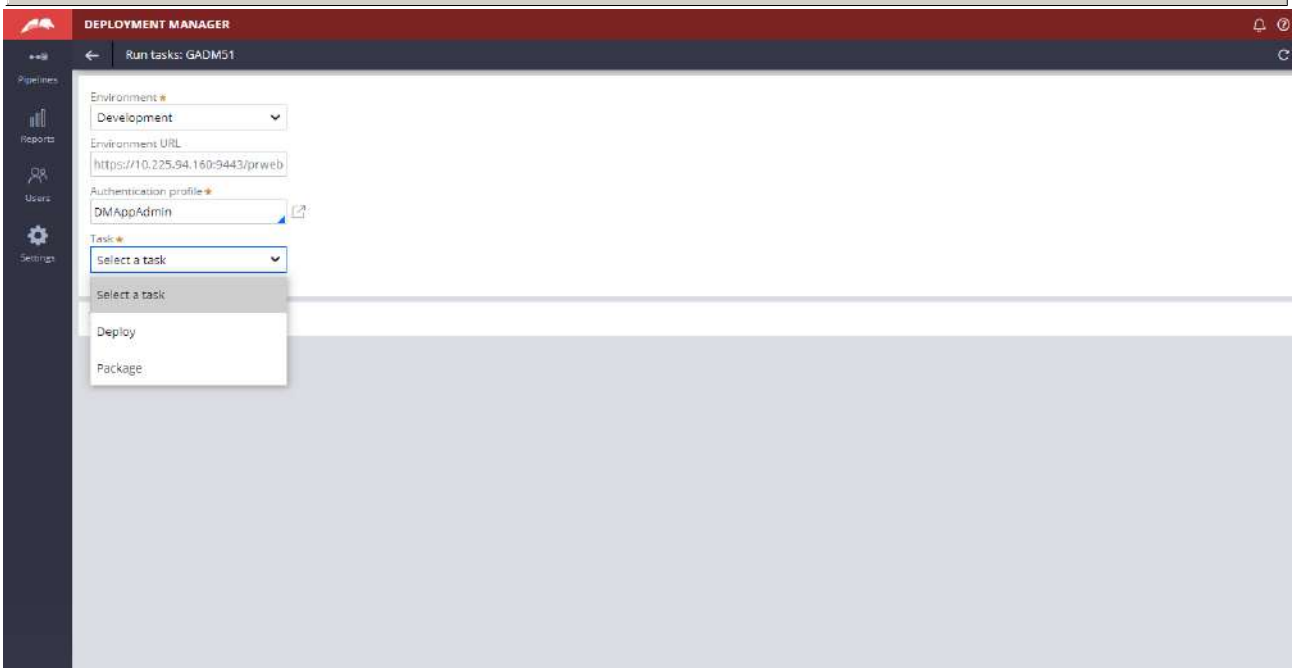
3. In the **Run ad hoc tasks** window, select an environment and an authentication profile.





4. Select a task that you want to run on an artifact:
    - To package an artifact, on the **Task** menu click **Package**, and enter the repository name, product name, and version that you want to package.
    - To deploy an artifact, on the **Task** menu click **Deploy**, and enter the repository name and artifact path for the package you want to deploy.





5. Select **Execute task**

Understanding App Studio publishing

App Studio publish enables Pega Platform users to publish and deploy application changes with no prior Deployment Manager experience necessary. This seamless functionality publishes minor application changes through deployment pipelines to create new artifacts without ever leaving App Studio.

Note: Some configurations do require Deployment Manager interaction. Please contact your system administrator for access to the orchestrator environment.

- Configuring App Studio for publishing

  Publish application updates in App Studio using an existing deployment pipeline to seamlessly create patch versions of an application and start a deployment.

- Publishing application changes in App Studio

  You can publish application changes that you make in App Studio to the pipeline. Publishing your changes creates a patch version of the application and starts a deployment. For example, you can change a life cycle, data model, or user interface elements in a screen and submit those changes to systems in the pipeline.

- Interpreting different publish statuses

App Studio provides various status updates throughout the publish cycle to inform you of various challenges or roadblocks your pipeline might encounter. These guardrails are intended to mitigate and resolve issues before publish failures.

Configuring App Studio for publishing

Publish application updates in App Studio using an existing deployment pipeline to seamlessly create patch versions of an application and start a deployment.

Before you begin: Validate the following:

- Pega Platform is on version 8.5.2 or higher.
- Deployment Manager is on 5.2 or higher.

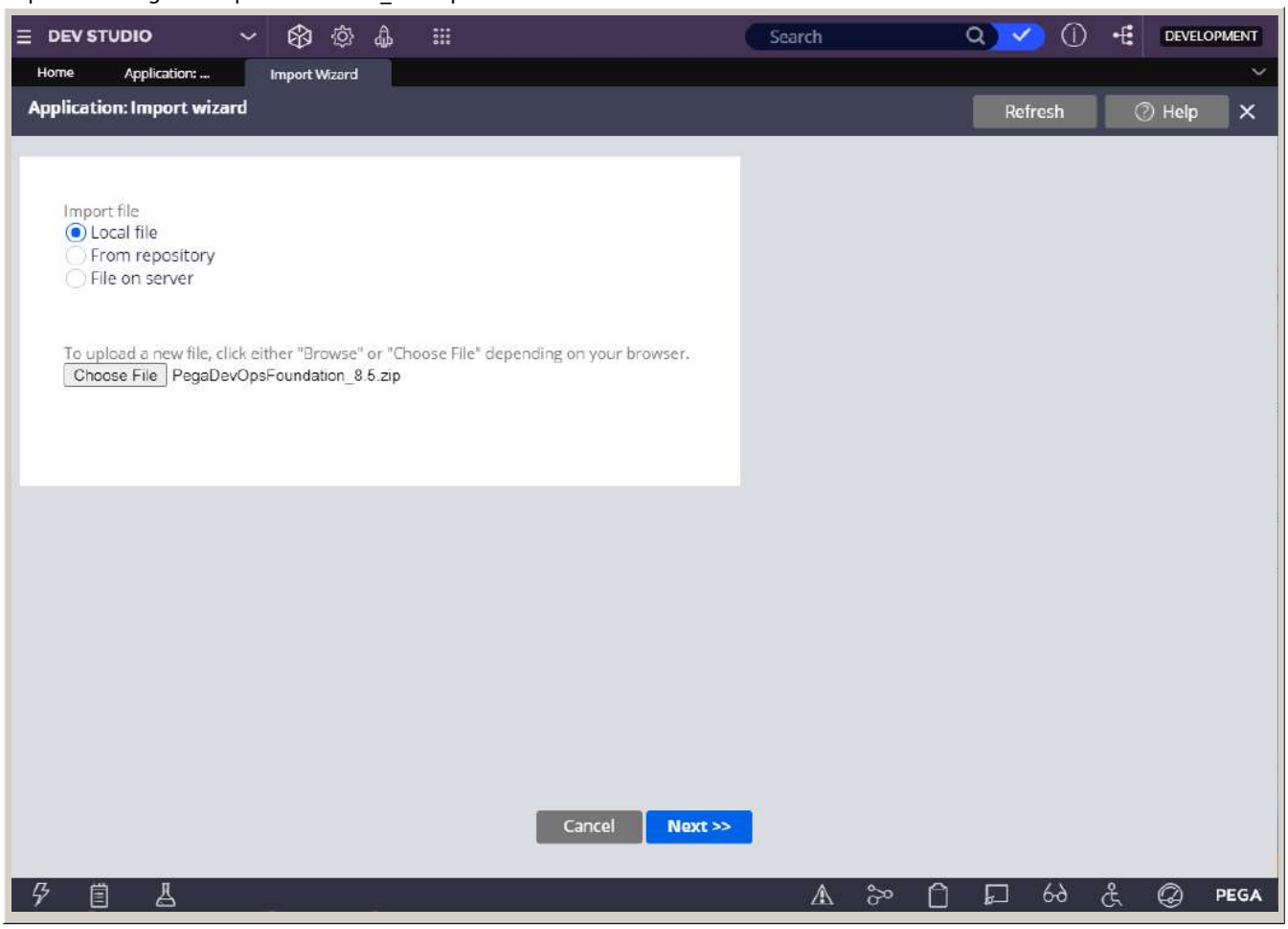
This publishing capability requires your candidate environments communicate with Deployment Manager. therefore you must follow the configuration steps below before proceeding.

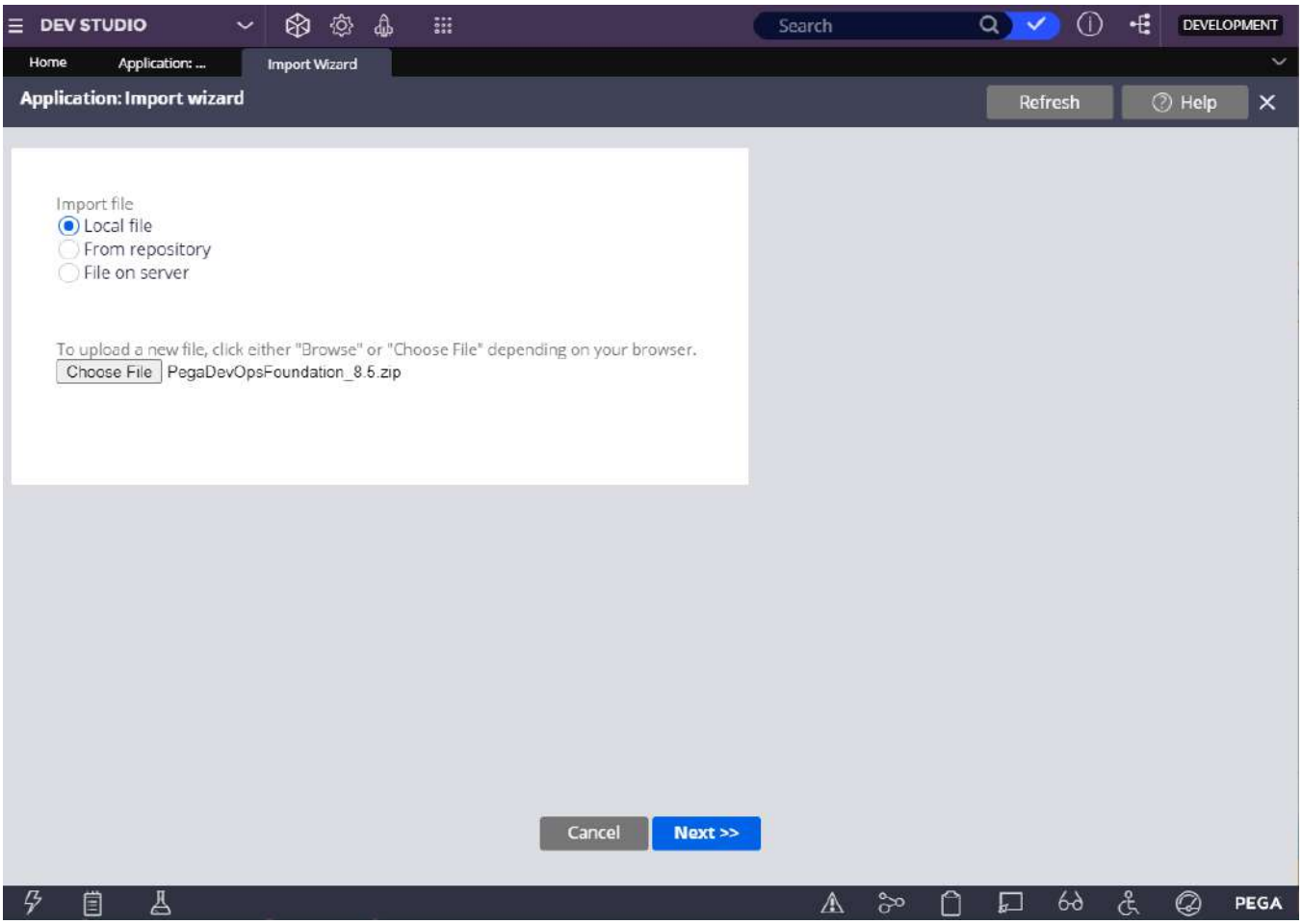
Enable Deployment Manager integration

You must properly configure and integrate Deployment Manager with App Studio before you can publish changes. These integration steps enable communication between the candidate environments and the orchestrator for seamless publishing of deployment pipelines.

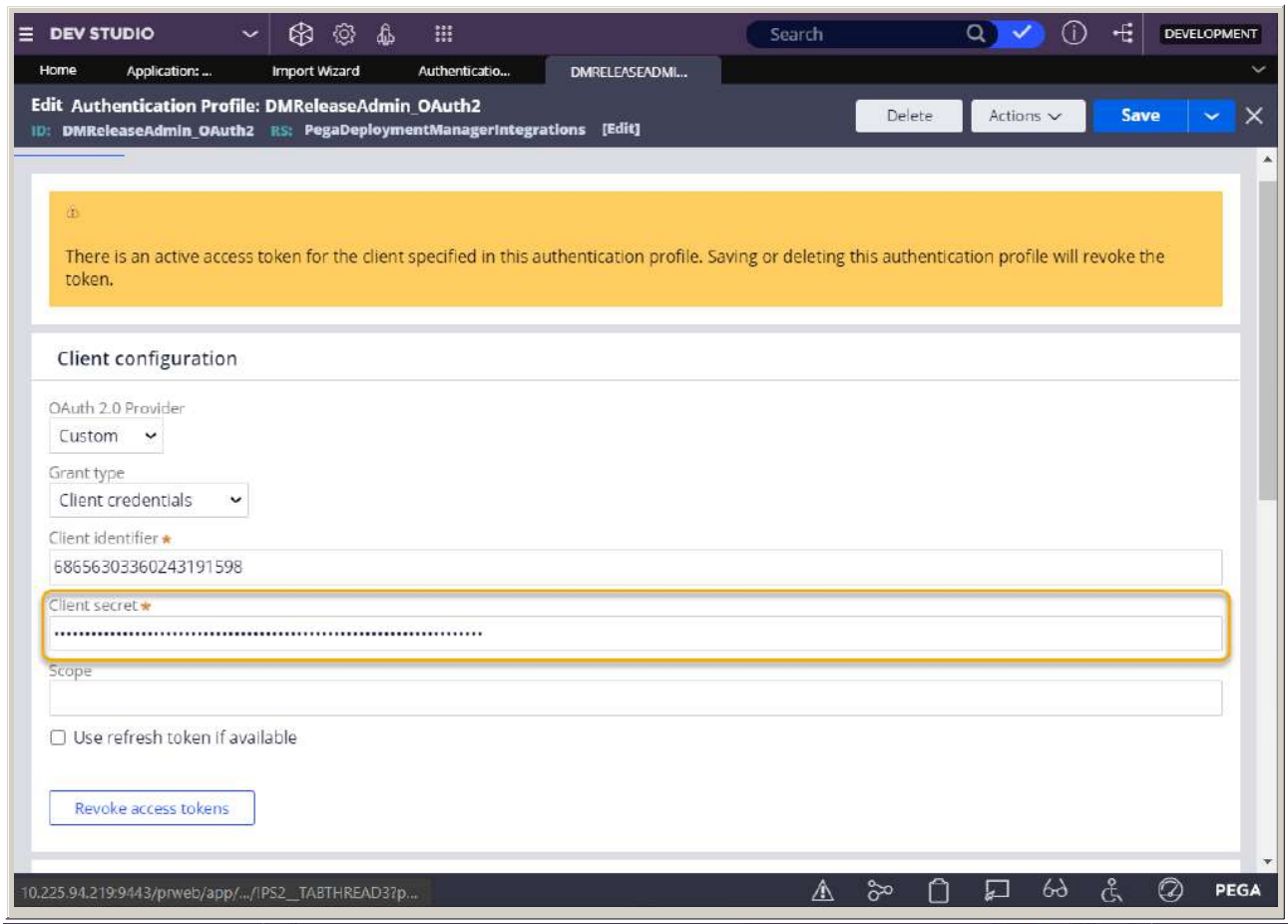
To enable Deployment Manager integration on the App Studio environment, follow the steps below:

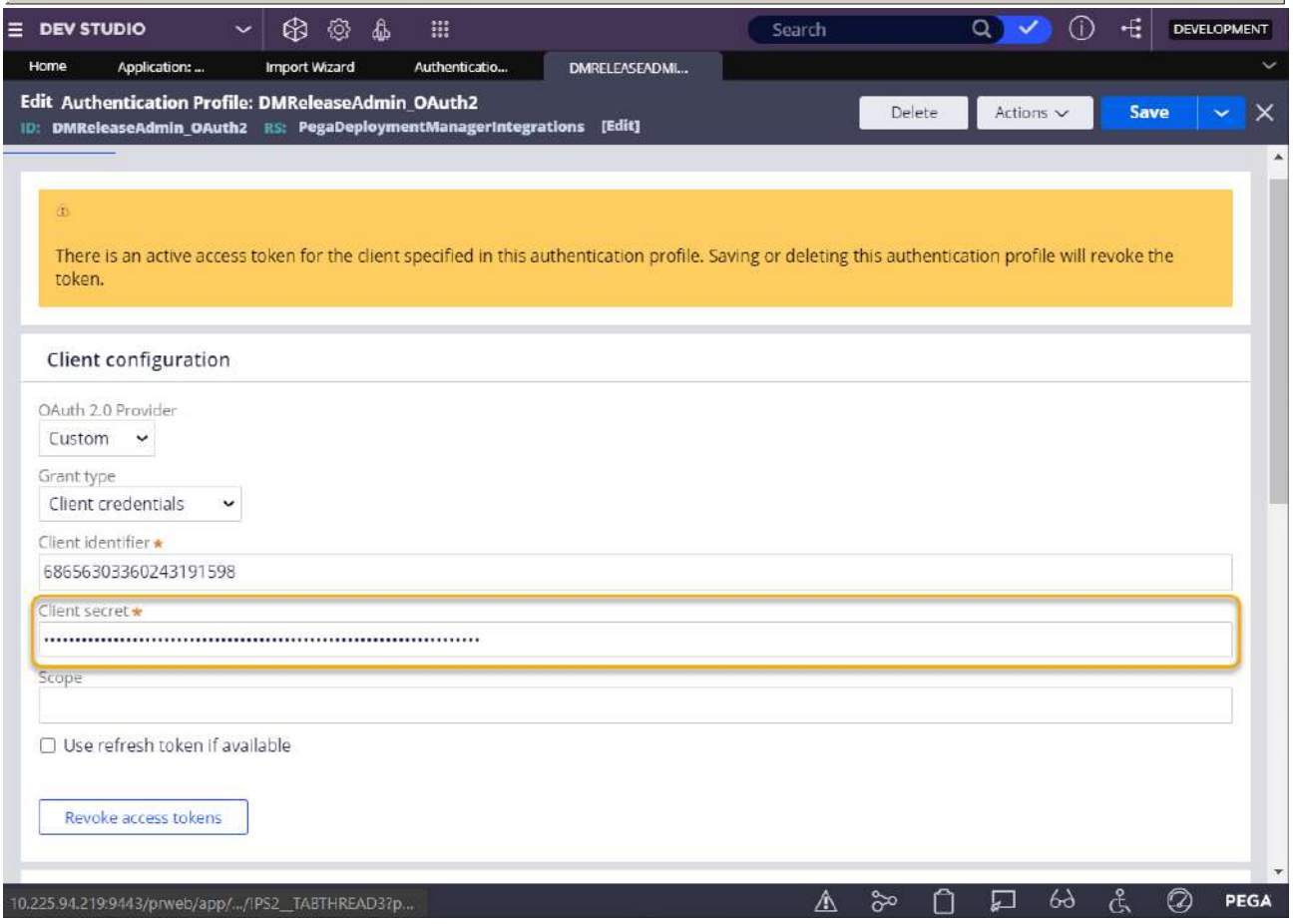
1. Navigate to the Deployment Manager Pega Marketplace page and download the latest version of Deployment Manager.
2. Extract the contents of the downloaded file.
3. Use the Import wizard to import files into the appropriate systems. For more information on using the Import wizard, see Importing rules and data by using the Import wizard.
4. On the development environment, switch to Dev Studio and click **Configure Application Distribution Import** and import the PegaDevOpsFoundation_8.5.zip file.





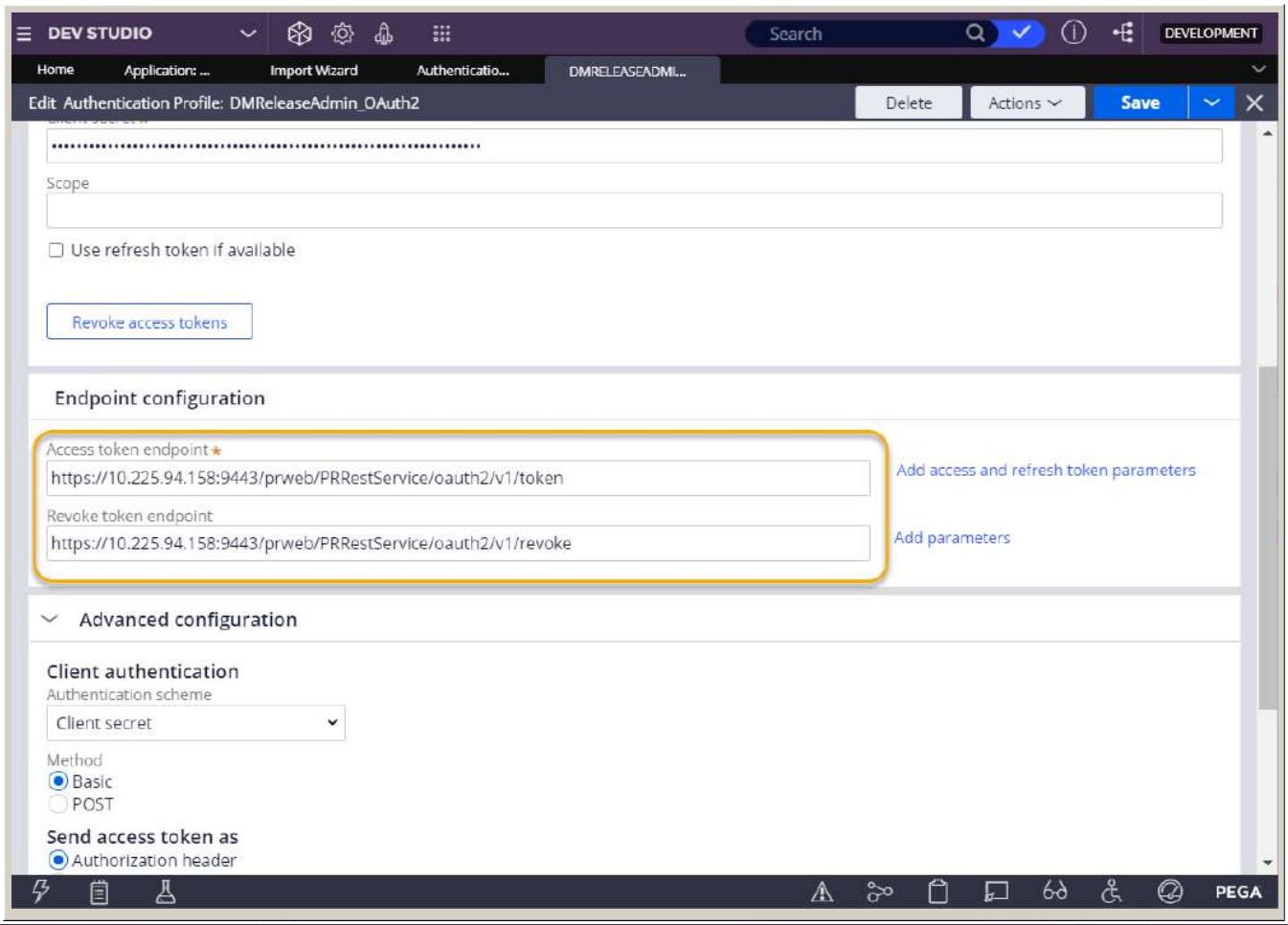
5. Next, configure the authentication profile. Deployment Manager cannot automatically populate the client secret to candidate environments, as it is not recommended to share this information across systems. To manually update the client secret information on your candidate environment:
    a. In Dev Studio, click **Records Explorer Security Authentication Profile** to receive a list of profile names available on the candidate environment.
    b. Select the DMReleaseAdmin_OAuth2 authentication profile.
    c. Update the client secret on the authentication profile. On the OAuth 2.0 tab, under the Client configuration section, enter the client secret in the **Client secret** field.

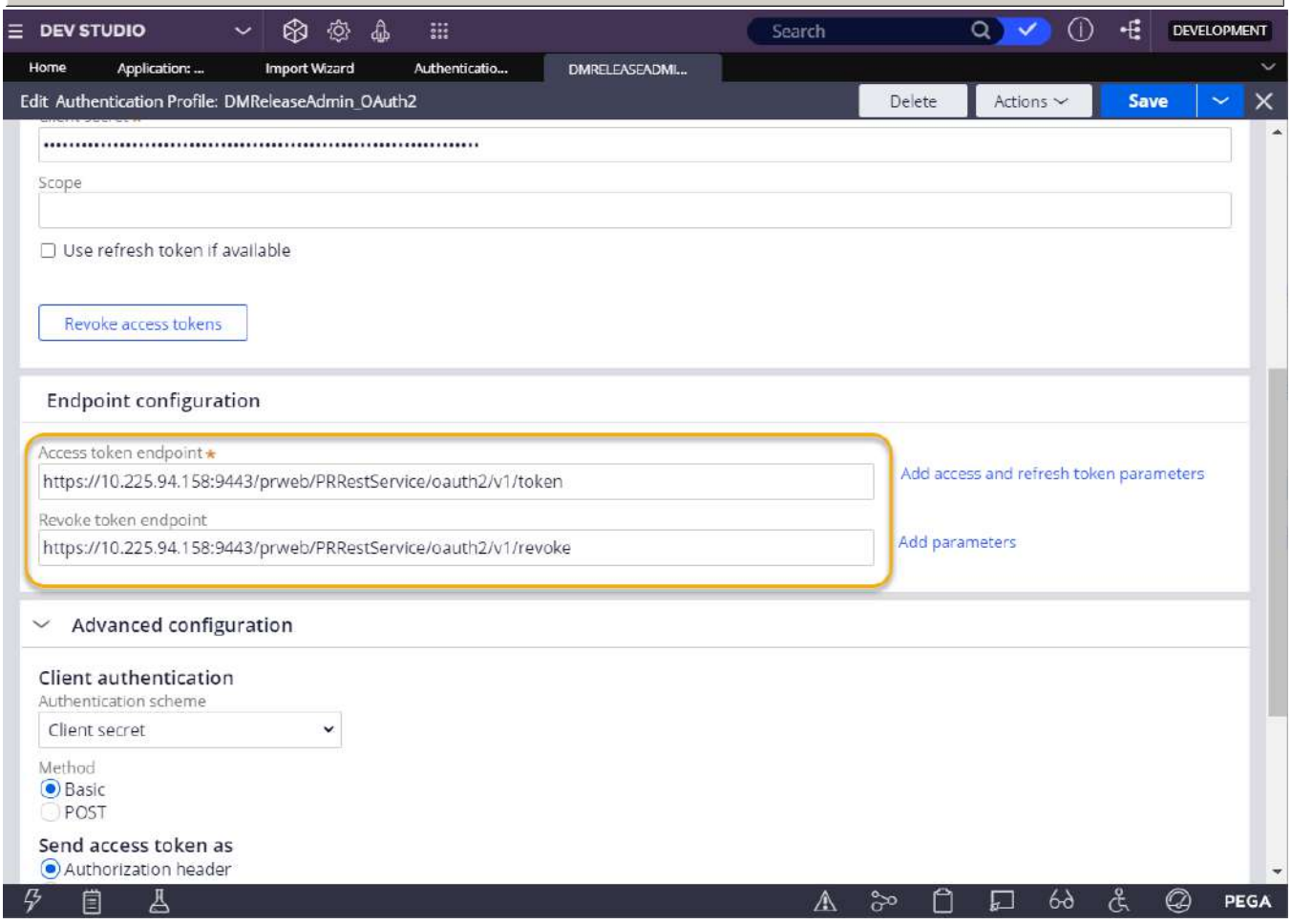




Note: The client secret is downloaded while setting up the orchestrator instance of Deployment Manager. Contact your Deployment Manager administrator to obtain the client secret.

6. Enter the **Access token endpoint** and the **Revoke token endpoint** in the Endpoint configuration section.





7. Confirm the changes by clicking **Save**.

8. To update the orchestrator server to manage this application deployment, you must update the **OrchestratorURL** on the environment.
    a. In the PegaDevopsShared ruleset, search OrchestratorURL Dynamic System Setting.
    b. Update the **Value** field to point to the orchestrator URL. Use this setting for Dev Studio and App Studio integration.
    c. Confirm the changes by clicking **Save**.

Configure a deployment pipeline

Create a deployment pipeline to model the release process set for your application or team. For more information on creating a deployment pipeline, see Creating a deployment pipeline.

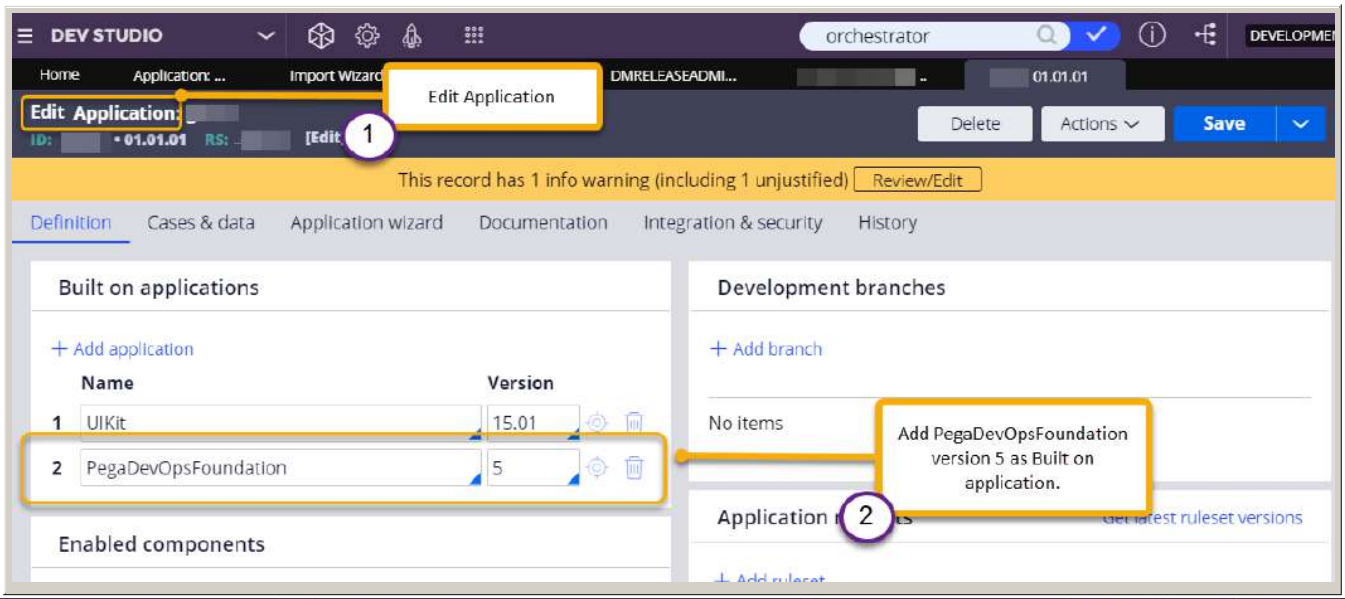
Ensure your pipeline has at least one quality assurance or staging stage with a manual task so that you do not deploy changes to production that have not been approved by stakeholders.

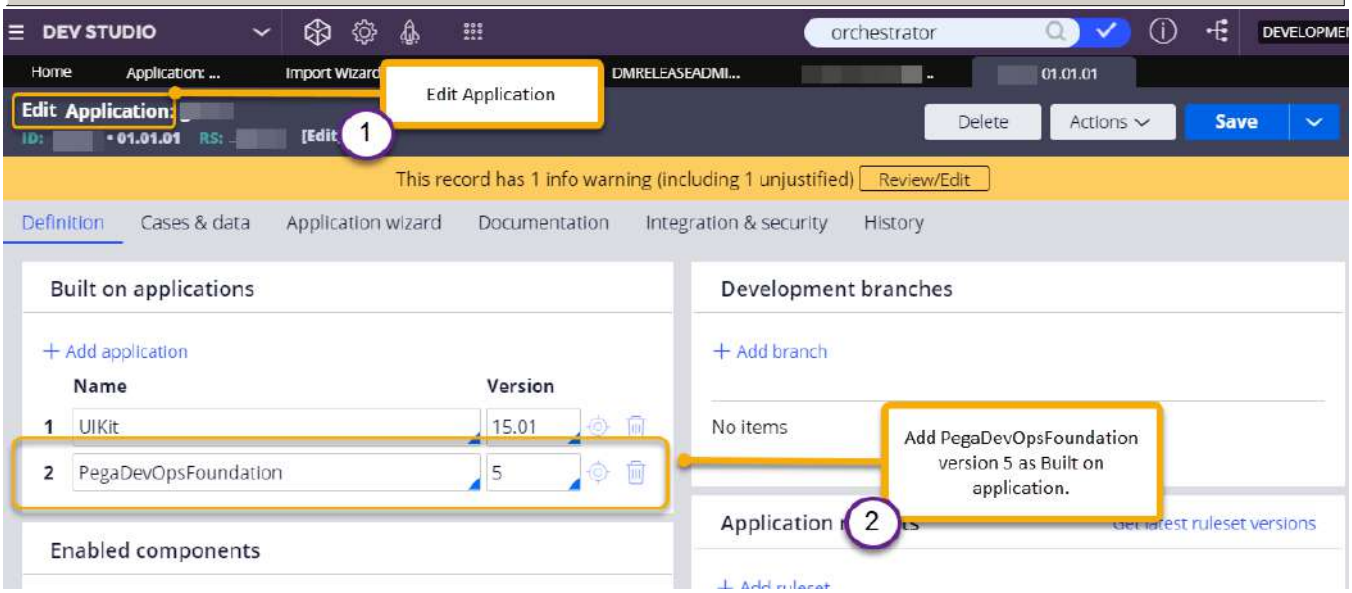
Configure an application to enable publishing

You must properly configure an application before you can publish updates using deployment pipelines.

To enable publishing on an application, follow the list below:

1. A product rule exists with the same name and version as the application being deployed. For more information see Creating a product rule that does not include associated data.
2. A pipeline has been created in Deployment Manager for the application being deployed.
3. There is at least one unlocked ruleset in the application.
4. The application is built on PegaDevOpsFoundation. Switch to Dev Studio, navigate from the Application drop down to definition. Add built-on application and enter PegaDevOpsFoundation under **name** field and update the major version used under **version**.





Configure user access

As an App Studio user, log in to the application you are deploying on the development system.

To enable access to deploy the changes, a user must have deployment-level access. Follow the steps below to enable accesses.

1. Switch to Deployment Manager from Dev Studio.
2. In the navigation pane of Deployment Manager, click **Users People** .
3. On the People page, click **Add user**.
4. In the Add user dialog box, click the **User** field and do one of the following actions:
   a. Select the user that you want to add.
   b. Enter an email address.
5. Click **Add**.
6. From the role list, select **PipelineUser**.
7. If you selected the App admin role or a custom role in the Applications field, enter the application name that the user can access.
8. Click **Send invite** to send an email containing the user name and a randomly generated password for Deployment Manager log in credentials.

Publishing application changes in App Studio

You can publish application changes that you make in App Studio to the pipeline. Publishing your changes creates a patch version of the application and starts a deployment. For example, you can change a life cycle, data model, or user interface elements in a screen and submit those changes to systems in the pipeline.

Before you begin: Review the steps in Configuring App Studio for publishing to ensure proper configurations of your systems before making application changes in App Studio. To publish an application change using App Studio:

1. Log in to your application in a development environment.
   Note: Application changes can only be published from a development environment.
2. In the top-right corner of App Studio, click **Development**.
   - If there are pending updates staged for publication, the application version states **Not published** in orange text.
   - If there are no updates, the application version states **No changes**.
3. Click **Manage versions**.
4. Your deployment pipeline is now visible under the Publish section, broken out into the pre-configured stages.
5. From the Development stage, click **Publish to [next stage]**.
   Note: Publishing a new application aborts all existing publishes in progress, queued, and pending-promotion versions, irrespective of phase or stage. Ensure there are no active publishes in your pipeline before starting a new one.
6. In the publish confirmation dialogue, enter a short description for the publish log. You can also add any associated user stories/bugs if you have Agile Workbench configured with App Studio.
7. Click **Ok**.
8. The publishing of your application now progresses through your deployment pipeline model.
   Note: If you have a manual transition configured, this is marked in the pipeline model by a dotted-line between stages. This requires you to take action from the Deployment Manager portal.
9. Click **Approve for production** to finalize the publish and stage for production. App studio only supports the approval of publishing. To cancel or abort your publishing, simply start a new publish to overwrite any existing publishes in the pipeline.

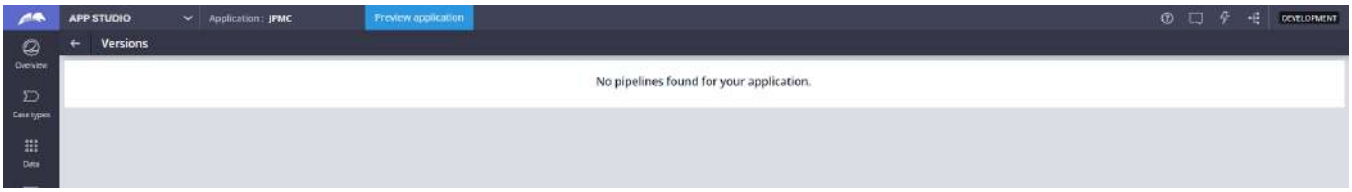
Interpreting different publish statuses

App Studio provides various status updates throughout the publish cycle to inform you of various challenges or roadblocks your pipeline might encounter. These guardrails are intended to mitigate and resolve issues before publish failures.

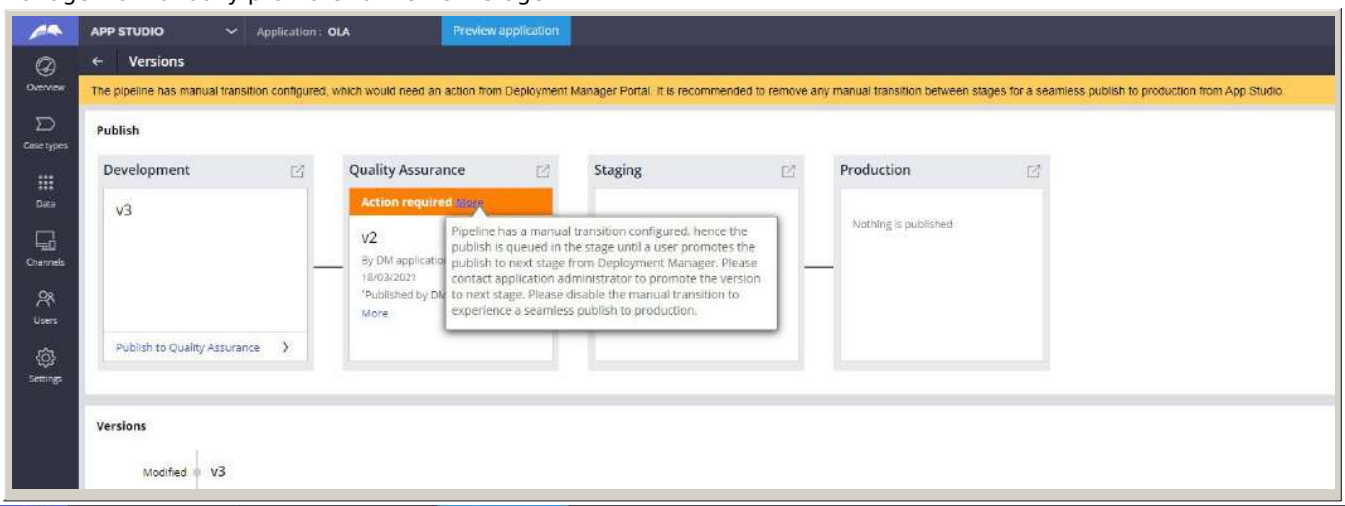
See below for a few examples of the App Studio publish guardrails:

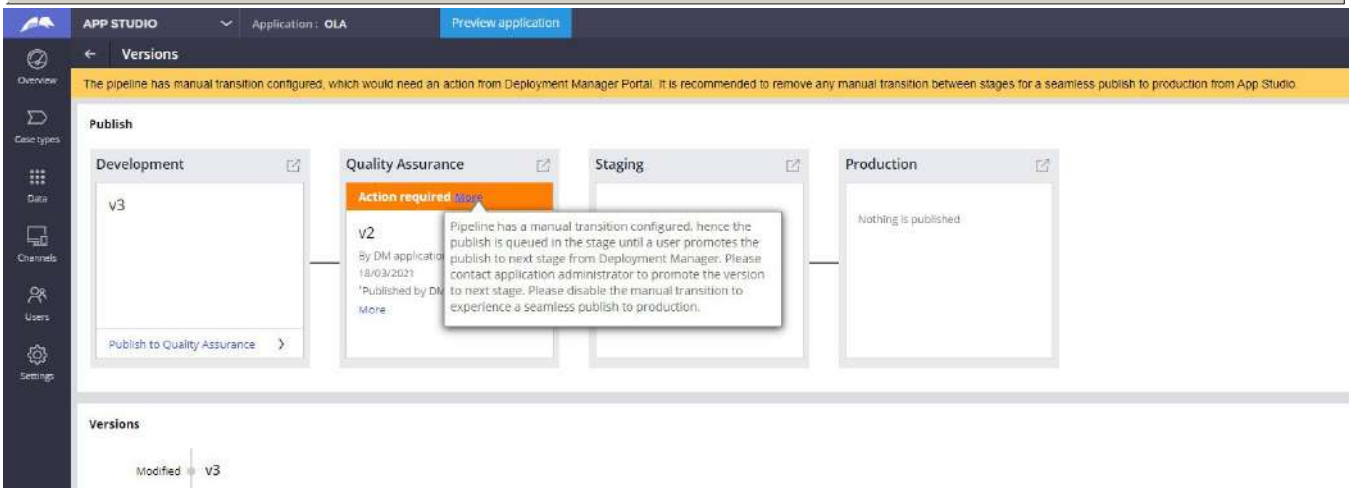
- App Studio can only publish if there is one configured deployment pipeline in Deployment Manager. App Studio returns an error if you attempt to publish with more than one deployment pipeline configured.

- If you have no pipelines configured, App Studio returns the following error. To add a pipeline in Deployment Manager, see Creating pipelines.

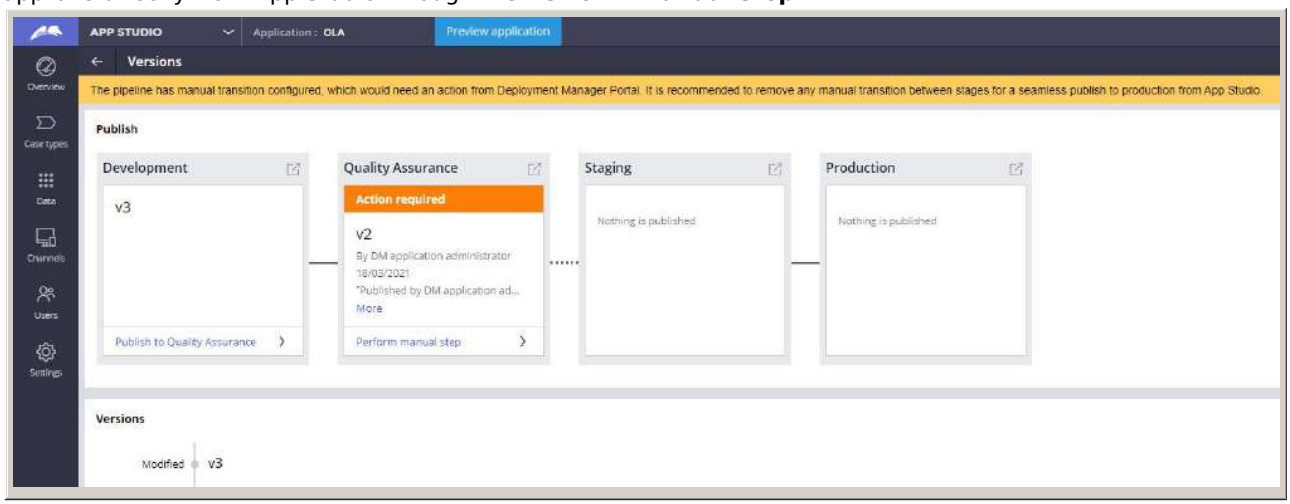


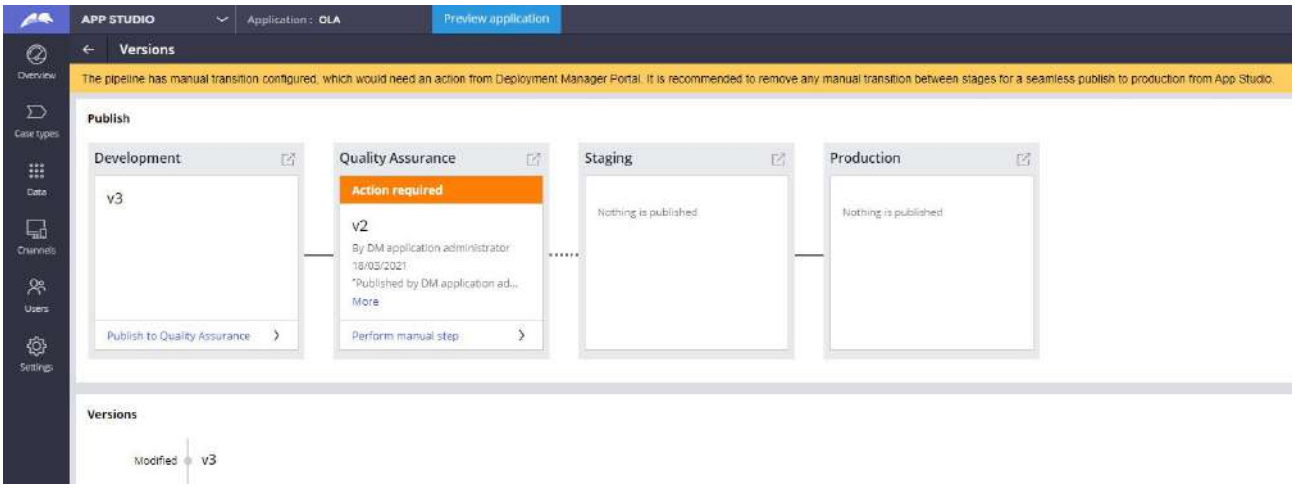
- It is not recommended to have a manual transition configured in your deployment pipeline. If a manual transition is required, you can configure this at the stage level. Once the stage encounters the manual transition (after all tasks complete in the stage), the **Action Required** alert is displayed on App Studio. You must switch to Deployment Manager to manually promote to the next stage.



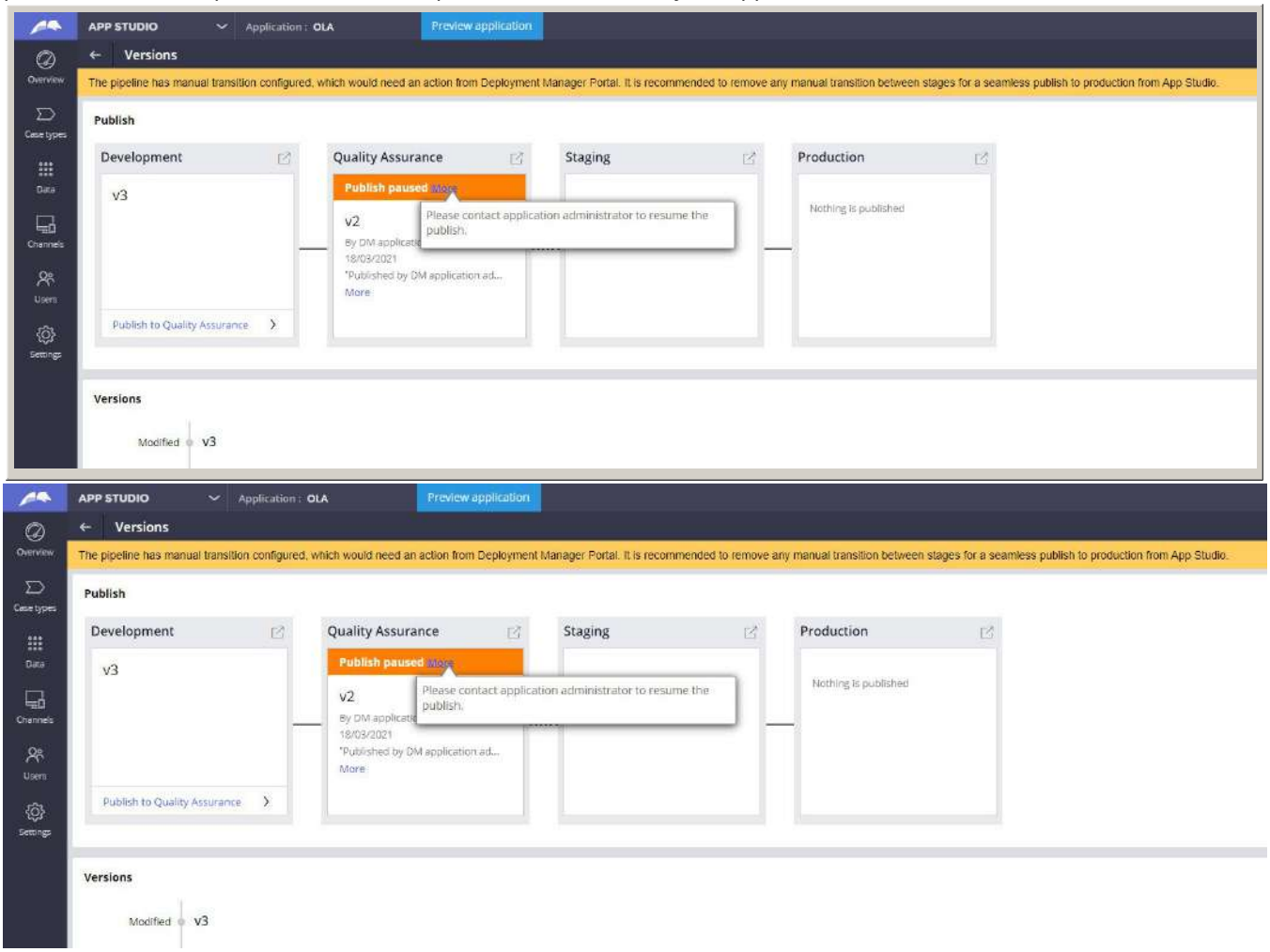


  - If there is any manual input or approval needed between tasks for the deployment to proceed, you can configure the Perform manual step task in the pipeline. If the deployment is waiting at Perform manual step, you can approve directly from App Studio through the **Perform manual step** link.

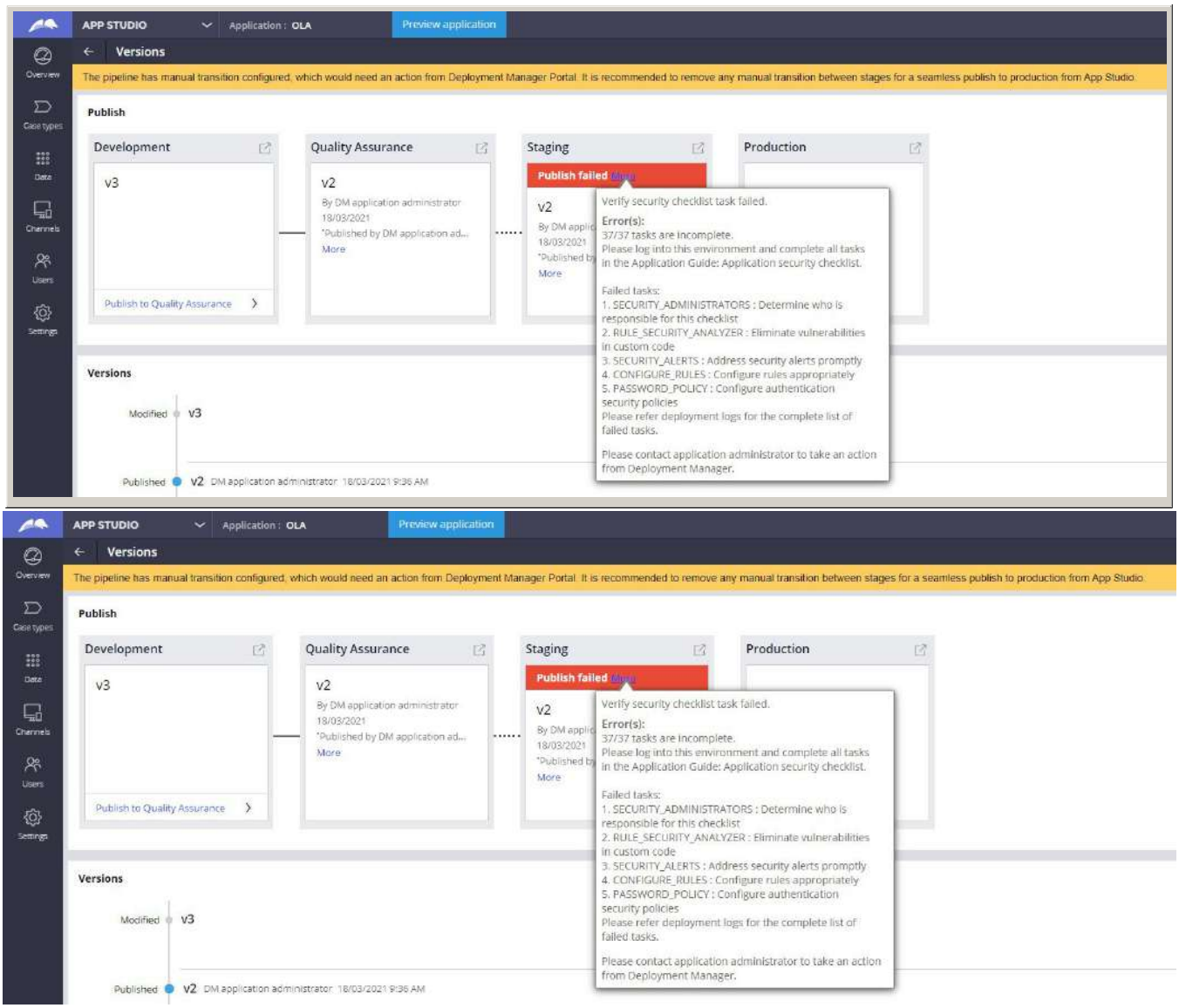




- If a publish is paused, the pipeline will reflect that status with a **Publish paused** warning on in the stage on which it paused. To resume publication from the paused state, contact your application administrator.



- If a task fails for any reason in the publish cycle, the application publish will stop. The pipeline model provides a error for the stage on which **Publish failed**, along with the list of failed tasks to remedy before attempting to publish your application again. Once the issue is resolved, contact your application administrator to access Deployment Manager and retry the failed task and resume publishing.



APP STUDIO
Application : OLA
Preview application
Versions
The pipeline has manual transition configured, which would need an action from Deployment Manager Portal. It is recommended to remove any manual transition between stages for a seamless publish to production from App Studio.
Publish
Development
v3
Publish to Quality Assurance
Quality Assurance
v2
By DM application administrator
18/03/2021
*Published by DM application ad...
More
Staging
Publish failed
v2
More
Production
Verify security checklist task failed.
Error(s):
37/37 tasks are incomplete.
Please log into this environment and complete all tasks in the Application Guide: Application security checklist.
Failed tasks:
1. SECURITY_ADMINISTRATORS : Determine who is responsible for this checklist
2. RULE_SECURITY_ANALYZER : Eliminate vulnerabilities in custom code
3. SECURITY_ALERTS : Address security alerts promptly
4. CONFIGURE_RULES : Configure rules appropriately
5. PASSWORD_POLICY : Configure authentication security policies
Please refer deployment logs for the complete list of failed tasks.
Please contact application administrator to take an action from Deployment Manager.
Versions
Modified v3
Published v2 DM application administrator 18/03/2021 9:36 AM

Troubleshooting issues with your pipeline

Deployment Manager provides several features that help you troubleshoot and resolve issues with your pipeline.

You can:

- View deployment logs for information about the completion status of operations.
- Run diagnostics to verify that your environment is correctly configured.
- Use a chatbot to obtain information about common issues.

- Viewing deployment logs in 5.1.x

  View logs for a deployment to see the completion status of operations, for example, when a deployment moves from staging to production. When the Deploy task runs, the application package is imported in to the candidate system. By default, logs record all the new rule and data instances and all the updated rule and data instances that are in this application package. You can disable the logging of such rule and data types and can change the logging level to control which events are displayed in the log.

- Diagnosing a pipeline in 5.1.x

  You can diagnose your pipeline to troubleshoot issues and verify that your pipeline is configured properly.

- Obtaining information about common issues by using the chatbot

  Deployment Manager provides a chatbot that you can use to obtain information about common issues, such as connectivity between systems, Jenkins configuration, and branch merging. After you enter your search text, the chatbot provides you with relevant answers and links to more information.

Viewing deployment logs in 5.1.x

View logs for a deployment to see the completion status of operations, for example, when a deployment moves from staging to production. When the Deploy task runs, the application package is imported in to the candidate system. By default, logs record all the new rule and data instances and all the updated rule and data instances that are in this application package. You can disable the logging of such rule and data types and can change the logging level to control which events are displayed in the log.

To view a deployment log, do the following steps:

1. In Dev Studio, on the appropriate candidate system, change the logging level to control which events the log displays. **For example:** For example, you can change logging levels of your deployment from INFO to DEBUG for troubleshooting purposes. For more information, see Logging Level Settings tool.
2. To disable logging of new and updated rule and data instances in imported application packages, perform the following steps:
    1. On the candidate system for which you want to disable reporting, in the navigation pane of Admin Studio, click **Resources Log categories** .
    2. On the **Log categories** page, for the *DeploymentManager.DeltaInstanceLogging* log level, click the **More** icon, and then click **Change logging level**.
    3. In the **Change pxBackgroundProcessing.Agents log level** dialog box, in the **Update log level of category to** list, select **OFF**.
    4. Click **Submit**.
3. If the pipeline is not open, in the navigation pane, click **Pipelines Application pipelines** .
4. Do one of the following actions:
    - To view the log for the current deployment, click the **More** icon, and then click **View logs**.
    - To view the log for a previous deployment, expand the **Deployment History** pane, and then click **Logs** for the deployment.

Diagnosing a pipeline in 5.1.x

You can diagnose your pipeline to troubleshoot issues and verify that your pipeline is configured properly.

For example, you can determine if the target application and product rule are in the development environment, connectivity between systems and repositories is working, and pre-merge settings are correctly configured.

1. If the pipeline is not open, in the navigation pane of Deployment Manager, click **Pipelines** and then click the name of the pipeline.
2. Click **Actions Diagnose pipeline** .
3. In the **Diagnostics** window, review the errors, if any.
   Note: If the OrchestratorURL dynamic system setting is not configured, Deployment Manager displays a message that you can disregard if you are not using branches, because you do not need to configure the dynamic system setting.

Obtaining information about common issues by using the chatbot

Deployment Manager provides a chatbot that you can use to obtain information about common issues, such as connectivity between systems, Jenkins configuration, and branch merging. After you enter your search text, the chatbot provides you with relevant answers and links to more information.

Before you begin: If the chatbot is disabled, enable it. For more information, see Enabling and disabling the chatbot. To use the Deployment Manager chatbot to help resolve issues, perform the following steps:

1. In the bottom right corner of the Deployment Manager portal, click the chatbot icon.
2. Do one of the following actions:
    a. Click the appropriate link from the list of issues that the chatbot displays.
    b. Enter text for which you want to receive more information, and then click **Enter**.
3. To clear the chatbot history, in the chatbot window, click the **More** icon, and then click **Clear chat history**.

- Enabling and disabling the chatbot

  Use the chatbot to obtain more information about common Deployment Manager issues, such as branch merging and pipeline configuration. You can disable and enable the chatbot. By default, the chatbot is enabled.

Enabling and disabling the chatbot

Use the chatbot to obtain more information about common Deployment Manager issues, such as branch merging and pipeline configuration. You can disable and enable the chatbot. By default, the chatbot is enabled.

Only super administrators can enable and disable the chatbot. For more information about user roles, see Setting up users and roles.

1. In the navigation pane of Deployment Manager click **Settings General settings** .
2. Do one of the following actions:
    - To enable the chatbot, select the **Enable self-service Deployment Manager web chatbot** check box.
    - To disable the chatbot, clear the check box.

3. Click **Save**.
4. At the top of the **General Settings** page, click the **Page back** icon.
5. Click the **Refresh** icon to refresh Deployment Manager and apply your changes.

Configuring additional settings in 5.1.x

As part of your pipeline, users can optionally receive notifications through email when events occur. For example, users can receive emails when tasks or pipeline deployments succeed or fail. For more information about the notifications that users can receive, see Understanding email notifications.

For either new Deployment Manager installations or upgrades, you must configure settings on the orchestration server so that users can receive email notifications. For more information, see Configuring email accounts on the orchestration server.

Additionally, you can configure Jenkins if you are using Jenkins tasks in a pipeline. For more information, see Configuring Jenkins in 5.1.x.

- Configuring email accounts on the orchestration server

  Deployment Manager leverages the Default email account from 5.x version and the DMEmailListener email listener. If you are configuring email accounts for the first time, specify your details for this account in Pega Platform. For more information, see .

- Configuring Jenkins in 5.1.x

  If you are using a Run Jenkins step task in your pipeline, configure Jenkins so that it can communicate with the orchestration server.

- Configuring Deployment Manager notifications

  You can enable notifications to receive updates about the events that occur in your pipeline. For example, you can choose to receive emails about whether unit tests failed or succeeded. You can receive notifications in the Deployment Manager notifications gadget, through email, or both. By default, all notifications are enabled for users who are configured in Deployment Manager.

- Dynamic system settings

  Deployment Manager exposes a few Dynamic System Settings (DSS) to override the default behavior. See below for a list of configurations available to override the default behavior. All of these settings are optional and are not required by default. Carefully evaluate each configuration before configuring for your application.

- Creating and using custom repository types for Deployment Manager

  In Deployment Manager 3.1.x and later, you can create custom repository types to store and move your artifacts. For example, you can create a Nexus repository and use it similarly to how you would use a Pega Platform-supported repository type such as file system. By creating custom repository types, you can extend the functionality of Deployment Manager through the use of a wider variety of repository types with your artifacts.

- Managing test cases separately in Deployment Manager

  Use Deployment Manager to package and deploy test cases separately on the candidate systems in the pipeline. When you configure a pipeline in Deployment Manager, you specify the details of the test package that you want to deploy.

- Configuring SSO and LDAP

  Deployment Manager supports LDAP and Single Sign-On (SSO) authentication, and assigns users roles based on the configuration settings in the active directory and SSO.

Configuring email accounts on the orchestration server

Deployment Manager leverages the Default email account from 5.x version and the DMEmailListener email listener. If you are configuring email accounts for the first time, specify your details for this account in Pega Platform. For more information, see Configuring email accounts for new Deployment Manager installations.

- Configuring email accounts for new Deployment Manager installations

  For new Deployment Manager installations, on the orchestration server, configure the Default email account so users can receive email notifications for events such as task completion or failure.

- Configuring email accounts when upgrading Deployment Manager

  If you are upgrading to latest version of Deployment Manager, you must perform certain steps so that you can send email notifications. Default email account is suggested to be used in version 5.x of Deployment Manager that listens to DMEmailListener email listener. If you are using Pega-Pipeline-CD email account, you should switch to the Default

email account.

- Understanding email notifications

  Emails are preconfigured with information about each notification type. For example, when a deployment failure occurs, the email that is sent provides information, such as the pipeline name and URL of the system on which the deployment failure occurred.

Configuring email accounts for new Deployment Manager installations

For new Deployment Manager installations, on the orchestration server, configure the Default email account so users can receive email notifications for events such as task completion or failure.

Do the following steps:

1. In the navigation pane of Dev Studio, click **Records**, and then click **Integration-Resources Email Account** .
2. Click **Default**.
3. In the **Edit Email Account** rule form, configure and save the email account.
   For more information about configuring email accounts, see Creating an email account in Dev Studio.

Configuring email accounts when upgrading Deployment Manager

If you are upgrading to latest version of Deployment Manager, you must perform certain steps so that you can send email notifications. Default email account is suggested to be used in version 5.x of Deployment Manager that listens to DMEmailListener email listener. If you are using Pega-Pipeline-CD email account, you should switch to the Default email account.

Do the following steps:

1. Update the email sender and recipient in Pega Platform.
   a. In the navigation pane of Dev Studio,, click **Records**, and then click **Integration-Resources Email Account** .
   b. Click **Default**.
   c. On the **Edit Email Account** form, configure and save the email account.
      For more information about configuring email accounts, see Creating an email account in Dev Studio
2. If you have an email listener other than DMEmaillistener that listens to the same email address that you configured in Deployment Manager in the previous step, delete the other listener.

Understanding email notifications

Emails are preconfigured with information about each notification type. For example, when a deployment failure occurs, the email that is sent provides information, such as the pipeline name and URL of the system on which the deployment failure occurred.

Preconfigured emails are sent in the following scenarios:

Email notification events

Event	Context	Description
Deployment start	Deployment	When a deployment starts, an email is sent to the release manager. If you are using branches, the email is sent to the operator who started the deployment.
Deployment step completion or failure	Deployment	When a step either completes or fails, an email is sent to the release manager. If you are using branches, the email is also sent to the operator who started the branch merge. The deployment pauses if there are any errors.
Deployment completion	Deployment	When a deployment is successfully completed, an email is sent to the release manager. If you are using branches, the email is sent to the operator who started the branch merge.
		When a stage in a deployment process either succeeds or fails,

Event	Context	Description
Stage completion or failure	Stage	an email is sent to the release manager. If you are using branches, the email is sent to the operator who started the branch merge.
Manual tasks requiring approval	Task input required	When a manual task requires email approval from a user, an email is sent to the user. That user can approve or reject the task from the email.
Stopped deployment	Task	When a deployment is stopped, an email is sent to the release manager. If you are using branches, the email is sent to the operator who started the branch merge.
PegaUnit testing success or failure	Task	If you are using the Run Pega unit tests task, and the task either succeeds or fails, an email is sent to the release manager. If you are using branches, the email is sent to the operator who started the branch merge.
Schema changes required	Task	If you do not have the required schema privileges to deploy schema changes on application packages that require those changes, an email is sent to the operator who started the deployment.
Guardrail compliance score success or failure	Task input required	If you are using the Check guardrail compliance task, an email is sent to the release manager if the task either succeeds or fails.
Approve for production	Task	If you are using the Approve for production task, which requires approval from a user before application changes are deployed to production, an email is sent to the user. The user can reject or approve the changes.
Verify security checklist success or failure	Task	If you are using the Verify security checklist task, which requires that all tasks be completed in the Application Security Checklist to ensure that the pipeline complies with security best practices, an email is sent to the release manager if the test either succeeds or fails.
Scenario testing success or failure	Task	If you are using the Run Pega scenario tests task, an email is sent to the release manager. If you are using branches, the email is sent to the operator who started the branch merge if Pega scenario testing either succeeds or fails.
Start test coverage success or failure	Task	If you are using the Enable test coverage task to generate a test coverage report, an email is sent to the release manager if the task either fails or succeeds.
Verify test coverage success or failure	Task	If you are using the Verify test coverage task, an email is sent to

Event	Context	Description
		the release manager if the task either fails or succeeds.
Application quality statistics refreshed	Task	If you are using the Refresh application quality statistics task, an email is sent to the release manager when the task is run.
Jenkins job success or failure	Task	If you are using a Jenkins task, an email is sent to the release manager if a Jenkins job either succeeds or fails.

Configuring Jenkins in 5.1.x

If you are using a Run Jenkins step task in your pipeline, configure Jenkins so that it can communicate with the orchestration server.

1. On the orchestration server, create an authentication profile that uses Jenkins credentials.
   - If you are using a version of Jenkins earlier than 2.17.6, create an authentication profile on the orchestration server that specifies the credentials to use.
     1. Click **Create Security Authentication Profile** .
     2. Enter a name, and then click **Create and open**.
     3. In the **User name** field, enter Jenkins user ID.
     4. Click **Set password**, enter the Jenkins password, and then click **Submit**.
     5. Click the **Preemptive authentication** check box.
     6. Click **Save**.
     7. Go to step 4.

     For more information about configuring authentication profiles, see Creating an authentication profile.
   - If you are using Jenkins 2.17.6 or later and want to use an API token for authentication, go to step 2.
   - If you are using Jenkins 2.17.6 or later and want to use a Crumb Issuer for authentication, go to step 3.
2. If you are using Jenkins version 2.17.6 or later and want to use an API token for authentication, do the following steps:
   a. Log in to the Jenkins server.
   b. Click **People**, click the user who is running the Jenkins job, and then click **Configure API token** .
   c. Generate the API token.
   d. Create an authentication profile on the orchestration server by clicking **Create Security Authentication Profile** .
   e. In the **User name** field, enter the Jenkins user ID.
   f. Click **Set password**, enter the API token that you generated, and then click **Submit**.
   g. Click the **Preemptive authentication** check box.
   h. Click **Save**.
   i. Go to step 4.

   For more information about configuring authentication profiles, see Creating an authentication profile.
3. If you are using Jenkins version 2.17.6 or later and want to use a Crumb Issuer for authentication, do the following steps:
   a. Log in to the Jenkins server.
   b. Click **Manage Jenkins Manage Plugins** and select the check box for the **Strict Crumb Issuer** plug-in.
   c. Click **Manage Jenkins Configure Global Security** .
   d. In the **CSRF protection** section, in the **Crumb Issuer** list, select **Strict Crumb Issuer**.
   e. Click **Advanced**, and then clear the **Check the session ID** check box.
   f. Click **Save**.
   g. Create an authentication profile on the orchestration server by clicking **Create Security Authentication Profile** .
   h. In the **User name** field, enter the Jenkins user ID.
   i. Click **Set password**, enter the Jenkins password, and then click **Submit**.
   j. Click the **Preemptive authentication** check box.
   k. Click **Save**.
   l. Go to step 4.

   For more information about configuring authentication profiles, see Creating an authentication profile.

   The following steps describe one of the ways to configure Jenkins jobs for the integration with Deployment Manager to work (using Jenkins Free Style Project as the option). This can be done in multiple different ways (e.g. with workflow plugin and groovy, or scripts in Ant, Gradle, etc.).
4. Install the Post build task plug-in.
5. Install the curl and jq commands on the Jenkins server. As mentioned above, these can be replaced with any other scripts as well.
6. Create a new freestyle project.
7. On the **General** tab, select the **This project is parameterized** check box.
8. Add the DeploymentID and CallBackURL parameters.

a. Click **Add parameter**, and then select **String parameter**.
b. In the **String** field, enter DeploymentID.
c. Click **Add parameter**, and then select **String parameter**.
d. In the **String** field, enter CallBackURL.

9. To add parameters that you can use in Run Jenkins step tasks in the pipeline, click **Add parameter**, select **String parameter**, and enter the string of the parameter. The system automatically populates these values in Jenkins tasks. You can add any of the following strings:
   a. If you are configuring Jenkins tasks in a merge pipeline, add any of the following strings:
      - **DeploymentID**: Deployment ID of the pipeline on which the task is triggered.
      - **DeploymentNumber**: Sequence number of the deployment.
      - **TaskID**: TaskID of the Jenkins task.
      - **CallBackURL**: URL to post the status of task.
      - **OrchestratorURL**: URL on which the Jenkins task is configured.
      - **PipelineName**: Pipeline name on which the Jenkins task is configured.
      - **PipelineID**: ID of the pipeline on which the Jenkins task is configured.
      - **ApplicationName**: Application name for which the pipeline is configured.
      - **ApplicationVersion**: Application version for which the pipeline is configured.
      - **RepositoryName**: Repository to publish the merged branch.
      - **BranchName**: Name of the branch for which the merge Jenkins task is configured.
      - **BranchFilePath**: File path to the branch artifact.
   b. If you are configuring Jenkins tasks in a deployment pipeline, add any of the following strings:
      - **DeploymentID**: Deployment ID of the pipeline on which the task is triggered.
      - **DeploymentNumber**: Sequence number of the deployment.
      - **TaskID**: TaskID of the Jenkins task.
      - **CallBackURL**: URL to post the status of task.
      - **OrchestratorURL**: URL on which the Jenkins task is configured.
      - **PipelineName**: Pipeline name on which the Jenkins task is configured.
      - **PipelineID**: ID of the pipeline on which the Jenkins task is configured.
      - **RepositoryName**: Repository to publish the merged branch.
      - **DeploymentArtifactName**: Artifact name that the Deploy task uses on the stage on which the Jenkins task is configured.
      - **ArtifactPath**: Full path to the artifact that the Deploy task uses.
      - **CurrentStage**: Name of the stage on which the Jenkins task is configured.
      - **CurrentStageURL**: URL of the system on which the Jenkins task is configured.
10. In the **Build Triggers** section, select the **Trigger builds** remotely check box.
11. In the **Authentication Token** field, select the token that you want to use when you start Jenkins jobs remotely.
12. In the **Build Environment** section, select the **Use Secret text(s) or file(s)** check box.
13. In the **Bindings** section, do the following actions:
    a. Click **Add**, and then select **User name and password (separated)**.
    b. In the **Username Variable** field, enter client_id.
    c. In the **Password Variable** field, enter client_secret.
    d. In the **Credentials** field, click **Specific credentials**.
    e. Click **Add**, and then select **Jenkins**.
    f. In the **Add credentials** dialog box, in the **Username** field, enter the client id configured in the oAuth client credentials on Deployment Manager.
    g. In the **Password** field, enter the client secret.
    h. Optionally enter a description, and click **Save**.
14. Add post-build tasks by doing one of the following actions:
    a. If Jenkins is running on Microsoft Windows, go to step 15.
    b. If Jenkins is running on Linux, go to step 16.
15. If Jenkins is running on Microsoft Windows, add the following post-build tasks:
    a. Click **Add post-build** action, and then select **Post build task**.
    b. In the **Post-Build Actions** section, in the **Log text** field, enter a unique string for the message that is displayed in the build console output when a build fails, for example BUILD FAILURE**.**
    c. In the **Script** field, enter @echo off for /F %%I in ('curl --insecure -d "client_id=%%client_id%%&client_secret=%%client_secret%%&grant_type=client_credentials" https://10.224.203.11:8443/prweb/PRRestService/oauth2/v1/token ^| jq -r .access_token') do set token=%%~I curl --insecure -H "Content-Type: application/json" -H "Accept: application/json" -H "Authorization:Bearer %token%" -k -X PUT --data "{\"taskStatus\":\"Resolved-Rejected\",\"errors\": [{\"errorMessage\":\"Jenkins job failed. Check here for logs: %BUILD_URL%}],\"taskInfo\":{\"outputParameters\":[{\"name\": \"BuildNumber\",\"type\": \"Text\",\"value\": \"%BUILD_NUMBER%\"},{\"name\": \"JenkinsBuildURL\",\"type\": \"Text\",\"value\": \"%BUILD_URL%\"}]}}" %CallBackURL%**.**
    d. Click **Add another task**.
    e. In the **Post-Build Actions** section, in the **Log text** field, enter a unique string for the message that is displayed in the build console output when a build is successful, for example BUILD SUCCESS**.**
    f. In the **Script** field, enter @echo off for /F %%I in ('curl --insecure -d "client_id=%%client_id%%&client_secret=%%client_secret%%&grant_type=client_credentials" https://10.224.203.11:8443/prweb/PRRestService/oauth2/v1/token ^| jq -r .access_token') do set token=%%~I curl --insecure -H "Content-Type: application/json" -H "Accept: application/json" -H "Authorization:Bearer %token%" -k -X PUT --data "{\"taskStatus\":\"Resolved-Completed\",\"taskInfo\": {\"outputParameters\":[{\"name\": \"BuildNumber\",\"type\": \"Text\",\"value\": \"%BUILD_NUMBER%\"},{\"name\": \"JenkinsBuildURL\",\"type\": \"Text\",\"value\": \"%BUILD_URL%\"}]}}" %CallBackURL%
    g. Click **Save**.
    h. Go to step 17.
16. If Jenkins is running on Linux, add the following post-build tasks. Use the dollar sign ($) instead of the percent sign (%) to access the environment variables:

a. Click **Add post-build action**, and then select **Post build task**.
b. In the **Log text** field, enter a unique string that for the message that is displayed in the build console output when a build fails, for example BUILD FAILURE**.**
c. In the **Script** field, enter export token=`curl --insecure -d "client_id=${client_id}&client_secret=${client_secret}&grant_type=client_credentials" https://10.224.203.11:8443/prweb/PRRestService/oauth2/v1/token | jq -r .access_token` curl --insecure -H "Content-Type: application/json" -H "Accept: application/json" -H "Authorization:Bearer ${token}" -k -X PUT --data "{\"taskStatus\":\"Resolved-Rejected\",\"errors\":[{\"errorMessage\":\"Jenkins job failed. Check here for logs: ${BUILD_URL}\"}],\"taskInfo\":{\"outputParameters\":[{\"name\": \"BuildNumber\",\"type\": \"Text\",\"value\": \"${BUILD_NUMBER}\"}, {\"name\": \"JenkinsBuildURL\",\"type\": \"Text\",\"value\": \"${BUILD_URL}\"}]}}" $CallBackURL
d. Click **Add another task**.
e. In the **Log text** field, enter a unique string that for the message that is displayed in the build console output when a build is successful, for example BUILD SUCCESS**.**
f. In the **Script** field, enter export token=`curl --insecure -d "client_id=${client_id}&client_secret=${client_secret}&grant_type=client_credentials" https://10.224.203.11:8443/prweb/PRRestService/oauth2/v1/token | jq -r .access_token` curl --insecure -H "Content-Type: application/json" -H "Accept: application/json" -H "Authorization:Bearer ${token}" -k -X PUT --data "{\"taskStatus\":\"Resolved-Completed\",\"taskInfo\":{\"outputParameters\":[{\"name\": \"BuildNumber\",\"type\": \"Text\",\"value\": \"${BUILD_NUMBER}\"},{\"name\": \"JenkinsBuildURL\",\"type\": \"Text\",\"value\": \"${BUILD_URL}\"}]}}" $CallBackURL
g. Click **Save**.
h. Go to step 17.

17. To stop a pipeline deployment if a Jenkins build fails, add a post-build script:
    a. Click **Add post-build action**, and then select **Post build task**.
    b. In the **Log text** field, enter a unique string for the message that is displayed in the build console output when a build fails, for example JENKINS BUILD FAILURE**.**
    c. In the **Script** field, enter (for Windows) for /F %%I in ('curl --insecure -d "client_id=%%client_id%%&client_secret=%%client_secret%%&grant_type=client_credentials" https://10.224.203.11:8443/prweb/PRRestService/oauth2/v1/token ^| jq -r .access_token') do set token=%%~I curl --insecure -H "Content-Type: application/json" -H "Accept: application/json" -H "Authorization:Bearer %token%" --data "{\"reasonForAbort\":\"Jenkins task failed\"}" -k -X PUT %OrchestratorURL%/PRRestService/DeploymentManager/v1/deployments/%DeploymentID%/abort
    d. In the **Script** field, enter (for Linux) export token=`curl --insecure -d "client_id=${client_id}&client_secret=${client_secret}&grant_type=client_credentials" https://10.224.203.11:8443/prweb/PRRestService/oauth2/v1/token | jq -r .access_token` curl --insecure -H "Content-Type: application/json" -H "Accept: application/json" -H "Authorization:Bearer ${token}" --data "{\"reasonForAbort\":\"Jenkins task failed\"}" -k -X PUT ${OrchestratorURL}/PRRestService/DeploymentManager/v1/deployments/${DeploymentID}/abort
    e. Click **Save**.

Configuring Deployment Manager notifications

You can enable notifications to receive updates about the events that occur in your pipeline. For example, you can choose to receive emails about whether unit tests failed or succeeded. You can receive notifications in the Deployment Manager notifications gadget, through email, or both. By default, all notifications are enabled for users who are configured in Deployment Manager.

By default, all notifications are enabled for users who are configured in Deployment Manager.

- Viewing and updating email accounts for notifications

  Receiving email notifications requires that an email account is configured on the orchestration server. You can view and update your email settings in Deployment Manager.

- Creating custom Deployment Manager notification channels in 5.1.x

  You can extend Deployment Manager notification capabilities by creating custom notification channels. For example, you can send text messages to mobile devices when tasks start, stop, and are unsuccessful.

- Managing notifications

  Enable and receive notifications so that you can remain informed about important tasks in your pipeline. For example, you can receive emails when certain tasks fail.

Viewing and updating email accounts for notifications

Receiving email notifications requires that an email account is configured on the orchestration server. You can view and update your email settings in Deployment Manager.

Changing your email settings requires access to Dev Studio, so your user role must have permission to access Dev Studio. For more information, see Understanding roles and users.

1. In the navigation pane of Deployment Manager click **Settings Email configuration** .
2. To update your email settings, perform the following steps:
    a. At the top of the **Settings: Email configuration** page, click **Dev Studio**.
    b. In the **Edit Email Account** form, configure and save the email account that you want to use to receive notifications.
    c. In the bottom left corner of Dev Studio, click **Back to Deployment Manager** to return to the Deployment Manager portal.
    d. Click the **refresh** icon to refresh your email configuration.

Creating custom Deployment Manager notification channels in 5.1.x

You can extend Deployment Manager notification capabilities by creating custom notification channels. For example, you can send text messages to mobile devices when tasks start, stop, and are unsuccessful.

To create a custom notification channel, complete the following steps:

1. On the orchestration server, in Pega Platform, create a custom notification channel.
   For more information, see Adding a custom notification channel.
2. Add the application ruleset, which contains the channel that you created, to the Deployment Manager application.
   a. In the header of Dev Studio, click **Deployment Manager**, and then click **Definition**.
   b. On the Edit Application rule form, in the **Application rulesets** section, click **Add ruleset**.
   c. Press the Down arrow key and select the ruleset and version that contains the custom notification channel.
   d. Save the rule form.
3. Enable the channel that you created on the appropriate notifications by saving the notification in the application ruleset that contains the channel. **For example:** If you want to use the Mobile channel for the *pyStartDeployment* notification, save the *pyStartDeployment* notification in the application ruleset that contains the Mobile channel.
4. Enable the channel on the notification.
   a. Open the notification by clicking **Records Notifications** .
   b. Click the **Channels** tab.
   c. On the **Channel configurations** page, select the channel that you want to use.
   d. Save the rule form.

- Understanding custom Deployment Manager notification channels

  When notifications are enabled, you can receive notifications about the events that occur in your pipeline, such as when tasks start or stop. You can receive notifications through email, the Deployment Manager notifications gadget, or both. You can also create custom notification channels to meet application requirements such as sending notifications as phone text messages or as push notifications on mobile devices.

Understanding custom Deployment Manager notification channels

When notifications are enabled, you can receive notifications about the events that occur in your pipeline, such as when tasks start or stop. You can receive notifications through email, the Deployment Manager notifications gadget, or both. You can also create custom notification channels to meet application requirements such as sending notifications as phone text messages or as push notifications on mobile devices.

Deployment Manager provides the following notifications to which you can add channels:

- *pyAbortDeployment*
- *pyTaskFailure*
- *pyTaskFailure*
- *pyTaskCompletion*
- *pyStartDeployment*
- *pyStageCompletion*
- *pySchemaChange*
- *pyDeploymentCompletion*
- *pyAgedUpdateActionTaken*
- *pyAgedUpdateActionRequired*

Managing notifications

Enable and receive notifications so that you can remain informed about important tasks in your pipeline. For example, you can receive emails when certain tasks fail.

To enable notifications and select the notifications that you want to receive, do the following steps:

1. In the navigation pane of Deployment Manager click your profile icon.
2. Click **Notification preferences**.
3. Select the events for which you want to receive notifications.
4. Specify how you want to receive notifications.
5. Click **Submit**.

Dynamic system settings

Deployment Manager exposes a few Dynamic System Settings (DSS) to override the default behavior. See below for a list of configurations available to override the default behavior. All of these settings are optional and are not required by default. Carefully evaluate each configuration before configuring for your application.

Service connection timeout

All interactions within Deployment Manager happen through REST services, and the default time for these connections is

30 seconds. The DSS *DMConnectorTimeOut* is responsible for setting the timeout in milliseconds. The value is set to 30000 milliseconds (30 sec.) by default. This dynamic system setting value can be modified as required to affect the connection timeout when calling the Deployment Manager service. Note that this will cause longer interaction times for all connectors.

Owning ruleset	PegaDeploymentManagerIntegrations
Purpose	DMConnectorTimeOut
Value	30000

Application Mapping in External LDAP System

This DSS *deploymentmanager/security/external_apps_mapping/enabled* drives the source of the application mapping with the operator. If set to **true**, the applications available to a user is managed from the external authentication system (SSO or LDAP). If enabled. any changes made within Deployment Manager are overwritten with values from the authentication system on next operator log in.

Owning ruleset	PegaDevopsShared
Purpose	deploymentmanager/security/external_apps_mapping/enabled
Value	False

Use JWT Grant Type

Merge actions are performed from the application instance of the developer. Currently, Deployment Manager uses client credentials to integrate with the orchestrator, however if there is a requirement to use JWT grant type, then the following dynamic system setting rule can be created and set to TRUE. This will enable tracking the initiator of a merge requests. In addition to this setting, the appropriate authentication profile and JWT client/server profiles must be updated on the candidate and orchestrator environments.

Owning ruleset	PegaDevopsShared
Purpose	deploymentmanager/security/use_jwt_grant_type/enabled
Value	No value

Deployment History Maximum Size

The maximum count of deployments fetched for deployment history of a pipeline is currently set to 50. If you want to customize the default value, the following DSS can be edited and set to a new default value greater than 50. Note that the value set here will be applied for all pipelines and may potentially fetch large amounts of data if set to an arbitrarily large value. In addition to this you may have to increase the timeout setting to accommodate for the increased data, otherwise you may see blank deployment history.

Owning ruleset	PegaDeploymentManagerCore
Purpose	deploymentmanager/pipeline/maximum_deployments_to_show/size
Value	50

Creating and using custom repository types for Deployment Manager

In Deployment Manager 3.1.x and later, you can create custom repository types to store and move your artifacts. For example, you can create a Nexus repository and use it similarly to how you would use a Pega Platform-supported repository type such as file system. By creating custom repository types, you can extend the functionality of Deployment Manager through the use of a wider variety of repository types with your artifacts.

To create a custom repository type to use with Deployment Manager, complete the following steps:

1. Create a custom repository type. For more information, see Creating a custom repository type.
2. If you are using Deployment Manager 3.3.x or 4.1.x or later on each candidate system, add the ruleset that contains the custom repository type as a production ruleset to the PegaDevOpsFoundation:Administrators access group.
    a. In the header of either Designer Studio (if you are using Deployment Manager 3.3.x) or Dev Studio (if you are using Deployment Manager 4.1.x or later), click **Records Security Access Group** .
    b. Click **PegaDevOpsFoundation:Administrators**.

c. Click **Advanced**
d. In the **Run time configuration** section, click the **Production Rulesets** field, press the Down arrow key, and select the ruleset that contains the custom repository type.
e. Save the rule form.

3. Import the ruleset on which the custom repository is configured in to the orchestration system and add the ruleset to the PegaDeploymentManager application stack.
   a. On the orchestration system, import the ruleset by using the Import wizard. For more information, see Importing rules and data by using the Import wizard.
   b. In either the Designer Studio or Dev Studio header, in the **Application** field, click **PegaDeploymentManager**, and then click **Definition**.
   c. On the **Edit Application** rule form, in the **Application rulesets** field, click **Add ruleset**.
   d. Click the field that is displayed, press the Down arrow key, and then select the ruleset that contains the custom repository type.
   e. Save the rule form.

- Installing and enabling Sonatype Nexus Repository component for Sonatype Nexus Repository Manager 3

  To create a connection between Pega Platform or Deployment Manager and Nexus Repository Manager 3, use the Sonatype Nexus Repository component. Use this repository for centralized storage, versioning, and metadata support for your application artifacts.

- Installing and enabling Sonatype Nexus Repository component for Sonatype Nexus Repository Manager 2

  Create a connection between Pega Platform or Deployment Manager and Sonatype Nexus Repository Manager 2 with the Sonatype Nexus Repository component. Use this repository for centralized storage, versioning, and metadata support for your application artifacts.

Installing and enabling Sonatype Nexus Repository component for Sonatype Nexus Repository Manager 3

To create a connection between Pega Platform or Deployment Manager and Nexus Repository Manager 3, use the Sonatype Nexus Repository component. Use this repository for centralized storage, versioning, and metadata support for your application artifacts.

The component for Sonatype Nexus Repository Manager 3 supports Pega 8.1, 8.2, 8.3, and 8.4.

Note: Because of potential conflicts, you should not use both Sonatype Nexus Repository Manager 2 and Sonatype Nexus Repository Manager 3 type repositories in one application. If you want to use both repository types, contact NexusComponentSupport@pega.com.

For answers to frequently asked questions, see the Nexus FAQ page.

For questions or issues, send an email to NexusComponentSupport@pega.com.

- Downloading and enabling the component

  Download and enable the component so that you can configure a Sonatype Nexus Repository Manager 2 repository.

- Creating a Sonatype Nexus Repository Manager 3 repository

  After downloading and enabling the Sonatype Nexus Repository Manager 3 component, create a repository in Pega Platform.

- Understanding API usage

  When you use repository APIs to interact with Nexus Repository Manager 3, note the following information:

Downloading and enabling the component

Download and enable the component so that you can configure a Nexus Repository Manager 3 repository.

To download and enable the component, do the following steps:

1. Download the component from Pega Marketplace.
2. In the header of Dev Studio, click the name of your application, and then click **Definition**.
3. In the **Application** rule form, on the **Definition** tab, in the **Enabled components** section, click **Manage components**.
4. Click **Install new**, select the file that you downloaded from Pega Marketplace, and then click **Open**.
5. Select the **Enabled** check box to enable this component for your application, and then click **OK**.
6. In the list of enabled components, select **PegaNexus3Repository**, select the appropriate version, and then click **Save**.
7. If you are using Deployment Manager, on each candidate system and on the orchestration system, perform one of the following tasks:
   - Download and enable the component by repeating steps 1 - 6.

- Add the *PegaNexus3:01-01* and *PegaNexusCommon:01-01* rulesets as production rulesets to the *PegaDevOpsFoundation:Administrators* access group.

- Creating a Sonatype Nexus Repository Manager 3 repository

Creating a Sonatype Nexus Repository Manager 3 repository

After downloading and enabling the Sonatype Nexus Repository Manager 3 component, create a repository in Pega Platform.

Note: You can create only raw type repositories.To create a repository, do the following steps:

1. In the header of Dev Studio, click **Create SysAdmin Repository** .
2. In the **Create Repository** rule form, enter a description and name for your repository, and then click **Create and open**.
3. In the **Edit Repository** rule form, on the **Definition** tab, click **Select**.
4. In the **Select repository type** dialog box, click **Nexus 3**.
5. In the **Repository configuration** section, configure location information for the repository:
   a. In the **System URL** field, enter the URL of your Nexus Repository Manager 3 server.
   b. In the **Repository name** field, enter the name of the repository.
   c. In the **Root path** field, enter the path of the folder where repository assets are stored. Do not include the repository folder in the path, and do not start or end the path with the slash (/) character. **For example:** To store assets in a folder with the URL http://myexusrepo.com/repository/raw/myCo/devops, enter the following information:
      - **System URL**: http://mynexusrepo.com
      - **Repository name**: raw
      - **Root path**: myCo/devops

      The connector will allow you to browse the assets in this folder from inside Pega Platform.
6. In the **Authentication** section, configure authentication information:
   a. In the **Authentication profile** field, enter the name of a new authentication profile, and then click the **Open** icon to configure the profile.

      The authentication profile stores the credentials that Pega Platform needs to authenticate with the Nexus Repository Manager 3 API.

   b. In the **Create Authentication Profile** rule form, in the **Type** list, select **Basic**.
      Only Basic authentication is supported. For more information about Basic authentication profiles, see Configuring a Basic authentication profile.
   c. Enter a name and description for the authentication profile.
   d. Click **Create and open**.
7. In the **Edit Authentication Profile** rule form, configure authentication information:
   a. Enter the user name, password, realm, and host name required to authenticate with Sonatype Nexus Repository Manager 3. For more information, see the Sonatype Nexus Repository Manager 3 documentation.
   b. Select the **Preemptive authentication** check box.
   c. Click **Save**.
8. To verify that the system URL, authentication profile, and repository name are configured properly, in the **Edit Repository** rule form, on the **Definition** tab, click **Test connectivity**.

   If there are any errors, ensure that the credentials in the authentication profile are correct and that Pega Platform can access the system URL that you entered.

   Note: Testing connectivity does not verify that the root path is configured properly.
9. Click **Save**.

- Downloading and enabling the component

Understanding API usage

When you use repository APIs to interact with Nexus Repository Manager 3, note the following information:

- Sonatype Nexus Repository Manager 3 does not support the create folder API (*D_pxNewFolder*), because the repository cannot have empty folders.
- The create file API (*D_pxNewFile*) and get file API (*D_pxGetFile*) support only Basic Authentication and support a file size of up to 5 GB.
- The delete API (*D_pxDelete*) does not work on folders, only files. If all the files in a folder are deleted, the folder is also deleted.

- Creating a Sonatype Nexus Repository Manager 3 repository

Installing and enabling Sonatype Nexus Repository component for Sonatype Nexus Repository Manager 2

Create a connection between Pega Platform or Deployment Manager and Sonatype Nexus Repository Manager 2 with the

Sonatype Nexus Repository component. Use this repository for centralized storage, versioning, and metadata support for your application artifacts.

For answers to frequently asked questions, see the Nexus FAQ page.

For questions or issues, send an email to NexusComponentSupport@pega.com.

- Downloading and enabling the component

  Download and enable the component so that you can configure a Nexus Repository Manager 3 repository.

- Creating a Sonatype Nexus Repository Manager 2 repository

  After downloading and enabling the component for Sonatype Nexus Repository Manager 2, create a repository in Pega Platform.

- Understanding repository API usage

  When you use repository APIs to interact with Nexus Repository Manager 2, note the following information:

Downloading and enabling the component

Download and enable the component so that you can configure a Sonatype Nexus Repository Manager 2 repository.

To download and enable the component, do the following steps:

1. Download the component from Pega Marketplace.
2. In the header of Dev Studio, click the name of your application, and then click **Definition**.
3. In the **Application** rule form, on the **Definition** tab, in the **Enabled components** section, click **Manage components**.
4. Click **Install new**, select the file that you downloaded from Pega Marketplace, and then click **Open**.
5. Select the **Enabled** check box to enable this component for your application, and then click **OK**.
6. In the list of enabled components, select **Pega Nexus Repository Connector**, select the appropriate version, and then click **Save**.
7. If you are using Deployment Manager, on each candidate system and on the orchestration system, perform one of the following tasks:
   - Download and enable the component by repeating steps 1 - 6.
   - Add the *PegaNexus:01-01* and *PegaNexusCommon:01-01* rulesets as production rulesets to the *PegaDevOpsFoundation:Administrators* access group.

- Creating a Sonatype Nexus Repository Manager 2 repository

Creating a Sonatype Nexus Repository Manager 2 repository

After downloading and enabling the component for Sonatype Nexus Repository Manager 2, create a repository in Pega Platform.

To create a repository, do the following steps:

1. In the header of Dev Studio, click **Create SysAdmin Repository** .
2. In the **Create Repository** rule form, enter a description and name for your repository, and then click **Create and open**.
3. In the **Edit Repository** rule form, on the **Definition** tab, click **Select**.
4. In the **Select repository type** dialog box, click **Nexus 2**.
5. In the **Repository configuration** section, configure location information for the repository:
   a. In the **System URL** field, enter the URL of your repository.
   b. In the **Repository ID** field, enter the ID of the repository, which you can find on the **Configuration** tab in Nexus Repository Manager 2.
   For more information, see the documentation for Nexus Repository Manager 2.
   c. In the **Root path** field, enter the path of the folder where repository assets are stored. Do not include the repository folder in the path, and do not start or end the path with the slash (/) character. **For example:** To store assets in a folder with the URL http://myexusrepo.com/repository/raw/myCo/devops, enter the following information:
      - **System URL**: http://mynexusrepo.com
      - **Repository ID**: raw
      - **Root path**: myCo/devops

   The connector will allow you to browse the assets in this folder from inside Pega Platform.
6. In the **Authentication** section, configure authentication information:
   a. In the **Authentication profile** field, enter the name of a new authentication profile and click the **Open** icon to configure the profile.

      The authentication profile stores the credentials that Pega Platform needs to authenticate with the Nexus Repository Manager 2 API.

   b. In the Create Authentication Profile rule form, in the **Type** list, select **Basic**.

Only Basic authentication is supported.
   c. Enter a name and description for your authentication profile.
   d. Click **Create and open**.
7. In the **Edit Authentication Profile** rule form, configure authentication information:
   a. Enter the user name, password, realm, and host name required to authenticate with Nexus Repository Manager 2. For more information, see the Nexus Repository Manager 2 documentation.
   b. Select the **Preemptive authentication** check box.
   c. Click **Save**.
8. To verify that the system URL and authentication profile are configured properly, in the **Edit Repository** rule form, on the **Definition** tab, click **Test connectivity**.

   If there are any errors, ensure that the credentials in the authentication profile are correct and that Pega Platform can access the system URL that you entered.

   Note: Testing connectivity does not verify that the repository ID or root path are configured properly.
9. Click **Save**.

- Downloading and enabling the component

Understanding repository API usage

When you use repository APIs to interact with Nexus Repository Manager 2, note the following information:

- Sonatype Nexus Repository Manager 2 does not consider recursiveDelete parameter for the delete API (*D_pxDelete*). All folder deletes are considered recursive.
- The create file API (*D_pxNewFile*) and get file API (*D_pxGetFile*) support only Basic Authentication and support a file size of up to 5 GB.

- Creating a Sonatype Nexus Repository Manager 2 repository

Managing test cases separately in Deployment Manager

Use Deployment Manager to package and deploy test cases separately on the candidate systems in the pipeline. When you configure a pipeline in Deployment Manager, you specify the details of the test package that you want to deploy.

Configure the Deploy task to define the stage to deploy your test artifact. Select the **Deploy test artifact?** on the Deploy task of the stage where you are deploying the test artifact.

To use a separate test package, you must create a test application layer on the development systems in your pipeline.

- Configuring the application stack on the development or source development system

  You must first configure your application stack on either the development or source development system.

- Configuring the application stack on the remote development system in a distributed, branch-based environment

  If you are using a distributed, branch-based environment, you must configure the application stack on the remote development system.

- Configuring pipelines to use test cases

  When you add or modify a pipeline, you specify whether you want to deploy test cases and then configure details for the test application, including its name and access group to which it belongs, in the Application test cases section.

Configuring the application stack on the development or source development system

You must first configure your application stack on either the development or source development system.

Configure the application stack according to one of the following scenarios:

- If you are using a distributed, branch-based environment, complete the following steps on the remote development system.
- If you are using a branch-based environment, complete the following steps on the development system.
- If you are not using branches, complete the following steps on the development system.

  Configure the application stack by performing the following steps:

1. Create the target application.
2. Create a test application, which contains the test rulesets that you want to separately deploy, that is built on the target application.
3. Create a development application that is built on top of the test application, which developers can log in to so that they can create and work in branches.

4. Lock both the target and test applications

Configuring the application stack on the remote development system in a distributed, branch-based environment

If you are using a distributed, branch-based environment, you must configure the application stack on the remote development system.

Complete the following steps:

1. Create the target application.
2. Create a test application, which contains the test rulesets that you want to separately deploy, that is built on the target application.
3. Lock both the target and test applications.
4. Lock both the target and test application rulesets.

Configuring pipelines to use test cases

When you add or modify a pipeline, you specify whether you want to deploy test cases and then configure details for the test application, including its name and access group to which it belongs, in the **Application test cases** section.

Note: To select a stage where you want to deploy the artifact, select **Deploy test artifact?** on the Deploy task for that stage. For more information about using Deployment Manager, see Configuring an application pipeline.

When you use separate product rules for test cases and run a pipeline, the Run Pega unit tests, Enable test coverage, and Validate test coverage tasks are run for the access group that is specified in the Application test cases section.

You must also perform the following steps on the candidate system on which you are running tests:

1. Log in to the test application.
2. In the header of Dev Studio, click **Configure Application Quality Settings** .
3. Select the **Include built-on applications** radio button, and then click **Save**.

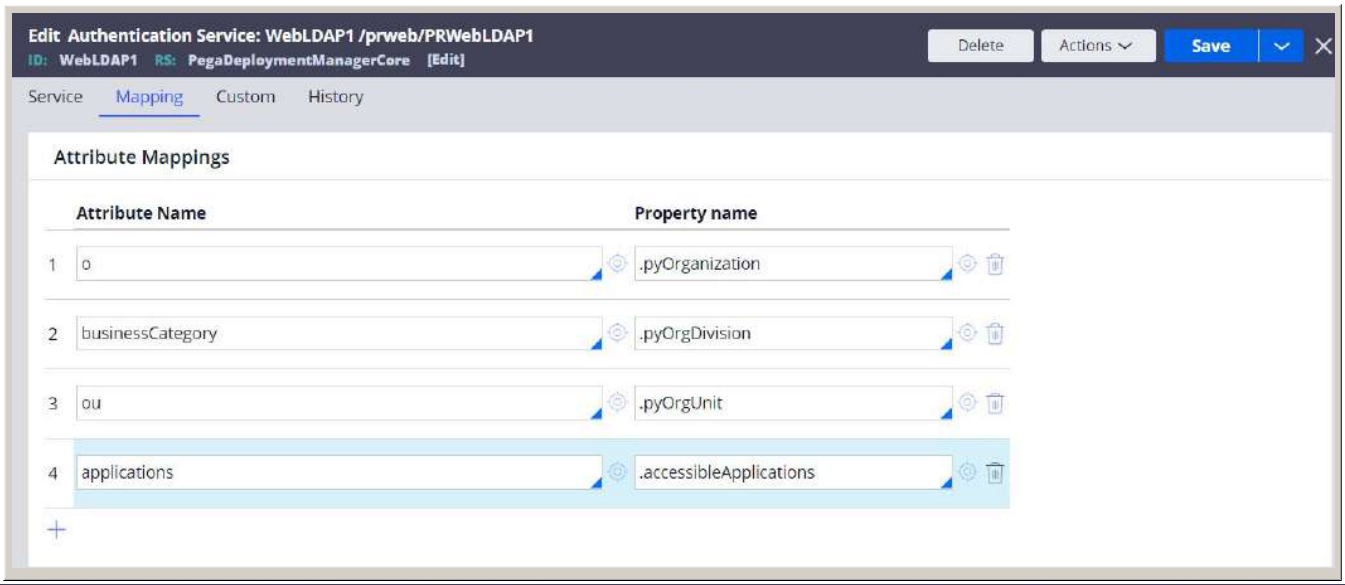
Configuring SSO and LDAP

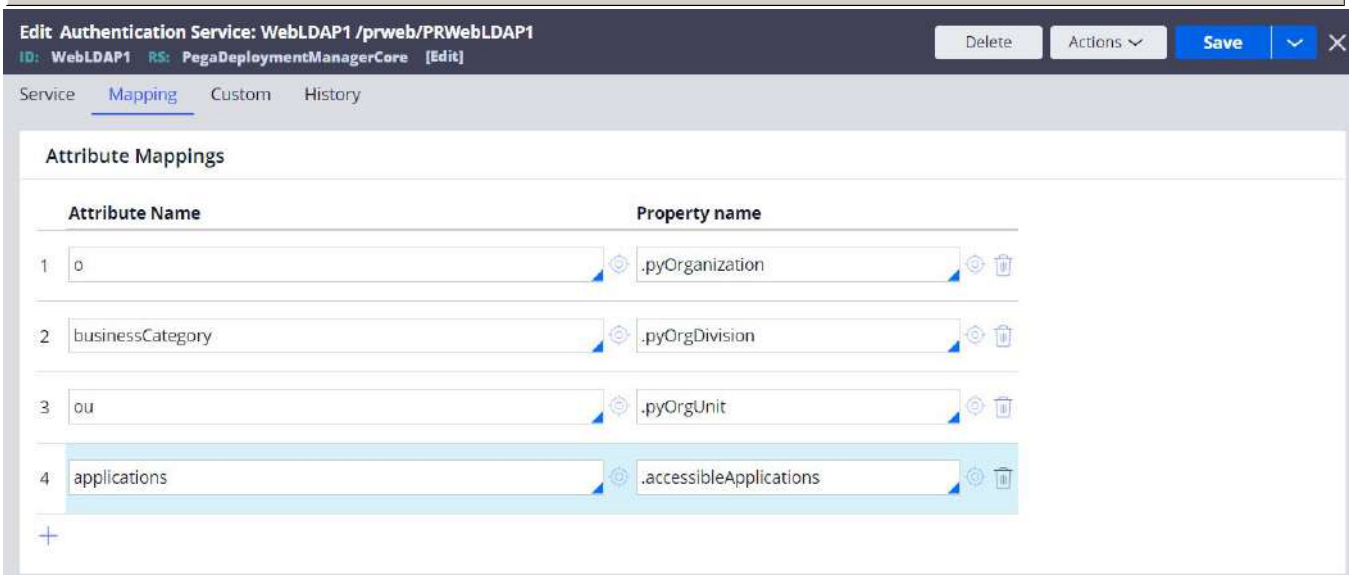
Deployment Manager supports LDAP and Single Sign-On (SSO) authentication, and assigns users roles based on the configuration settings in the active directory and SSO.

When a role other than **SuperAdmin** is specified for an operator, an application must be provided. Applications needed for the operator must be mapped to the *accessibleApplications* property when configuring LDAP and SSO. This property extends application accesses for each operator without SuperAdmin privileges. To assign the attribute to an operator, you must create the *accessibleApplications* property and provide the unauthenticated user access to the property.

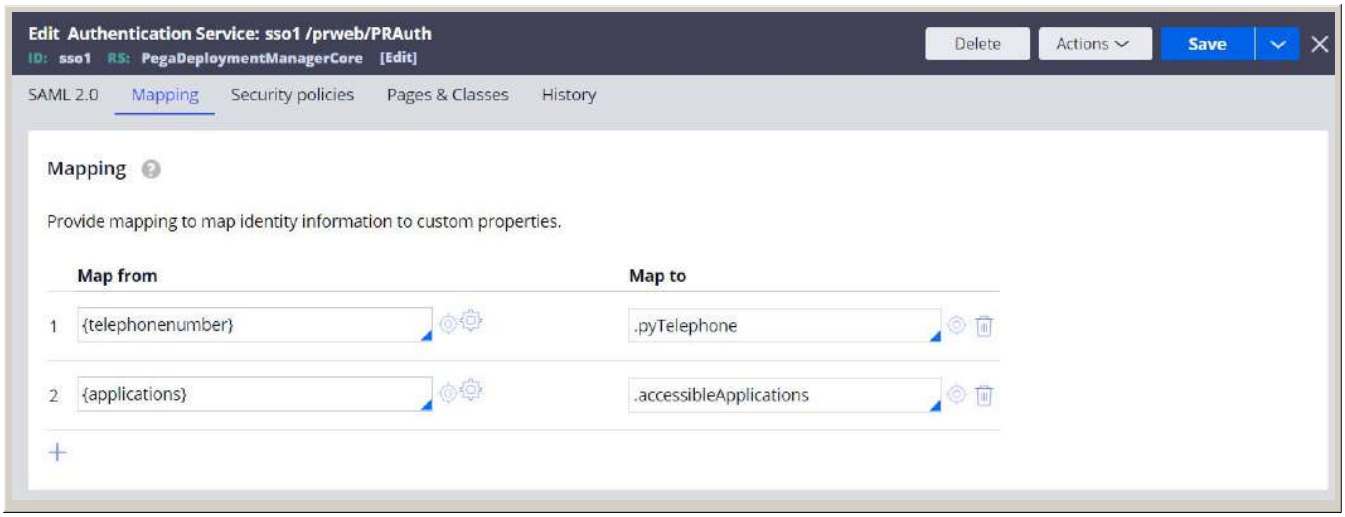
1. Create a new ruleset and create the property *accessibleApplications*. Set the class as **Data-Admin-Operator-ID**.
2. Create a new application and add the ruleset above in the application stack.
3. Create a new Access Group and provide access to the new application that you create in step 2.
4. Add the Access Group to the **Access Group Name** field in the Browser Requestor Type screen.
   Configure the LDAP or SSO authentication service to map attributes to the *accessibleApplications* property, as shown in the following figures:

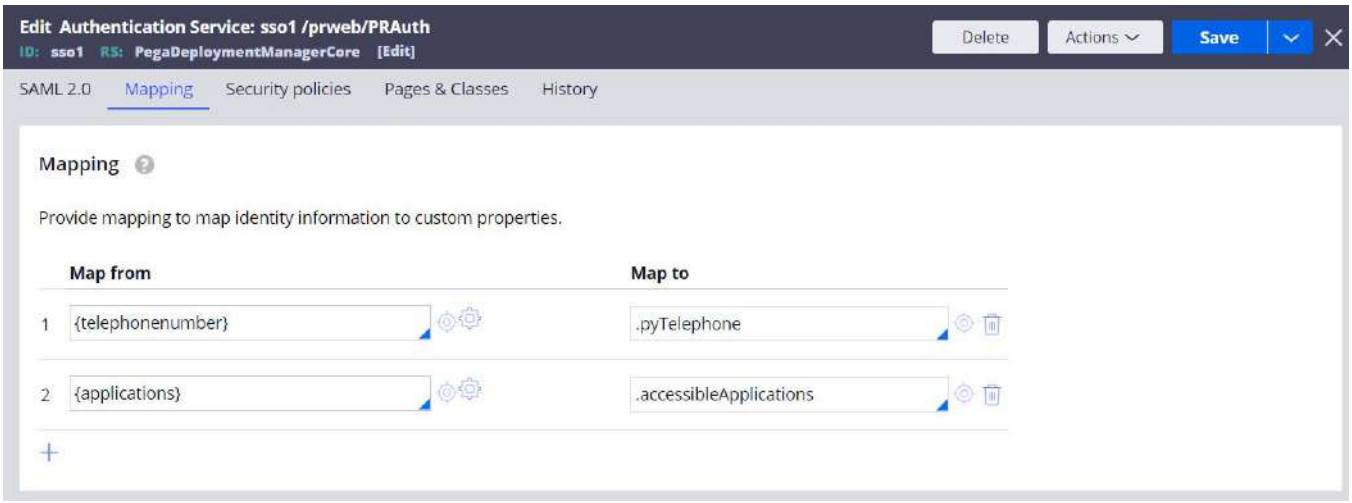
   LDAP mapping


Edit Authentication Service: WebLDAP1 /prweb/PRWebLDAP1
ID: WebLDAP1 RS: PegaDeploymentManagerCore [Edit]
Delete
Actions
Save
Service
Mapping
Custom
History
Attribute Mappings
Attribute Name
Property name
1 o .pyOrganization
2 businessCategory .pyOrgDivision
3 ou .pyOrgUnit
4 applications .accessibleApplications


Edit Authentication Service: WebLDAP1 /prweb/PRWebLDAP1
ID: WebLDAP1 RS: PegaDeploymentManagerCore [Edit]
Delete
Actions
Save
Service
Mapping
Custom
History
Attribute Mappings
Attribute Name
Property name
1 o .pyOrganization
2 businessCategory .pyOrgDivision
3 ou .pyOrgUnit
4 applications .accessibleApplications

SSO mapping


Edit Authentication Service: sso1 /prweb/PRAuth
ID: sso1 RS: PegaDeploymentManagerCore [Edit]
Delete
Actions
Save
SAML 2.0
Mapping
Security policies
Pages & Classes
History
Mapping
Provide mapping to map identity information to custom properties.
Map from
Map to
1 {telephonenumber} .pyTelephone
2 {applications} .accessibleApplications



5. To map applications externally, set dynamic system setting *PegaDevopsShared • deploymentmanager/security/external_apps_mapping/enabled* to True. If this is set to False, applications updated for operators in Deployment Manager will take precedence, and applications mentioned in the LDAP/SSO directory will not be applicable.

Accessing API documentation

Deployment manager provides REST APIs for interacting with resources that in the Deployment Manager interface. Use these APIs to create and manage pipelines by using automated scripts or external information.

To access API documentation, open the Documentation/readme-for-swagger.md file in the artifact file that you downloaded from marketplace.

Release notes

These release notes provide information about enhancements, known issues, issues related to updating from a previous release, and issues that were resolved in each release of Deployment Manager.

For answers to frequently asked questions, see the Deployment Manager FAQ page.

- Deployment Manager 5.2.1

  Deployment Manager 5.2.1 includes the following new feature and resolved issues.

- Deployment Manager 5.1.1

  Deployment Manager 5.1.1 includes the following new features and enhancements.

- Deployment Manager 4.8.4

  Deployment Manager 4.8.4 includes the following resolved issues.

- Deployment Manager 4.8.3

  Deployment Manager 4.8.3 includes the following resolved issues.

- Deployment Manager 4.8.2

  Deployment Manager 4.8.2 includes the following new feature and resolved issues.

- Deployment Manager 4.8.1

  Deployment Manager 4.8.1 includes the following enhancements and resolved issues:

- Deployment Manager 4.7.1

  Deployment Manager 4.7.1includes the following enhancements.

- Deployment Manager 4.6.1

  Deployment Manager 4.6.1 includes the following enhancements and resolved issues.

- Deployment Manager 4.5.1

  Deployment Manager 4.5.1 includes the following enhancements and resolved issues.

- Deployment Manager 4.4.2

  Deployment Manager 4.4.2 includes the following resolved issues.

- Deployment Manager 4.4.1

  Deployment Manager 4.4.1 includes the following enhancements and known issues.

- Deployment Manager 4.3.2

  Deployment Manager 4.3.2 includes the following resolved issues.

- Deployment Manager 4.3.1

  Deployment Manager 4.3.1 includes the following enhancements.

- Deployment Manager 4.2.1

  Deployment Manager 4.2.1 includes the following enhancements.

- Deployment Manager 4.1.1

  Deployment Manager 4.1.1 includes the following enhancements.

- Deployment Manager 3.4.1

  Deployment Manager 3.4.1 includes the following enhancements.

- Deployment Manager 3.3.1

  Deployment Manager 3.3.1 includes the following enhancements and known issues.

- Deployment Manager 3.2.1

  Deployment Manager 3.2.1 includes the following enhancements.

- Deployment Manager 3.1.1

  Deployment Manager 3.1.1 includes the following enhancements.

- Deployment Manager 2.1.4

  Deployment Manager 2.1.4 includes the resolved issues.

- Deployment Manager 2.1.3

  Deployment Manager 2.1.4 includes the following enhancements.

- Deployment Manager 2.1.2

  Deployment Manager 2.1.2 includes the known issues.

- Deployment Manager 1.1.3

  Deployment Manager 1.1.3 includes the following enhancements.

- Deployment Manager 1.1.2

  Deployment Manager 1.1.2 includes the following known and resolved issues.

Deployment Manager 5.2.1

Deployment Manager 5.2.1 includes the following new feature and resolved issues.

Enhancements

The following new features are available in this release:

- You can now publish application changes that you make in App Studio to the pipeline. Publishing your changes creates a patch version of the application and starts a deployment. For example, you can change a life cycle, data model, or user interface elements in a screen and submit those changes to systems in the pipeline. For more information, see Publishing application changes in App Studio.

Dependencies and issues

Deployment Manager 5.2.1 is released with the following dependencies and known issues:

- Pega Platform version 8.5.2 has a known issue with Nexus 2 and Nexus 3 repositories, resulting in an error only when running diagnostics. This issue has no impact on the run-time execution of a pipeline with these repositories.
- If the default email account in the orchestrator is not configured, or if the email ID is not configured for the operator, an exception stack trace is printed in the logs. To avoid this issue, turn off the logging level for *com.pega.platform.integrationengine.client.email.PegaEmailClient* and *com.pegarules.generated.pega_integrationengine_default*.
- Deployment Manager will always consume the latest application version, unless otherwise specified.
- Applications being managed by a business change pipeline cannot automatically increment a minor version higher than v99. You must manually roll over your versioning should your application or ruleset versions encounter this scenario.
- If you are upgrading from Deployment Manager 5.1 to 5.2, business change pipelines must be recreated due to an internal property configuration.

Resolved issues

The following bug fixes have been implemented in Deployment Manager 5.2.1:

- Diagnostics now properly identify the communication between candidate environment and orchestrator. This no longer impacts the execution of data migration pipelines.
- When an active deployment ends in a rollback failed state, a new deployment will queue as intended instead of moving to an in-progress state.
- The deployment success frequency metric has been fixed to show the accurate deployment record count.
- Complete artifact history has been restored when merging multiple branches and generating a new artifact.

Deployment Manager 5.1.1

Deployment Manager 5.1.1 includes the following new features and enhancements.

New features

The following new features are available in this release:

- Flexible pipeline configurations now allow for customization of your pipeline model. Pipelines now support more than four stages, along with the ability to rename stages and add or remove tasks as required.
- **Multispeed deployments** provide the ability to queue multiple deployments in the same stage of a pipeline. With this functionality, developers can regularly merge their changes for to selective testing and promotion of deployments without blocking the pipeline. This approach introduces the concept of transition between stages that require user input to proceed.
- Pipeline templates have been added to define recommended release processes. Deployment Manager supports the following application pipeline templates:
  - Merge
  - Deployment
  - Business Change

  For more information about pipeline templates, see Pipeline templates.
- Business change pipelines support everyday, business-as-usual changes to your application. With this new pipeline type, users can respond to changing requirements by modifying and deploying application rules in a controlled manner. For more information, see Managing the business-as-usual changes.**Note:** The business change pipeline feature in version Deployment Manager 5.1 is supported with customers on Pega Platform 8.5.2 or later. Customers are not recommended to upgrade to Deployment Manager 4.8.4, as the suggested upgrade path from Deployment Manager 4.8.3 is to 5.1. Candidate environments must be on Pega Platform 8.5.2 to use the business change pipeline feature.
- **Environment templates** are now utilized when creating or editing a pipeline. This feature automatically populates pipeline stages with associated tasks and determines what validations to apply. If you do not want to use an environment template, you can select a custom template to use all available tasks.
- **Change sets** are a new resource type used to track rule changes that are incorporated into a branch, which is further passed on to a deployment. With this functionality, users can trace the source of a deployment and associated work items.
- OAuth 2.0 is now used as the standard way to authenticate interactions. Deployment Manager supports candidates on basic authentication, but OAuth 2.0 authentication is recommended for secure two-way communication in candidate systems.
- Single Sign-On (SSO) is now supported. You can now follow the guided approach to configure SSO and an LDAP system integration. See Configuring SSO and LDAP for more information.
- As of Deployment Manager 5.1 and later, the existing *cicd/v1* REST APIs have been deprecated and replaced with additional API support. For more information about the update, see the Swagger documentation.

Enhancements

The following enhancements have been implemented to existing Deployment Manager functionality:

- A new environments tab has been added to the pipeline creation screen, which stores the environment URL and authentication profile information. This tab is only visible during pipeline creation. You can modify this information

after creating the pipeline by clicking directly on the appropriate stage.
- The task selection and configuration experience has been updated. When you add a task to a stage, a right-pane overlay is displayed so that you can enter required information.
- A single pipeline no longer supports both pre-merge criteria and deployment stages. As of Deployment Manager 5.1 and later, this criteria is represented by two separate pipelines, a merge pipeline and a deployment pipeline.
- A complete list of available tasks in Deployment Manager is now available in the task catalog. For more information, see Task catalog.

Dependencies and issues

Deployment Manager 5.1.1 is released with the following dependencies and known issues:

- Deployment Manager orchestrator system is supported only on Pega Platform version 8.5.2 or later.
- Pega Platform version 8.5.2 has a known issue with Nexus 2 and Nexus 3 repositories that results in an error only when you run diagnostics. That issue has no impact on the run-time execution of a pipeline with these repositories.
- Deployment Manager version 5.1.1 does not support custom tasks or integration with App Studio publishing. Later versions of Deployment Manager will re-introduce these features.
- If the default email account in the orchestrator is not configured or if the email ID is not configured for the operator, an exception stack trace is printed in the logs. To avoid this issue, turn off the logging level for *com.pega.platform.integrationengine.client.email.PegaEmailClient* and *com.pegarules.generated.pega_integrationengine_default*.
- Deployment Manager version 5.1.1 does not support the option to skip a failed task.
- Jenkins integrations now require OAuth 2.0 to be enabled.
- Diagnostics fail to identify the ability of the candidate environment to access the orchestrator. This currently impacts the execution of data migration pipelines. Ensure that your candidate can access the orchestrator by following the steps outlined in Setting up candidate environments.
- Deployment Manager 5.1 does not currently support the ability to trigger a deployment using an existing artifact.

Deployment Manager 4.8.4

Deployment Manager 4.8.4 includes the following resolved issues.

Enhancements

The following new features are available in this release:

- Deployment Manager now enables users on Pega Platform 8.1.x through 8.5.1 to use the enhancements introduced in Deployment Manager 5.1.**Note:** Revision management users of Deployment Manager are not recommended to promote to Deployment Manager 4.8.4. To use the business change pipeline feature in Deployment Manager 5.1, candidate environments must be on Pega Platform 8.5.2 or later. Refer to Deployment Manager 5.1 documentation if the candidate is managed by 5.1 systems.
- If candidate systems are between Pega Platform 8.1 and Pega Platform 8.5.1, the candidate must have 4.8.4 Pega DevOps Foundation. If candidates are managed by an orchestrator on version 5 or later, you must create the PegaDevopsShared configuration set it to true. If not set, the candidate will fall back to using the older 4.x APIs for interactions with Deployment Manager and the pipelines will not be functional if using a 5.x orchestrator. Creating this configuration and setting it to true ensures the candidates leverage the 5.x API service.
  - **Ruleset**: PegaDevopsShared
  - **Rule**: PegaDevopsShared.deploymentmanager/orchestrator/managed_by_5x/enabled
  - **Value**: True

Deployment Manager 4.8.3

Deployment Manager 4.8.3 includes the following resolved issues.

Resolved issues

The following issues were resolved in this release:

- Resolved Deployment Manager crash issues.

  Work object data being stored after the execution of an activity caused Deployment Manager to run out of memory, resulting in limited responsiveness. This issue has been resolved and up-time has been restored.

- Merges being rejected due to draft flows being contained in the branch.

  The merge activity now ignores draft flows if they are included in the branch being imported.

Deployment Manager 4.8.2

Deployment Manager 4.8.2 includes the following new feature and resolved issues.

Enhancements

The following new features are available in this release:

- You can now independently perform `package` and `deploy` actions with ad hoc tasks, outside of the context of a deployment.
- The connection timeout of rest connectors can now be customized as a user-input value (in milliseconds) through a new dynamic system setting, `DMConnectorTimeOut`. The value specified should be greater than 30000 (30 seconds), otherwise, the default value specified in the rest connector will take precedence.

Resolved issues

The following issues were resolved in this release:

- Unable to create new roles for users

  The add role functionality failed if a description for the role was not provided. This dependency has been removed, and roles can now be created with or without descriptions.

- External links to documentation were not working

  The hyperlinks to Pega documentation, including user guides and release notes, were not updated properly for the 4.8 release. This issue has been resolved and all documentation is now accessible from within Deployment Manager.

- Users without permissions to manage aged updates were not prompted with an error or manual intervention request during deployment.

  If a rule or data instance in an application package is older than an instance on a system to which you want to deploy, this is considered an aged update. If this is encountered by a user without permissions to manage aged updates, the user will be notified during the deployment process.

- If a draft flow was withdrawn or marked as "not draft" in a previous ruleset version, this caused an error on all subsequent deployments using a higher version of that ruleset.

  Draft flows have the ability to cause unintended failures if deployed to a production environment, and any ruleset containing draft flows (even if archived) would result in this error. Draft flows now marked as not available, blocked, or withdrawn are ignored during deployment.

- Changing the assigned resource of a task would overwrite the task name in the Pipeline model UI

  Task resources, names, and descriptions can now be edited as intended directly in the pipeline model.

- People landing page only showed 50 users

  Pagination restricting the amount of shown records has been disabled, and all users can now be viewed.

- Unable to create a pipeline if a previous version had failed

  Resubmitting a modified, resolved version of a previously failed pipeline would result in a stage duplication error due to a caching issue with the case life cycle properties. This has been resolved and pipelines that fail can be modified and recreated as necessary.

- No indication of URL redirects in diagnostics or logs

  A new category has been added to the Verifying intersystem connections page, which identifies URL redirections when connecting to development environments.

- Deployment Manager configured with Jenkins resulted in a JSON parsing error when trying to run a job

  Invalid data field values were provided due to an incorrect data transform script. The activity has been modified to resolve this issue.

- Logs provided incorrect user information regarding publishing and approving artifacts

  All log actions throughout the deployment process were previously associated to the operator who started the deployment. Now, all actions are properly associated to the users who performed them.

- Application version field only allowed numeric values

  Applications can now be versioned alphanumerically.

- Schema changes did not properly update in the target environment.

  Deploying an artifact with schema changes now properly applies the updates in the target environment as requested.

Deployment Manager 4.8.1

Deployment Manager 4.8.1 includes the following enhancements and resolved issues:

Enhancements

The following enhancements are available in this release:

- Assess application security by using the enhanced Pega Security Checklist task

  With the Pega Security Checklist, you can easily secure your applications and systems in one convenient area of Pega Platform™. The Security Checklist:
  - Performs a detailed assessment of your current security configuration to determine whether the settings follow best practices for application development.
  - Provides a status on each task in the Security Checklist page and blocks your application deployment if any task fails.
  - Stores an audit trail of the security configuration analysis and status at the time of deployment.

- Deploy revision packages by using Deployment Manager

  With Deployment Manager 4.8, you can now use Deployment Manager pipelines for deploying application revisions. If you choose not to migrate your deployments to a DevOps pipeline, Revision Manager remains backward compatible.

- Configure authentication profiles in Deployment Manager

  Deployment Manager service administrators can now create or edit authentication profiles in the Deployment Manager UI instead of accessing profiles in Dev Studio.

- Application-level rollback to a restore point

  Application-level rollbacks now provide a more granular approach to restore points, which you can use to revert rules and data instances in a specific application. This feature requires Pega Platform 8.4 and later.

- Trigger deployments by using existing deployment artifacts

  You can now trigger deployments by using production-ready artifacts from a previous production deployment. Two new fields, **Pipeline** and **Deployment**, replace the existing fields, **Select a repository** and **Select an artifact,** to provide a more efficient deployment approach and eliminate repository interactions from Orchestrator.

  You cannot use development artifacts to trigger a deployment.

- Changes to pipeline dependencies configuration

  The Deployment Manager UI for updating application dependencies is now read-only for pipelines that are created in versions 4.7 and earlier. This functionality supports client migrations to the new deployment-centric method of dependency configuration.

- Add Jenkins tasks to any pipeline phase

  You can now add Jenkins tasks to the Continuous Integration (CI) phase of a deployment pipeline. For a list of available parameters for a Jenkins task in CI, see the **Configuring Jenkins** documentation.

- Block draft flows in systems with production level 5

  Deployment manager now blocks deployments in systems with a production level of 5 if the artifact contains draft flows. If the production level is lower than 5, a warning message is displayed in the *Deployment History and Reports* section, which indicates that draft flows might cause production failures.
  - Any draft flows with availability set as withdrawn, not available, or blocked are ignored and the deployment will proceed. A complete list of draft flows encounter during import is shown in the deployment logs.
  - For dependencies and test artifact archives with draft flows, imports are processed normally regardless of system production level.

Resolved issues in Deployment Manager 4.8.1

In this release, minor security issues are resolved.

Deployment Manager 4.7.1

Deployment Manager 4.7.1includes the following enhancements.

Enhancements

The following enhancements are available in this release:

- Stop all ongoing deployments for a pipeline at once.

  You can now stop all the ongoing deployments for a pipeline at once. Stop all deployments to quickly troubleshoot issues and resolve failed pipelines.

- Use a chatbot to obtain information about common issues.

  You can now use a self-service chatbot to obtain troubleshooting tips and more information about common Deployment Manager issues. When you search for information, the chatbot provides you with answers and links to more information.

- Troubleshoot pipelines with enhanced diagnostics.

  Deployment Manager now provides enhanced diagnostics so that you can troubleshoot more issues. You receive warnings if you are using the defaultstore repository or Pega type repository in any environment.

- Perform new tasks with usability enhancements.

  Usability enhancements in Deployment Manager 4.7.1 now include:

  - Start another pipeline by using the Trigger deployment task in an active pipeline, which allows you to add pipeline stages.
  - Stop a deployment if a Jenkins task in the pipeline fails.
  - Archive inactive pipelines. By default, archived pipelines do not appear in the Deployment Manager interface.
  - Temporarily disable pipelines that frequently fail to prevent additional deployments on the pipeline.
  - Start a new test coverage session for the Enable test coverage task every time you run a pipeline. Starting a new session prevents deployments from failing if a test coverage session is already running on the pipeline.
  - Filter pipelines by application name and version on the Deployment Manager landing page.
  - In deployment logs, view all the new rule and data instances and all the changed rule and data instances that are in an application package that imported into a candidate system.

- Use APIs for new features.

  Deployment Manager now provides APIs so that you can provide for new features in your applications:

  - Run diagnostics remotely, and retrieve diagnostics results.
  - Disable and enable pipelines.
  - Archive and unarchive pipelines.

    The `Documentation/readme-for-swagger.md` file in the `DeploymentManager04_07_0x.zip` file provides documentation about API usage.

Deployment Manager 4.6.1

Deployment Manager 4.6.1 includes the following enhancements and resolved issues.

Enhancements

The following enhancements are available in this release:

- Ability to use Deployment Manager to automate data migration pipelines

  Data migration pipelines allow you export data from a production environment to a simulation environment where you can test impact of the changes made to your decision framework safely without having to deploy to a production environment. You can now use Deployment Manager to create data migration pipelines that allow you to automatically export data from a production environment and import it into a simulation environment. Additionally, you can configure a job scheduler rule to run pipelines during a specified period of time.

  For a tutorial on configuring simulation pipelines, including how to use Deployment Manager with them, see Deploying sample production data to a simulation environment for testing.

  For more information about configuring and using simulation pipelines with Deployment Manager, see Data migration pipelines with Deployment Manager 4.6.x.

- Ability to provide access to Dev Studio to a role

  You can now allow a role to access Dev Studio, and all the users of that role can switch to Dev Studio from the Operator icon. By being able to switch to Dev Studio, users can access Dev Studio tools to further troubleshoot issues that Deployment Manager cannot diagnose.

- Ability to easily move to new orchestration systems by configuring a dynamic system setting

  When you move from an existing orchestration system to a new one, you can now configure a dynamic system setting that specifies the URL of the new orchestration system.

Resolved issues in Deployment Manager 4.6.1

The following issues were resolved in this release:

- The position of the Validate test coverage task was not retained.

  If you added a Validate test coverage task in a pipeline, the task automatically moved under the Add task menu option after you saved the pipeline configuration. The position of the task is now saved.

- Deployment Manager installation failed on IBM Db2.

  Deployment Manager installations on systems running on Db2 failed with a database error. You can now install Deployment Manager on Db2.

- Not all API requests included PRRestService.

  Some HTTP requests to the api service package did not include PRRestService. It is now included in all requests if it is needed to direct all traffic to the API node.

- Tasks could not be added before the Deploy task in Deployment Manager 4.5.1 when using the API.

  When you used the API to create pipelines, you could not add tasks before the Deploy task, although you could add a task when you configured the pipeline in Deployment Manager. You can now add tasks before the Deploy task with the API.

- Test changes in branches were merged into incorrect ruleset versions.

  Sometimes, test changes in branches were merged into an incorrect ruleset version if multiple application versions were used and a test application was configured on the pipeline. Test changes in branches are now merged into the correct ruleset versions.

- Deployment Manager displayed a message for reaching the limit for pending changes.

  Sometimes, Deployment Manager displayed an error message that you reached the maximum limit for pending changes. The limit has been increased, and the error no longer appears.

- The Jenkins configuration diagnostics check failed when cross-site request forgery (CSRF) protection was disabled.

  When CSRF protection was disabled in Jenkins, pipeline diagnostics for Jenkins configuration failed with an error message that the Jenkins server was not reachable, even though the Jenkins task in the pipeline worked correctly. Jenkins diagnostics checks no longer fail in this scenario.

Deployment Manager 4.5.1

Deployment Manager 4.5.1 includes the following enhancements and resolved issues.

Enhancements

The following enhancements are provided in this release:

- Ability to add tasks before Deploy and Publish tasks

  For additional validation or environment provisioning, you can now add any task before the Deploy and Publish tasks, which are automatically added to the pipeline. You can add tasks before the Deploy task in any stage of the pipeline or before the Publish task in the development stage.

- Ability to associate bugs and user stories to branch merges

  When you start a deployment by submitting a branch into the Merge Branches wizard, you can now associate user stories and bugs from Agile Workbench so that you can track branch merges.

- New REST API to deploy existing artifacts

  Deployment Manager now provides a REST API to deploy existing artifacts so that you can start a production pipeline with the output of the development pipeline for the same application. You can view the **Documentation/readme-for-swagger.md** file for more information on using the API.

- Ability to access and pass all relevant parameters of the current deployment for Jenkins tasks

  For Jenkins tasks, you can now access and pass all the relevant Jenkins parameters for the current deployment, which include PipelineName, DeploymentID, RespositoryName, and ArtifactPath. When you configure the Jenkins task in a pipeline, the values of the parameters are automatically populated.

- More diagnostics to troubleshoot pipelines

You can now automatically diagnose more issues with your pipeline so that you spend less time manually troubleshooting. For example, you can now verify that Jenkins steps are properly configured, and you can also obtain more information about repository connections with enhanced troubleshooting tips.

- Elimination of post-upgrade steps when upgrading from Deployment Manager versions 3.2.1 and later

  For upgrades from Deployment Manager 3.2.1 or later to version 4.5.1, you no longer need to run activities or do any other post-upgrade steps. After the upgrade completes, Deployment Manager performs health checks before running post-upgrade steps for both on-premises and Pega Cloud Services environments.

Resolved issues

The following issue is resolved in Deployment Manager 4.5.1:

- Unable to configure keystores in Pega Cloud Services environments

  If your target environment is SSL-enabled with private certificates, you can now set the keystore for Deployment Manager connectors so that they can receive and process tokens. You first configure a keystore and then update a dynamic system setting to reference the keystore ID. For more information, see "Step 3a: Configuring authentication profiles on the orchestration server and candidate systems" for your version of *Installing, upgrading, and configuring Deployment Manager*.

Deployment Manager 4.4.2

Deployment Manager 4.4.2 includes the following resolved issues.

Resolved issues

The following issues were resolved in this release:

- Incorrect status displayed for the Run Pega unit test task

  If you refreshed a merge request quickly, the status of the Run Pega unit tests task might have been incorrectly displayed as the status of the merge. The correct status for the task is now displayed.

- Duplicate operator IDs displayed for the Manual task

  When you assigned manual tasks to an operator ID, the Manual task auto-complete displayed duplicate entries for the same operator ID if the operator ID was added as an administrator or user for multiple applications. The Manual task no longer displays duplicate entries.

- Pipeline deployments sometimes froze

  Sometimes, a pipeline deployment might freeze if it could not update the task with the status that it received from the task. The pipeline no longer freezes.

- No error messages displayed for issues with artifacts and repositories

  The Deploy existing artifact dialog box now validates the repository that you select. Error messages are also displayed when the repository does not list available artifacts or if the repository does not have any artifacts in it.

- Verify security checklist task failed and displayed a Pega Diagnostic Cloud (PDC) error

  The Verify security checklist failed when a pipeline had only one stage (development) and the Production ready check box was selected on the pipeline configuration. A PDC error message was displayed. The task no longer fails for pipelines with such a configuration.

- 32 character token limit for Jenkins tasks

  For the Jenkins task, you could only enter a 32 character token to remotely start a Jenkins job. You can now enter a token with more than 32 characters.

- Dependent applications were not deployed

  On pipelines on which dependent applications were configured, they were not deployed. They are now deployed correctly.

Deployment Manager 4.4.1

Deployment Manager 4.4.1 includes the following enhancements and known issues.

Enhancements

The following enhancements are provided in this release:

- Simplified configuration and workflow when merging branches in a distributed branch-based environment

  The process for merging branches in distributed branch-based environments has been simplified. On the remote development system, you can now merge branches and start a deployment by using the Merge Branches wizard to merge branches onto the source development system without having to use a Pega repository type.

- Ability to submit locked branches to the Merge Branches wizard

  You can now submit locked branches to the Merge Branches wizard so that you can follow best practices when working with branches. Best practices include locking branches to prevent changes from being made to them.

- Using the Merge Branches wizard to make merge requests now stores the branch in the development repository

  When you use the Merge Branches wizard to merge branches and start a deployment, the wizard now stores the branch in the development repository. Also, after the merge is completed, Deployment Manager deletes the branch from the development system. By storing branches in the development repository, Deployment Manager keeps a history, which you can view, of the branches in a centralized location.

- Ability to create separate product rules for test cases

  You can now separately manage both application changes and test cases in the same pipeline by using a separate product rule that contains only test cases. You can also choose a stage until which test cases are deployed to ensure that test cases are not deployed on environments such as staging and production, where they might not be needed. When you create test and production applications in Deployment Manager on your development system by using the New Application wizard, the wizard automatically creates separate product rules for your production and test applications.

- API documentation now available

  Documentation for Deployment Manager APIs is now included in the Documentation/readme-for-swagger.md file. This file is included in the DeploymentManager04_04_0x.zip file, which you can download from Pega Exchange. For example, you can quickly create pipelines without using the Deployment Manager interface.

- Usability enhancements
  - For the Check guardrail compliance task, the default guardrail compliance score has been increased to 97.
  - Email notifications for Jenkins jobs now include a link to the Jenkins job.
  - You can now start a Jenkins job when Jenkins has cross-site request forgery (CSRF) protection enabled.
  - For pipelines that have Jenkins tasks, job history details for successful deployments have a link to the Jenkins job.
  - The Pipeline list in the Merge Branches wizard no longer displays pipelines that are not configured to support branches; previously, you received an error after submitting pipelines that did not support branches.
  - If you are using the Merge Branches Wizard but do not have pipelines configured for an application, you can use still use the wizard to merge branches into target applications.

Known issues

The following are known issues in this release:

- The Pega Platform 8.1 and 8.2 versions of the Rule rebasing and Rebasing rules to obtain latest versions help topics should state that rule rebasing is supported in Deployment Manager.
- The Publishing a branch to a repository help topic should state that you can use Deployment Manager to start a deployment by publishing a branch to the source development system even if you have multiple pipelines per application version. Also, the note in this help topic no longer applies.

Deployment Manager 4.3.2

Deployment Manager 4.3.2 includes the following resolved issues.

Resolved issues

The following issue has been resolved:

- Pipelines not visible on the Deployment Manager landing page

  On systems running Pega CRM applications, pipelines were not visible on the Deployment Manager landing page when the datapage/newgenpages dynamic system setting was set to false. This setting disabled the new clipboard implementation for optimized read-only data pages. Pipelines are now visible regardless of the dynamic system setting value.

Deployment Manager 4.3.1

Deployment Manager 4.3.1 includes the following enhancements.

Enhancements

The following enhancements are provided in this release:

- Ability to configure notifications in Deployment Manager

  You can now configure notifications in Deployment Manager without having to configure an email account and listener in Dev Studio. You can also choose which notifications to receive such as whether Pega unit test tasks succeeded or failed. You can receive notifications through email, in the notification gadget, or both, and you can create custom notification channels to receive notifications through other means such as text messages or mobile push notifications. **Note:** To use notifications, you must install or upgrade to Pega Platform 8.1.3 on the orchestration server.

- Publishing application changes has been consolidated with viewing application versions in App Studio

  You can now publish application changes in App Studio and view information about your Deployment Manager application versions on one page. By accessing publishing features and viewing information in one place, you can more intuitively use Deployment Manager with App Studio.

Deployment Manager 4.2.1

Deployment Manager 4.2.1 includes the following enhancements.

Enhancements

The following enhancements are provided in this release:

- Ability to add and manage roles, privileges, and users

  Deployment Manager now provides default roles that specify privileges for super administrators and application administrators. Super administrators can add roles and specify their privileges, and both super administrators and application administrators can add users and assign them roles for specified applications. By specifying roles and privileges for Deployment Manager users, you can manage your users more effectively by controlling access to features for each type of user.

- New Deployment Manager portal

  Deployment Manager now provides a dedicated Deployment Manager portal that does not require access to the Dev Studio portal to access Deployment Manager features. The portal also provides enhancements such as a navigation panel from which you can easily access features such as reports, without having to open specific pipelines. Additionally, when you add a pipeline or modify pipeline settings, you can now open the rule forms for repositories and authentication profiles in Dev Studio from within Deployment Manager.

- Ability to merge branches that span multiple application layers

  You can now merge a branch that has rulesets that are in multiple applications if all the rulesets are in the application stack for the pipeline application. By doing so, you can, for example, merge changes that affect both a framework and an application layer. You can also merge test assets with the rules that you are testing without the test assets and rules being in the same application.

Deployment Manager 4.1.1

Deployment Manager 4.1.1 includes the following enhancements.

Enhancements

The following enhancements are provided in this release:

- Redesigned, more intuitive landing page and user interface

  Deployment Manager has been redesigned to have a more intuitive interface so that you can quickly access features as you interact with your pipeline. The Deployment Manager landing page now displays a snapshot of your pipeline configuration, which provides status information such as whether a deployment failed and on what stage the failure occurred. Additionally, when you click a pipeline to open it, Deployment Manager now displays important information about your pipeline such as the number of branches that are queued for merging on the development system.

- Manage aged updates

  You can now manage rules and data types, which are in an application package, that are older than the instances that are on a system. By importing aged updates, skipping the import, or manually deploying application packages on a system, you have more flexibility in determining the application contents that you want to deploy.

- New testing tasks, which include running Pega scenario tests

Several new test tasks have been added so that you deliver higher quality software by ensuring that your application meets the test criteria that you specify. On the candidate systems in your pipeline, you can now perform the following actions:
    - Run Pega scenario tests, which are end-to-end, UI-based tests that you create within Pega Platform.
    - Start and stop test coverage at the application level to generate a report that identifies the executable rules in your application that are covered or not covered by tests.
    - Refresh the Application Quality dashboard with the latest information so that you can see the health of your application and identify areas that need improvement before you deploy your application.

- Enhancements to publishing application changes to a pipeline in App Studio

  You can submit application changes to a pipeline in App Studio to start a deployment in Deployment Manager. The following enhancements have been made:
    - When you submit application changes into a pipeline, patch versions of the main application are now created.
    - You can now add comments, which will be published with your application.
    - You can now associate user stories and bugs with an application.
    - You can now view information such as who published the application and when for the application versions that you have submitted
- Run Pega unit tests on branches before merging

  You can now run Pega unit tests on branches before they are merged in the pipeline for either the pipeline application or an application that is associated with an access group. By validating your data against Pega unit tests, you can deploy higher quality applications.

Deployment Manager 3.4.1

Deployment Manager 3.4.1 includes the following enhancements.

Enhancements

The following enhancements are provided in this release:

- Manage aged updates

  You can now manage rules and data types, which are in an application package, that are older than the instances that are on a system. By importing aged updates, skipping the import, or manually deploying application packages on a system, you have more flexibility in determining the application contents that you want to deploy.

- Ability to merge branches that span multiple application layers

  You can now merge a branch that has rulesets that are in multiple applications if all the rulesets are in the application stack for the pipeline application. By doing so, you can, for example, merge changes that affect both a framework and an application layer. You can also merge test assets with the rules that you are testing without the test assets and rules being in the same application.

Deployment Manager 3.3.1

Deployment Manager 3.3.1 includes the following enhancements and known issues.

Enhancements

The following enhancements are provided in this release:

- New Verify security checklist task

  You can now use the Verify security checklist task to ensure that your pipeline complies with security best practices. It is automatically added to the stage before production when you create a pipeline.

- Ability to diagnose pipelines

  You can now diagnose your pipeline to verify information such as whether the target application and product rule are on the development environment, connectivity between systems and repositories is working, and pre-merge settings are correctly configured. You can also view troubleshooting tips and download logs.

Known issues

The following known issue exists in this release:

- Rollback does not work for Pega CRM applications

  If you are using a CRM application, you cannot roll back a deployment to a previous deployment.

Deployment Manager 3.2.1

Deployment Manager 3.2.1 includes the following enhancements.

Enhancements

The following enhancements are provided in this release:

- Simplified pipeline setup

  Pipeline setup has been simplified when you install Deployment Manager and when you configure pipelines. The following enhancements have been made:
  - Deployment Manager now provides the Pega Deployment Manager application with default operators and authentication profiles when you install it. You do not need to create authentication profiles for communication between candidate systems and the orchestration server.
  - If you are using Pega Cloud, Deployment Manager is automatically populated with the URLs of all the systems in your pipeline so that you do not need to configure them.

- New Check guardrail compliance task.

  You can now use the Check guardrail compliance task to ensure that the deployment does not proceed if the application does not comply with best practices for building applications in Pega Platform. This task is automatically added to all the stages in your pipeline.

- New Approve for production task

  Deployment Manager now provides an Approve for production task, which is automatically added to the stage before production when you create a pipeline. You can assign this task to a user who approves the application changes before the changes are deployed to production.

- Ability to specify the test suite ID and access group for Pega unit testing tasks

  For Pega unit testing tasks, you can now run all the Pega unit tests that are defined in a test suite for the application pipeline. By using a test suite ID, you can run a subset of Pega unit tests instead of all Pega unit tests for a pipeline application. You can also run all the Pega unit tests for an application that is associated with an access group so that you can run Pega unit tests for an application other than the pipeline application.

- Deployment Manager now supports first time deployments

  Deployment Manager now supports first-time deployments, so you do not have to import your application into each Pega Platform server on your candidate systems the first time that you configure Deployment Manager.

Deployment Manager 3.1.1

Deployment Manager 3.1.1 includes the following enhancements.

Enhancements

The following enhancements are provided in this release:

- Ability to create custom repository types

  You can now create custom repository types and manage your artifacts with them when you use Deployment Manager. For example, you can create a Nexus repository type and use it to move your application package between candidate systems in a pipeline. By creating custom repository types, you can use a wider variety of repository types with your artifacts to extend the functionality of Deployment Manager.

- Use the Merge Branches wizard to submit branches into a continuous integration and delivery pipeline.

  You can now submit branches into a continuous integration and delivery (CI/CD) pipeline by using the Merge Branches wizard in Designer Studio. Deployment Manager can then run pre-merge criteria on branches on one system so that you do not need to configure additional systems for both branch development and merging.

- Support for Pega Cloud.

  Beginning with Pega 7.4, all current and new Pega Cloud customers have a free dedicated sandbox to run Deployment Manager, which provides the following features:
  - Default repositories that store and move your application package between systems in the pipeline.
  - Ability to view, download, and remove application packages from repositories so that you can manage your cloud storage space.
  - Ability to deploy an existing application package.
  - Ability to create multiple pipelines for one version of an application.
  - Ability to create multiple pipelines for one version of an application. For example, you can create a pipeline with

only a production stage if you want to deploy a build to production separately from the rest of the pipeline.

- Ability to manage application package artifacts.

  You can now browse, download, and delete application package artifacts from the orchestration server. You do not have to log in to repositories to delete artifacts from them.

- Ability to move existing artifacts through pipelines.

  You can move existing artifacts through your pipelines. Existing artifacts are maintained in repositories, and you can move them through progressive stages in the pipeline.

Deployment Manager 2.1.4

Deployment Manager 2.1.4 includes the resolved issues.

Issues addressed in this release

The following issues were addressed in this release:

- Publishing application packages to the production repository sometimes fails in multinode environments

  In multinode staging environments, a node retrieves an application package from the development repository and places it into its service export folder to be published to the production repository. However, Deployment Manager sometimes cannot publish it to the production repository, because the request might be sent to a different node. This issue has been fixed so that if Deployment Manager sends a request to a node that does not have the application package, that node retrieves the package from the development repository and publishes it to the production repository.

Deployment Manager 2.1.3

Deployment Manager 2.1.4 includes the following enhancements.

Enhancements

The following enhancement is provided in this release:

- Improved structure and content of email notifications.

  Improvements have been made to email notifications that are sent to users when an event has occurred. For example, the email that is sent when a step on which PegaUnit test task fails now includes an attached log file that provides details of each failed PegaUnit test case.

Deployment Manager 2.1.2

Deployment Manager 2.1.2 includes the known issues.

Known issues

The following issue exists in this release:

- The PegaDevOps-ReleaseManager agent points to the wrong access group.

  To resolve the issue, after you import and install Deployment Manager 02.01.02, perform the following steps on the orchestration server:

  Because this agent is not associated with the correct access group, it cannot process Deployment Manager activities in the background.

  1. Update your Pega Platform application so that it is built on PegaDeploymentManager 02.01.02:
     1. In the Designer Studio header, click the name of your application, and then click **Definition**.
     2. In the **Built on application** section, in the **Version** field, press the Down Arrow key and select **02.01.02**.
     3. Click **Save**.
  2. Update the agent schedule for the Pega-DevOps-ReleaseManager agent to use the PegaDeploymentManager:Administrators access group.
     1. In Designer Studio, click **RecordsSysAdminAgent Schedule**.
     2. Click the **Pega-DevOps-ReleaseManager** agent.
     3. Click **Security**.
     4. In the **Access Group** field, press the Down Arrow key and select **PegaDeploymentManager:Administrators**.
     5. Click **Save**.

Deployment Manager 1.1.3

Deployment Manager 1.1.3 includes the following enhancements.

Enhancements

The following enhancement is provided in this release:

- Improved structure and content of email notifications

  Improvements have been made to email notifications that are sent to users when an event has occurred. For example, the email that is sent when a step on which PegaUnit test task fails now includes an attached log file that provides details of each failed PegaUnit test case.

Deployment Manager 1.1.2

Deployment Manager 1.1.2 includes the following known and resolved issues.

Known issues

The following issue exists in this release:

- The PegaDevOps-ReleaseManager agent points to the wrong access group.

  Because this agent is not associated with the correct access group, it cannot process Deployment Manager activities in the background.

  To resolve the issue, after you import and install Deployment Manager 01.01.02, perform the following steps on the orchestration server:

  1. Update your Pega Platform application so that it is built on PegaDeploymentManager 01.01.02:
     1. In the Designer Studio header, click the name of your application, and then click **Definition**.
     2. In the **Built on application** section, in the **Version** field, press the Down Arrow key and select **01.01.02**.
     3. Click **Save**.
  2. Update the agent schedule for the Pega-DevOps-ReleaseManager agent to use the PegaDeploymentManager:Administrators access group.
     1. In Designer Studio, click **Records SysAdmin Agent Schedule** .
     2. Click the **Pega-DevOps-ReleaseManager** agent.
     3. Click **Security**.
     4. In the **Access Group** field, press the Down Arrow key and select **PegaDeploymentManager:Administrators**.
     5. Click **Save**.

Resolved issues

The following issue was resolved in this release:

- Selections that were made to the Start build on merge check box were not applied when editing a pipeline.

  When you edit a pipeline and either select or clear the Start build on merge check box, your changes are now applied. Additionally, the check box is cleared by default.

Past release documentation

The Deployment Manager releases for the corresponding versions of documentation are no longer available to be downloaded from Pega Marketplace.

The following documentation is archived and available for reference:

- Deployment Manager 4.8.x

  Use Deployment Manager to configure and run continuous integration and delivery (CI/CD) workflows for your Pega applications from within Pega Platform. You can create a consistent deployment process so that you can deploy high-quality releases without the use of third-party tools.

- Deployment Manager 3.4.x

  Use Deployment Manager to configure and run continuous integration and delivery (CI/CD) workflows for your Pega applications from within Pega Platform. You can create a standardized deployment process so that you can deploy predictable, high-quality releases without using third-party tools.

Deployment Manager 4.8.x

Use Deployment Manager to configure and run continuous integration and delivery (CI/CD) workflows for your Pega applications from within Pega Platform. You can create a consistent deployment process so that you can deploy high-quality releases without the use of third-party tools.

With Deployment Manager, you can fully automate your CI/CD workflows, including branch merging; application package generation; artifact management; and package promotion, to different stages in the workflow.

Deployment Manager 4.8.x is compatible with Pega 8.1, 8.2, 8.3, and 8.4. You can download it for Pega Platform from the Deployment Manager Pega Marketplace page.

For answers to frequently asked questions, see the Deployment Manager FAQ page.**Note:** Each customer Virtual Private Cloud (VPC) on Pega Cloud Services has a dedicated orchestrator instance to use Deployment Manager. You do not need to install Deployment Manager to use it with your Pega Cloud application.**Note:** To use notifications, you must install or upgrade to Pega 8.1.3 on the orchestration server.

- Installing, upgrading, and configuring Deployment Manager 4.8.x

  Use Deployment Manager to create continuous integration and delivery (CI/CD) pipelines, which automate tasks and allow you to quickly deploy high-quality software to production.

- Configuring and running pipelines with Deployment Manager 4.8.x

  Use Deployment Manager to create continuous integration and delivery (CI/CD) pipelines, which automate tasks so that you can quickly deploy high-quality software to production.

- Using data migration pipelines with Deployment Manager 4.8.x

  Data migration tests provide you with significant insight into how the changes that you make to decision logic affect the results of your strategies. To ensure that your simulations are reliable enough to help you make important business decisions, you can deploy a sample of your production data to a dedicated data migration test environment.

Installing, upgrading, and configuring Deployment Manager 4.8.x

Use Deployment Manager to create continuous integration and delivery (CI/CD) pipelines, which automate tasks and allow you to quickly deploy high-quality software to production.

Note: You should make changes only in the development environment and then move them to higher environments. Do not make changes in any other environment. **Note:** Each customer virtual private cloud (VPC) on Pega Cloud Services has a dedicated orchestrator instance to use Deployment Manager. If you are upgrading from an earlier release, contact Pegasystems Global Client Support (GCS) support to request a new version. **Note:** This document describes the procedures for the latest version of Deployment Manager 4.8.x. To use notifications, you must install or upgrade to Pega 8.1.3 on the orchestration server.

For information on configuring Deployment Manager for data migration pipelines, see Installing, upgrading, and configuring Deployment Manager 4.8.x for data migration pipelines.

- Installing or upgrading to Deployment Manager 4.8.x

  You must install Deployment Manager if you are using it on-premises. Because Pega Cloud Services manages the orchestration server in any Pega Cloud subscription, Pega Cloud Services manages the installation and upgrades of Deployment Manager orchestration servers. Deployment Manager is supported on Pega Platform versions 8.1-8.5.

- Running post-upgrade steps

  If you are upgrading from Deployment Manager versions earlier than 3.2.1, you must run post-upgrade steps to complete the upgrade. Before you run post-upgrade steps, ensure that no deployments are running, have errors, or are paused. In Pega Cloud Services environments, the orchestration server name is similar to [environmentname]-DevOps.

- Configuring systems in the pipeline

  Configure the orchestration server and candidates in your pipeline for all supported CI/CD workflows. If you are using branches, you must configure additional settings on the development system after you perform the required steps.

- Configuring the development system for branch-based development

  If you are using branches in either a distributed or nondistributed branch-based environment, configure the development system so that you can start deployments when branches are merged. Configuring the development system includes defining the URL of the orchestration server, creating development and target applications, and locking application rulesets.

- Configuring additional settings

As part of your pipeline, users can optionally receive notifications through email when events occur. For example, users can receive emails when tasks or pipeline deployments succeed or fail. For more information about the notifications that users can receive, see .

Installing or upgrading to Deployment Manager 4.8.x

You must install Deployment Manager if you are using it on-premises. Because Pega Cloud Services manages the orchestration server in any Pega Cloud subscription, Pega Cloud Services manages the installation and upgrades of Deployment Manager orchestration servers. Deployment Manager is supported on Pega Platform versions 8.1-8.5.

To install Deployment Manager on-premises, do the following steps:

1. Install the latest supported version of Pega Platform on all systems in the pipeline.
2. On each system, browse to the Deployment Manager Pega Marketplace page, and then download the **DeploymentManager04.08.0x.zip** file for your version of Deployment Manager.
3. Extract the **DeploymentManager04.08.0x.zip** file.
4. Use the Import wizard to import files into the appropriate systems. For more information about the Import wizard, see Importing rules and data by using the Import wizard.
5. On the orchestration server, import the following files:
   - **PegaDevOpsFoundation_4.8.zip**
   - **PegaDeploymentManager_4.8.zip**
6. On the candidate systems, import the **PegaDevOpsFoundation_4.8.zip** file.
7. If you are using a distributed development for CI/CD workflows, on the remote development system, import the **PegaDevOpsFoundation_4.8.zip** file.
8. Do one of the following actions:
   - If you are upgrading from version 3.2.1 or later, the upgrade automatically runs, and you can use Deployment Manager when post-upgrade steps are run. You do not need to perform any of the required configuration procedures but can configure Jenkins and email notifications. For more information, see Configuring additional settings.
   - If you are not upgrading, continue the installation procedure at Configuring authentication profiles.

Running post-upgrade steps

If you are upgrading from Deployment Manager versions earlier than 3.2.1, you must run post-upgrade steps to complete the upgrade. Before you run post-upgrade steps, ensure that no deployments are running, have errors, or are paused. In Pega Cloud Services environments, the orchestration server name is similar to *[environmentname]*-DevOps.

If you are upgrading from Deployment Manager 3.2.1 or later, skip this section; otherwise, do the following steps:

1. On each candidate system, update the PegaDevOpsFoundation application version to the version of Deployment Manager that you are using.
   a. In the header of Dev Studio,, click the name of your application, and then click **Definition**.
   b. In the **Built on application** section for the PegaDevOpsFoundation application, in the **Version** field, press the Down arrow key and select the version of Deployment Manager that you are using.
   c. Click **Save**.
2. Modify the current release management application so that it is built on PegaDeploymentManager:4.7.x:
   a. In the header of Dev Studio, click the name of your application, and then click **Definition**.
   b. In the **Edit Application** rule form, on the **Definition** tab, in the **Built on application** section, for the PegaDeploymentManager application, press the Down arrow key and select **4.6**.
   c. Click **Save**.
3. If you do not see the pipelines that you created in earlier releases, run the *pxMigrateOldPipelinesTo42* activity:
   a. In the header of Dev Studio, search for pxMigrateOldPipelinesTo42, and then click the activity in the dialog box that displays the results.
   b. Click **Actions Run** .
   c. In the dialog box that is displayed, click **Run**.
4. On the orchestration server, run the *pxUpdateDescription* activity.
   a. In the header of Dev Studio search for pxUpdateDescription, and then click the activity in the dialog box that displays the results.
   b. Click **Actions Run** .
   c. In the dialog box that is displayed, click **Run**.
5. On the orchestration server, run the pxUpdatePipeline activity.
   a. In the header of Dev Studio, search for pxUpdatePipeline, and then click the activity in the dialog box that displays the results.
   b. Click **Actions Run** .
   c. In the dialog box that is displayed, click **Run**.
6. Merge rulesets to the PipelineData ruleset.
   a. Click **Configure System Refactor Rulesets** .
   b. Click **Copy/Merge RuleSet**.
   c. Click the **Merge Source RuleSet(s) to Target RuleSet** radio button.
   d. Click the **RuleSet Versions** radio button.
   e. In the **Available Source Ruleset(s)** section, select the first open ruleset version that appears in the list, and then click the **Move** icon. **Result:** All your current pipelines are stored in the first open ruleset.

7. If you modified this ruleset after you created the application, select all the ruleset versions that contain pipeline data.
    a. In the target **RuleSet/Information** section, in the **Name** field, press the Down arrow key and select **Pipeline Data**.
    b. In the **Version** field, enter 01-01-01.
    c. For the **Delete Source RuleSet(s) upon completion of merge?** option, click **No**.
    d. Click **Next**.
    e. Click **Merge** to merge your pipelines to the PipelineData:01-01-01 ruleset.
    f. Click **Done**.

Result: Your pipelines are migrated to the Pega Deployment Manager application. **What to do next:** Log out of the orchestration server and log back in to it with the DMReleaseAdmin operator ID and the password that you specified for it. **Note:** You do not need to perform any of the required steps in the remainder of this document. If you want to use Jenkins tasks to configure email notifications, see Configuring Jenkins.

Configuring systems in the pipeline

Configure the orchestration server and candidates in your pipeline for all supported CI/CD workflows. If you are using branches, you must configure additional settings on the development system after you perform the required steps.

To configure systems in the pipeline, do the following steps:

1. Configuring authentication profiles
2. Configuring the orchestration server
3. Configuring candidate systems
4. Creating repositories on the orchestration server and candidate systems

- Configuring authentication profiles

  Deployment Manager provides default operator IDs and authentication profiles. You must enable the default operator IDs and configure the authentication profiles that the orchestration server uses to communicate with the candidate systems.

- Configuring the orchestration server

  The orchestration server is the system on which the Deployment Manager application is installed and release managers configure and manage CI/CD pipelines. Configure settings on it before you can use it in your pipeline.

- Configuring candidate systems

  Configure each system that is used for the development, QA, staging, and production stage in the pipeline.

- Creating repositories on the orchestration server and candidate systems

  If you are using Deployment Manager on premises, create repositories on the orchestration server and all candidate systems to move your application between all the systems in the pipeline. You can use a supported repository type that is provided in Pega Platform, or you can create a custom repository type.

Configuring authentication profiles

Deployment Manager provides default operator IDs and authentication profiles. You must enable the default operator IDs and configure the authentication profiles that the orchestration server uses to communicate with the candidate systems.

Configure the default authentication profile by following these steps:

1. On the orchestration server, enable the DMReleaseAdmin operator ID and specify its password.
    a. Log in to the orchestration server with administrator@pega.com/install.
    b. In the header of Dev Studio, click **Records Organization Operator ID** , and then click **DMReleaseAdmin**.
    c. On the **Edit Operator ID** rule form, click the **Security** tab.
    d. Clear the **Disable Operator** check box.
    e. Click **Save**.
    f. Click **Update password**.
    g. In the **Change Operator ID Password** dialog box, enter a password, reenter it to confirm it, and then click **Submit**.
    h. Log out of the orchestration server.
2. On each candidate system, which includes the development, QA, staging, and production systems, enable the DMAppAdmin operator ID.
   If you want to create your own operator IDs, ensure that they point to the PegaDevOpsFoundation application.
    a. Log in to each candidate system with administrator@pega.com/install.
    b. In the header of Dev Studio, click **Records Organization Operator ID** , and then click **DMAppAdmin**.
    c. In the **Explorer** panel, click the operator ID initials, and then click **Operator**.
    d. On the **Edit Operator ID** rule form, click the **Security** tab.
    e. Clear the **Disable Operator** check box.
    f. Click **Save**.
    g. Click **Update password**.

h. In the **Change Operator ID Password** dialog box, enter a password, reenter it to confirm it, and then click **Submit**.
i. Log out of each candidate system.

3. On each candidate system, update the DMReleaseAdmin authentication profile to use the new password. All candidate systems use this authentication profile to communicate with the orchestration server about the status of the tasks in the pipeline.
   a. Log in to each candidate system with the DMReleaseAdmin operator ID and the password that you specified.
   b. In the header of Dev Studio, click **Records Security Authentication Profile** .
   c. Click **DMReleaseAdmin**.
   d. On the Edit Authentication Profile rule form, click **Set password**.
   e. In the **Password** dialog box, enter the password, and then click **Submit**.
   f. Save the rule form.
4. On the orchestration server, modify the DMAppAdmin authentication profile to use the new password. The orchestration server uses this authentication profile to communicate with candidate systems so that it can run tasks in the pipeline.
   a. Log in to the orchestration server with the DMAppAdmin user name and the password that you specified.
   b. In the header of Dev Studio, click **Records Security Authentication Profile** .
   c. Click **DMAppAdmin**.
   d. On the **Edit Authentication Profile** rule form, click **Set password**.
   e. In the **Password** dialog box, enter the password, and then click **Submit**.
   f. Save the rule form.
5. If your target environment is SSL-enabled with private certificates, configure the Deployment Manager connectors so that they can receive and process tokens by doing setting the keystore:
   a. In the header of Dev Studio, create and configure a keystore. For more information, see Creating a keystore for application data encryption.
   b. Configure the *Pega-DeploymentManager/TrustStore* dynamic system setting to reference the keystore ID by clicking **Records SysAdmin Dynamic System Settings** .
      - If on Deployment Manager 4.8.4, this configuration is called *PegaDeploymentManagerIntegration/TrustStore*.
   c. Click the **Pega-DeploymentManager/TrustStore** dynamic system setting.
   d. On the **Settings** tab, in the **Value** field, enter the ID of the keystore that you created in the previous step.
   e. Click **Save**.

   For more information about dynamic system settings, see Creating a dynamic system setting.

6. Do one of the following actions:
   - If you are upgrading to Deployment Manager 4.8.x, resume the post-upgrade procedure from step 2. For more information, see Running post-upgrade steps.
   - If you are not upgrading, continue the installation procedure. For more information, see Configuring the orchestration server.

- Understanding default authentication profiles and operator IDs

  When you install Deployment Manager on all the systems in your ecosystem participating in the release process, there are applications, operator IDs, and authentication profiles installed by default. Authentication profiles enable communication between the orchestration server and candidate systems.

Understanding default authentication profiles and operator IDs

When you install Deployment Manager on all the systems in your ecosystem participating in the release process, there are applications, operator IDs, and authentication profiles installed by default. Authentication profiles enable communication between the orchestration server and candidate systems.

On the orchestration server, the following items are installed:

- The Pega Deployment Manager application.
- The DMReleaseAdmin operator ID, which release managers use to log in to the Pega Deployment Manager application. You must enable this operator ID and specify its password.
- The DMAppAdmin authentication profile. The orchestration server uses this authentication profile to communicate with candidate systems so that it can run tasks in the pipeline. You must update this authentication profile to use the password that you specified for the DMAppAdmin operator ID, which is configured on all the candidate systems.
- From version 5.x, Deployment Manager Service is the core system that the Portal and Agents are built on. The following items are installed:
  - The DMStudioUser authentication profile. The Deployment Manager Portal (GUI) uses this authentication profile to communicate with the Deployment Manager Services hosted on orchestrator system. Since the authentication profile type is oAuth2, the client secret gets updated by clicking the **Generate client secret** (Update authentication profile) button through the Deployment Manager Portal.
  - The DMAgentUser authentication profile. The Deployment Manager agent responsible for communicating task processes to candidate systems uses this authentication profile to fetch pending tasks from the Deployment Manager Services hosted on the orchestrator.

On all candidate systems, the following items are installed:

- The PegaDevOpsFoundation application.
- The DMAppAdmin operator ID, which points to the PegaDevOpsFoundation application. You must enable this operator ID and specify its password.
- The DMReleaseAdmin authentication profile. All candidate systems use this authentication profile to communicate with the orchestration server about the status of the tasks in the pipeline. You must update this authentication profile to use the password that you specified for the DMReleaseAdmin operator ID, which is configured on the orchestration server.
- From version 5.x, Deployment Manager Service is the core system that the Portal and Agents are built on. The following items are installed:
    - The DMReleaseAdmin_OAuth2 authentication profile. All candidate systems use this authentication profile to communicate with the orchestration server about the status of the tasks in the pipeline. Since the authentication profile type is oAuth2, the client secret must be updated with the client secret generated on the orchestrator (using the **Generate client secret** option from general settings on the Deployment Manager Portal).

Note: The DMReleaseAdmin and DMAppAdmin operator IDs do not have default passwords.

Configuring the orchestration server

The orchestration server is the system on which the Deployment Manager application is installed and release managers configure and manage CI/CD pipelines. Configure settings on it before you can use it in your pipeline.

To configure the orchestration server, do the following steps:

1. If your system is not configured for HTTPS, verify that TLS/SSL settings are not enabled on the api and cicd service packages.
    a. In the header of Dev Studio, click **Records Integration-Resources Service Package** .
    b. Click **api**.
    c. On the **Context** tab, verify that the **Requires authentication** check box is checked.
    d. On the **Context** tab, verify that the **Require TLS/SSL for REST services in this package** check box is cleared.
    e. Click **Records Integration-Resources Service Package** .
    f. Click **cicd**.
    g. On the **Context** tab, verify that the **Require TLS/SSL for REST services in this package** check box is cleared.
2. To move the orchestration server to a different environment, first migrate your pipelines to the new orchestration server, and then configure its URL on the new orchestration server.
   This URL will be used for callbacks and for diagnostics checks.
    a. In the header of Dev Studio, click **Create SysAdmin Dynamic System Settings** .
    b. In the **Owning Ruleset** field, enter *Pega-DeploymentManager*.
    c. In the **Setting Purpose** field, enter *OrchestratorURL*.
    d. Click **Create and open**.
    e. On the **Settings** tab, in the **Value** field, enter the URL of the new orchestration server in the format http://*hostname:port*/prweb.
    f. Click **Save**.

What to do next: Configure the candidate systems in your pipeline. For more information, see Configuring candidate systems.

Configuring candidate systems

Configure each system that is used for the development, QA, staging, and production stage in the pipeline.

To configure candidate systems, complete the following steps:

1. On each candidate system, add the PegaDevOpsFoundation application to your application stack.
    a. In the header of Dev Studio, click the name of your application, and then click **Definition**.
    b. In the **Built on application** section, click **Add application**.
    c. In the **Name** field, press the Down arrow key and select **PegaDevOpsFoundation**.
    d. In the **Version** field, press the Down arrow key and select the version of Deployment Manager that you are using.
    e. Click **Save**.
2. If your system is not configured for HTTPS, verify that TLS/SSL settings are not enabled on the api and cicd service packages.
    a. Click **Records Integration-Resources Service Package** .
    b. Click **api**.
    c. On the **Context** tab, verify that the **Requires authentication** check box is checked.
    d. On the **Context** tab, verify that the **Require TLS/SSL for REST services in this package** check box is cleared.
    e. Click **RecordsIntegration-ResourcesService Package**.
    f. Click **cicd**.
    g. On the **Context** tab, verify that the **Require TLS/SSL for REST services in this package** check box is cleared.
3. To use a product rule for your target application, test application, or both, other than the default rules that are created

by the New Application wizard, on the development system, create product rules that define the test application package and the target application package that will be moved through repositories in the pipeline. For more information, see Creating a product rule that includes associated data.

When you use the New Application wizard, a default product rule for your target application is created that has the same name as your application. Additionally, if you are using a test application, a product rule is created with the same name as the target application, with _Tests appended to the name.

4. If you are on Pega Platform 8.1 to Pega Platform 8.5.1 and you are setting up the candidate with 4.8.4 Pega DevOps Foundation. Please proceed with the steps below is the target environment is SSL-enabled with private certificates, configure the Deployment Manager connectors such that they can receive and process tokens by setting the KeyStore; For more information, see Creating a keystore for application data encryption.
    a. Configure the *PegaDeploymentManagerIntegration/TrustStore* dynamic system setting to reference the keystore ID by clicking **Records SysAdmin Dynamic System Settings** .
    b. In the **Value** field, enter the ID of the keystore that you created in the previous step.
    c. Click **Save**.

What to do next: Configure repositories through which to move artifacts in your pipeline. For more information, see Creating repositories on the orchestration server and candidate systems

Creating repositories on the orchestration server and candidate systems

If you are using Deployment Manager on premises, create repositories on the orchestration server and all candidate systems to move your application between all the systems in the pipeline. You can use a supported repository type that is provided in Pega Platform, or you can create a custom repository type.

If you are using Deployment Manager on Pega Cloud Services, default repositories, named "pegacloudcustomerroot" for both the development and production repositories, are provided. If you want to use repositories other than the ones provided, you can create your own.

For more information about creating a supported repository, see Creating a repository.

For more information about creating a custom repository type, see Creating and using custom repository types for Deployment Manager.

When you create repositories, note the following information:

- You cannot use the Pega repository type to store application artifacts for the following reasons:
    - The Pega repository type points to the temporary folder where the Pega Platform node that is associated with Deployment Manager stores caches. This node might not be persistent.
    - This repository type is suitable only for single node deployments. In multinode deployments, each time a requestor is authenticated, the requestor could be in a different node, and published artifacts are not visible to the repository.
    - At most companies, the security practice is that lower environments should not connect to higher environments. Using a Pega repository typically means that a lower environment can access a higher environment.

      You can only use Pega type repositories if are rebasing your development system to obtain the most recently committed rulesets after merging them.

      You can use file system type repositories if you do not want to use proprietary repositories such as Amazon s3 or JFrog Artifactory.
- You cannot use the defaultstore repository type to host artifacts or product archives for the production applications. It is a system-managed file system repository; it points to the temporary folder where the Pega Platform node that is associated with Deployment Manager stores caches.
- Ensure that each repository has the same name on all systems.
- When you create JFrog Artifactory repositories, ensure that you create a Generic package type in JFrog Artifactory. Also, when you create the authentication profile for the repository on Pega Platform, you must select the **Preemptive authentication** check box.

After you configure a pipeline, you can verify that the repository connects to the URL of the development and production repositories by clicking **Test Connectivity** on the **Repository** rule form.

Configuring the development system for branch-based development

If you are using branches in either a distributed or nondistributed branch-based environment, configure the development system so that you can start deployments when branches are merged. Configuring the development system includes defining the URL of the orchestration server, creating development and target applications, and locking application rulesets.

1. On the development system (in nondistributed environment) or the main development system (in a distributed environment), create a dynamic system setting to define the URL of the orchestration server, even if the orchestration server and the development system are the same system.
    a. Click **Create Records SysAdmin Dynamic System Settings** .

b. In the **Owning Ruleset** field, enter PegaDevOpsShared.
c. In the **Setting Purpose** field, enter RMURL.
d. Click **Create and open**.
e. On the **Settings** tab, in the **Value** field, enter the URL of the orchestration server in the format http://*hostname:port*/prweb/PRRestService.
f. Click **Save**.

For more information about dynamic system settings, see Creating a dynamic system setting

2. Complete the following steps on either the development system (in a non-distributed environment) or the remote development system (in a distributed environment).
   a. Use the New Application wizard to create a new development application that developers will log in to. This application allows development teams to maintain a list of development branches without modifying the definition of the target application.
   b. Add the target application of the pipeline as a built-on application layer of the development application by first logging into the application.
   c. In the header of Dev Studio, click the name of your application, and then click **Definition**.
   d. In the **Built-on application** section, click **Add application**.
   e. In the **Name** field, press the Down arrow key and select the name of the target application.
   f. In the **Version** field, press the Down arrow key and select the target application version.
   g. Click **Save**.
3. Lock the application rulesets to prevent developers from making changes to rules after branches have been merged.
   a. In the header of Dev Studio, click the name of your application, and then click **Definition**.
   b. In the **Application rulesets** section, click the **Open icon** for each ruleset that you want to lock.
   c. Click **Lock and Save**.
4. Copy the development repository that you configured on the remote development system to the source development system.
5. If you are managing test cases separately from the target application, create a test application. For more information, see Managing test cases separately in Deployment Manager.
6. **Optional:** To rebase your development application to obtain the most recently committed rulesets after you merge your branches, configure Pega Platform so that you can use rule rebasing. For more information, see Understanding rule rebasing.

Configuring additional settings

As part of your pipeline, users can optionally receive notifications through email when events occur. For example, users can receive emails when tasks or pipeline deployments succeed or fail. For more information about the notifications that users can receive, see Understanding email notifications.

For either new Deployment Manager installations or upgrades, you must configure settings on the orchestration server so that users can receive email notifications. For more information, see Configuring email accounts on the orchestration server.

Additionally, you can configure Jenkins if you are using Jenkins tasks in a pipeline. For more information, see Configuring Jenkins.

- Configuring email accounts on the orchestration server

  Deployment Manager provides the Pega-Pipeline-CD email account and the DMEmailListener email listener.

- Configuring Jenkins

  If you are using a Run Jenkins step task in your pipeline, configure Jenkins so that it can communicate with the orchestration server.

Configuring email accounts on the orchestration server

Deployment Manager provides the Pega-Pipeline-CD email account and the DMEmailListener email listener.

If you are configuring email accounts for the first time, specify your details for this account in Pega Platform. For more information, see Configuring email accounts for new Deployment Manager installations.

Otherwise, if you are upgrading, do the appropriate steps for the email account that you are using. See one of the following topics for more information:

- Configuring an email account when upgrading and using the Pega-Pipeline-CD email account
- Configuring email accounts when upgrading and using the Default email account.

- Configuring email accounts for new Deployment Manager installations

  For new Deployment Manager installations, on the orchestration server, configure the Pega-Pipeline-CD email account so users can receive email notifications for events such as task completion or failure.

- Configuring an email account when upgrading and using the Pega-Pipeline-CD email account

If you are upgrading to Deployment Manager 4.8.x and using the Pega-Pipeline-CD email account for sending emails, the DMEmailListener email listener always listens to the Pega-Pipeline-CD account. If you are using a different listener, you must delete it.

- Configuring email accounts when upgrading and using the Default email account

  If you are upgrading to Deployment Manager and using the Default email account, after you upgrade to Deployment Manager 4.7.x, you must do certain steps so that you can send email notifications.

- Understanding email notifications

  Emails are preconfigured with information about each notification type. For example, when a deployment failure occurs, the email that is sent provides information, such as the pipeline name and URL of the system on which the deployment failure occurred.

Configuring email accounts for new Deployment Manager installations

For new Deployment Manager installations, on the orchestration server, configure the Pega-Pipeline-CD email account so users can receive email notifications for events such as task completion or failure.

Do the following steps:

1. In the navigation pane of Dev Studio, click **Records**, and then click **Integration-Resources Email Account** .
2. Click **Pega-Pipeline-CD**.
3. In the **Edit Email Account** rule form, configure and save the email account.
   For more information about configuring email accounts, see Creating an email account in Dev Studio.

Configuring an email account when upgrading and using the Pega-Pipeline-CD email account

If you are upgrading to Deployment Manager 4.8.x and using the Pega-Pipeline-CD email account for sending emails, the DMEmailListener email listener always listens to the Pega-Pipeline-CD account. If you are using a different listener, you must delete it.

Delete the listener that is listening to the Pega-Pipeline-CD account by doing the following steps:

1. In the header of Dev Studio, click **Configure Integration Email Email listeners** .
2. On the **Email: Integration** page, on the **Email Listeners** tab, click the listener that you want to delete.
3. Click **Delete**.

Configuring email accounts when upgrading and using the Default email account

If you are upgrading to Deployment Manager and using the Default email account, after you upgrade to Deployment Manager 4.7.x, you must do certain steps so that you can send email notifications.

Do the following steps:

1. Update the email sender and recipient in Pega Platform.
   a. In the navigation pane of Dev Studio,, click **Records**, and then click **Integration-Resources Email Account** .
   b. Click **Default**.
   c. On the **Edit Email Account** form, configure and save the email account.
      For more information about configuring email accounts, see Creating an email account in Dev Studio
2. If you have an email listener that listens to the same email address that you configured in Deployment Manager in the previous step, delete the listener to ensure that the DMEmailListener is listening to the email account that you configured.
   a. In the header of Dev Studio,, click **Configure Integration Email Email listeners** .
   b. On the **Email: Integration** page, on the **Email Listeners** tab, click the listener that you want to delete.
   c. Click **Delete**.

Understanding email notifications

Emails are preconfigured with information about each notification type. For example, when a deployment failure occurs, the email that is sent provides information, such as the pipeline name and URL of the system on which the deployment failure occurred.

Preconfigured emails are sent in the following scenarios:

- Deployment start – When a deployment starts, an email is sent to the release manager and, if you are using branches, to the operator who started a deployment.
- Deployment step completion or failure – When a step either completes or fails, an email is sent to the release manager and, if you are using branches, to the operator who started the branch merge. The deployment pauses if

there are any errors.

- Deployment completion – When a deployment is successfully completed, an email is sent to the release manager and, if you are using branches, to the operator who started the branch merge.
- Stage completion or failure – When a stage in a deployment process either succeeds or fails, an email is sent to the release manager and, if you are using branches, to the operator who started the branch merge.
- Manual tasks requiring approval – When a manual task requires email approval from a user, an email is sent to the user, who can approve or reject the task from the email.
- Stopped deployment – When a deployment is stopped, an email is sent to the release manager and, if you are using branches, to the operator who started the branch merge.
- Pega unit testing success or failure – If you are using the Run Pega unit tests task, and the task either succeeds or fails, an email is sent to the release manager and, if you are using branches, to the operator who started the branch merge.
- Schema changes required – If you do not have the required schema privileges to deploy schema changes on application packages that require those changes, an email is sent to the operator who started the deployment.
- Guardrail compliance score success or failure – If you are using the Check guardrail compliance task, an email is sent to the release manager if the task either succeeds or fails.
- Approve for production – If you are using the Approve for production task, which requires approval from a user before application changes are deployed to production, an email is sent to the user. The user can reject or approve the changes.
- Verify security checklist success or failure – If you are using the Verify security checklist task, which requires that all tasks be completed in the Application Security Checklist to ensure that the pipeline complies with security best practices, an email is sent to the release manager if the test either succeeds or fails.
- Pega scenario testing success or failure – If you are using the Run Pega scenario tests task, an email is sent to the release manager and, if you are using branches, to the operator who started the branch merge, if Pega scenario testing either succeeds or fails.
- Start test coverage success or failure – If you are using the Enable test coverage task to generate a test coverage report, an email is sent to the release manager if the task either fails or succeeds.
- Verify test coverage success or failure – If you are using the Verify test coverage task, an email is sent to the release manager if the task either fails or succeeds.
- Application quality statistics refreshed – If you are using the Refresh application quality statistics task, an email is sent to the release manager when the task is run.
- Jenkins job success or failure – If you are using a Jenkins task, an email is sent to the release manager if a Jenkins job either succeeds or fails.

Configuring Jenkins

If you are using a Run Jenkins step task in your pipeline, configure Jenkins so that it can communicate with the orchestration server.

1. On the orchestration server, create an authentication profile that uses Jenkins credentials.
   - If you are using a version of Jenkins earlier than 2.17.6, create an authentication profile on the orchestration server that specifies the credentials to use.
     1. Click **Create Security Authentication Profile** .
     2. Enter a name, and then click **Create and open**.
     3. In the **User name** field, enter Jenkins user ID.
     4. Click **Set password**, enter the Jenkins password, and then click **Submit**.
     5. Click the **Preemptive authentication** check box.
     6. Click **Save**.
     7. Go to step 4.

     For more information about configuring authentication profiles, see Creating an authentication profile.

   - If you are using Jenkins 2.17.6 or later and want to use an API token for authentication, go to step 2.
   - If you are using Jenkins 2.17.6 or later and want to use a Crumb Issuer for authentication, go to step 3.
2. If you are using Jenkins version 2.17.6 or later and want to use an API token for authentication, do the following steps:
   a. Log in to the Jenkins server.
   b. Click **People**, click the user who is running the Jenkins job, and then click **Configure API token** .
   c. Generate the API token.
   d. Create an authentication profile on the orchestration server by clicking **Create Security Authentication Profile** .
   e. In the **User name** field, enter the Jenkins user ID.
   f. Click **Set password**, enter the API token that you generated, and then click **Submit**.
   g. Click the **Preemptive authentication** check box.
   h. Click **Save**.
   i. Go to step 4.

   For more information about configuring authentication profiles, see Creating an authentication profile.

3. If you are using Jenkins version 2.17.6 or later and want to use a Crumb Issuer for authentication, do the following steps:
   a. Log in to the Jenkins server.
   b. Click **Manage Jenkins Manage Plugins** and select the check box for the **Strict Crumb Issuer** plug-in.
   c. Click **Manage Jenkins Configure Global Security** .

d. In the **CSRF protection** section, in the **Crumb Issuer** list, select **Strict Crumb Issuer**.
e. Click **Advanced**, and then clear the **Check the session ID** check box.
f. Click **Save**.
g. Create an authentication profile on the orchestration server by clicking **Create Security Authentication Profile** .
h. In the **User name** field, enter the Jenkins user ID.
i. Click **Set password**, enter the Jenkins password, and then click **Submit**.
j. Click the **Preemptive authentication** check box.
k. Click **Save**.
l. Go to step 4.

For more information about configuring authentication profiles, see Creating an authentication profile.

4. Install the Post build task plug-in.
5. Install the curl command on the Jenkins server.
6. Create a new freestyle project.
7. On the **General** tab, select the **This project is parameterized** check box.
8. Add the BuildID and CallBackURL parameters.
   a. Click **Add parameter**, and then select **String parameter**.
   b. In the **String** field, enter BuildID.
   c. Click **Add parameter**, and then select **String parameter**.
   d. In the **String** field, enter CallBackURL.
9. To add parameters that you can use in Run Jenkins step tasks in the pipeline, click **Add parameter**, select **String parameter**, and enter the string of the parameter. The system automatically populates these values in Jenkins tasks. You can add any of the following strings:
   a. If you are using Jenkins tasks for continuous integration (you are using branches), add any of the following strings:
      - **PipelineName**: Pipeline name on which the Run Jenkins step task is configured.
      - **ApplicationName**: Application for which the pipeline is configured.
      - **RepositoryName**: Repository to which the merged branch is published.
      - **BranchName**: Branch for which the Run Jenkins step task is configured.
      - **BranchFilePath**: Location of the branch.
      - **OrchestratorURL**: URL of the orchestration server. Add this parameter to stop a pipeline when the Run Jenkins step fails in a pipeline.
      - **PipelineID**: ID of the pipeline on which the Run Jenkins step task is configured.
   b. If you are using Run Jenkins step tasks for continuous delivery (in staging, QA, or production environments), add any of the following strings:
      - **PipelineName**: Pipeline name on which the Run Jenkins step task is configured.
      - **RepositoryName**: Repository that the Deploy task uses for the stage (for example, staging) on which the Run Jenkins step task is configured.
      - **DeploymentID**: ID of the current deployment.
      - **DeploymentArtifactName**: Artifact name that the Deploy task uses on the stage on which the Run Jenkins step task is configured.
      - **StartedBy**: ID of the operator who started the deployment.
      - **CurrentStage**: Name of the stage on which the Run Jenkins step task is configured.
      - **ArtifactPath**: Full path to the artifact that the Deploy task uses.
      - **OrchestratorURL**: URL of the orchestration server. Add this parameter to stop a pipeline when the Run Jenkins step fails in a pipeline.
      - **PipelineID**: ID of the pipeline on which the Run Jenkins step task is configured. Add this parameter to stop a pipeline when the Run Jenkins step task fails in a pipeline.
10. In the **Build Triggers** section, select the **Trigger builds** remotely check box.
11. In the **Authentication Token** field, select the token that you want to use when you start Jenkins jobs remotely.
12. In the **Build Environment** section, select the **Use Secret text(s) or file(s)** check box.
13. In the **Bindings** section, do the following actions:
    a. Click **Add**, and then select **User name and password (conjoined)**.
    b. In the **Variable** field, enter RMCREDENTIALS
    c. In the **Credentials** field, click **Specific credentials**.
    d. Click **Add**, and then select **Jenkins**.
    e. In the **Add credentials** dialog box, in the **Username** field, enter the operator ID of the release manager operator that is configured on the orchestration server.
    f. In the **Password** field, enter the password.
    g. Click **Save**.
14. Add post-build tasks by doing one of the following actions:
    a. If Jenkins is running on Microsoft Windows, go to step 15.
    b. If Jenkins is running on Linux, go to step 16.
15. If Jenkins is running on Microsoft Windows, add the following post-build tasks:
    a. Click **Add post-build** action, and then select **Post build task**.
    b. In the **Post-Build Actions** section, in the **Log text** field, enter a unique string for the message that is displayed in the build console output when a build fails, for example BUILD FAILURE**.**
    c. In the **Script** field, enter curl --user %RMCREDENTIALS% -H "Content-Type: application/json" -X POST --data "{\"jobName\":\"%JOB_NAME%\",\"buildNumber\":\"%BUILD_NUMBER%\",\"pyStatusValue\":\"FAIL\",\"pyID\":\"%BuildID%\"}" "%CallBackURL%"**.**
    d. Click **Add another task**.
    e. In the **Post-Build Actions** section, in the **Log text** field, enter a unique string for the message that is displayed

in the build console output when a build is successful, for example BUILD SUCCESS**.**
    f. In the **Script** field, enter curl --user %RMCREDENTIALS% -H "Content-Type: application/json" -X POST --data "{\"jobName\":\"%JOB_NAME%\",\"buildNumber\":\"%BUILD_NUMBER%\",\"pyStatusValue\":\"SUCCESS\",\"pyID\":\"%BuildID%\"}" "%CallBackURL%"
    g. Click **Save**.
    h. Go to step 17.
16. If Jenkins is running on Linux, add the following post-build tasks. Use the dollar sign ($) instead of the percent sign (%) to access the environment variables:
    a. Click **Add post-build action**, and then select **Post build task**.
    b. In the **Log text** field, enter a unique string that for the message that is displayed in the build console output when a build fails, for example BUILD FAILURE**.**
    c. In the **Script** field, enter curl --user $RMCREDENTIALS -H "Content-Type: application/json" -X POST --data "{\"jobName\":\"$JOB_NAME\",\"buildNumber\":\"$BUILD_NUMBER\",\"pyStatusValue\":\"FAIL\",\"pyID\":\"$BuildID\"}" "$CallBackURL"
    d. Click **Add another task**.
    e. In the **Log text** field, enter a unique string that for the message that is displayed in the build console output when a build is successful, for example BUILD SUCCESS**.**
    f. In the **Script** field, enter curl --user $RMCREDENTIALS -H "Content-Type: application/json" -X POST --data "{\"jobName\":\"$JOB_NAME\",\"buildNumber\":\"$BUILD_NUMBER\",\"pyStatusValue\":\"SUCCESS\",\"pyID\":\"$BuildID\"}" "$CallBackURL"
    g. Click **Save**.
    h. Go to step 17.
17. To stop a pipeline deployment if a Jenkins build fails, add a post-build script:
    a. Click **Add post-build action**, and then select **Post build task**.
    b. In the **Log text** field, enter a unique string for the message that is displayed in the build console output when a build fails, for example JENKINS BUILD FAILURE**.**
    c. In the **Script** field, enter curl --user %RMCREDENTIALS% -H "Content-Type: application/json" -X PUT --data "{"AbortNote":"Aborted from jenkins job"}" %OrchestratorURL%/PRRestService/cicd/v1/pipelines/%PipelineID%/builds/%DeploymentID%/abort
    d. Click **Save**.

Configuring and running pipelines with Deployment Manager 4.8.x

Use Deployment Manager to create continuous integration and delivery (CI/CD) pipelines, which automate tasks so that you can quickly deploy high-quality software to production.

On the orchestration server, release managers use the Deployment Manager landing page to configure CI/CD pipelines for their Pega Platform applications. The landing page displays all the running and queued application deployments, branches that are to be merged, and reports that provide information about your DevOps environment such as key performance indicators (KPIs).

Note: These topics describes the features for the latest version of Deployment Manager 4.8.x. **Note:** To use notifications, you must install or upgrade to Pega 8.1.3 on the orchestration server.

For more information about using Deployment Manager and data migration pipelines, see Exporting and importing simulation data automatically with Deployment Manager.

- Logging in to Deployment Manager

  Deployment Manager provides a dedicated portal from which you can access features.

- Accessing the Dev Studio portal

  If your role has the appropriate permission, you can access Dev Studio from within Deployment Manager. You can switch to Dev Studio to access features such as additional tools to troubleshoot issues. You can also open, modify, and create repositories and authentication profiles.

- Accessing API documentation

  Deployment manager provides REST APIs for interacting with resources that in the Deployment Manager interface. Use these APIs to create and manage pipelines by using automated scripts or external information.

- Understanding roles and users

  Define roles and users to manage which users can access Deployment Manager and which features they can access. For example, you can create a role that does not permit users to delete pipelines for a specific application.

- Understanding Deployment Manager notifications

  You can enable notifications to receive updates about the events that occur in your pipeline. For example, you can choose to receive emails about whether unit tests failed or succeeded. You can receive notifications in the Deployment Manager notifications gadget, through email, or both. By default, all notifications are enabled for users who are configured in Deployment Manager.

- Configuring an application pipeline

  When you add a pipeline, you specify merge criteria and configure stages and steps in the continuous delivery workflow. For example, you can specify that a branch must be peer-reviewed before it can be merged, and you can specify that Pega unit tests must be run after a branch is merged and is in the QA stage of the pipeline.

- Accessing systems in your pipeline

  You can open the systems in your pipeline and log in to the Pega Platform instances on each system. For example, you can access the system on which the QA stage is installed.

- Filtering pipelines in the dashboard

  You can filter the pipelines that the dashboard displays by application name, version, and pipeline deployment status. By filtering pipelines, the dashboard displays only the information that is relevant to you.

- Viewing merge requests

  You can view the status of the merge requests for a pipeline to gain more visibility into the status of your pipeline. For example, you can see whether a branch was merged in a deployment and when it was merged.

- Viewing deployment reports for a specific deployment

  Deployment reports provide information about a specific deployment. You can view information such as the number of tasks that you configured on a deployment that have been completed and when each task started and ended.

- Starting deployments

  You can start deployments in a number of ways. For example, you can start a deployment manually if you are not using branches, by submitting a branch into the Merge Branches wizard, or by publishing application changes in App Studio to create a patch version of your application. Your user role determines if you can start a deployment.

- Pausing a deployment

  When you pause a deployment, the pipeline completes the task that it is running, and stops the deployment at the next step.

- Stopping a deployment

  Stop a deployment to discontinue processing.

- Performing actions on a deployment that has errors

  If a deployment has errors, the pipeline stops processing on it. You can perform actions on it, such as rolling back the deployment or skipping the step on which the error occurred.

- Rolling back a deployment

  A rollback can be performed on any deployment that encounters an error or fails for any reason. The rollback feature relies on restore points, which are automatically generated every time an import happens. Any change implemented after the import but before the next restore point will be lost when the rollback action is triggered.

- Performing ad hoc tasks

  By using ad hoc tasks, you can independently perform package and deploy actions outside of the context of a deployment, which is useful for exporting artifacts from the production environment and importing them into a lower environment, or for packaging and deploying artifacts if a Deployment Manager pipeline is not available.

- Troubleshooting issues with your pipeline

  Deployment Manager provides several features that help you troubleshoot and resolve issues with your pipeline.

- Understanding schema changes in application packages

  If an application package that is to be deployed on candidate systems contains schema changes, the Pega Platform orchestration server checks the candidate system to verify that you have the required privileges to deploy the schema changes. One of the following results occurs:

- Completing or rejecting a manual step

  If a manual step is configured on a stage, the deployment pauses when it reaches the step, and you can either complete it or reject it if your role has the appropriate permissions. For example, if a user was assigned a task and completed it, you can complete the task in the pipeline to continue the deployment. Deployment Manager also sends you an email when there is a manual step in the pipeline. You can complete or reject a step either within the pipeline or through email.

- Managing aged updates

  You can manage aged updates in a number of ways such as importing them, skipping the import, or manually deploying applications. Managing aged updates gives you more flexibility in how you deploy application changes.

- Archiving and activating pipelines

If your role has the appropriate permissions, you can archive inactive pipelines so that they are not displayed on the Deployment Manager landing page.

- Disabling and enabling a pipeline

  If your role has the appropriate permissions, you can disable a pipeline on which errors continuously cause a deployment to fail. Disabling a pipeline prevents branch merging, but you can still view, edit, and stop deployments on a disabled pipeline.

- Deleting an application pipeline

  When you delete a pipeline, its associated application packages are not removed from the repositories that the pipeline is configured to use.

Logging in to Deployment Manager

Deployment Manager provides a dedicated portal from which you can access features.

To log in to Deployment Manager, on the orchestration server, enter the DMReleaseAdmin operator ID and the password that you specified for it.

Accessing the Dev Studio portal

If your role has the appropriate permission, you can access Dev Studio from within Deployment Manager. You can switch to Dev Studio to access features such as additional tools to troubleshoot issues. You can also open, modify, and create repositories and authentication profiles.

To access Dev Studio click **Operator icon Switch to Dev Studio** .

For more information on enabling a role to access Dev Studio, see Providing access to the Dev Studio portal.

- Understanding roles and users

Accessing API documentation

Deployment manager provides REST APIs for interacting with resources that in the Deployment Manager interface. Use these APIs to create and manage pipelines by using automated scripts or external information.

To access API documentation, open the Documentation/readme-for-swagger.md file in the artifact file that you downloaded from marketplace.

Understanding roles and users

Define roles and users to manage which users can access Deployment Manager and which features they can access. For example, you can create a role that does not permit users to delete pipelines for a specific application.

Deployment Manager provides two default roles, which you cannot modify or delete, that define privileges for super administrators and application administrators. Privileges for super administrators are applied across all applications, and privileges for application administrators are applied to specific applications. Super administrators can also add roles and specify the privileges to assign to them. Super administrators and application administrators can add users and assign them access to the applications that they manage.

- Using roles and privileges by creating a dynamic system setting

  You can create roles that have specific privileges and then assign users to those roles to manager Deployment Manager users. To use roles and privileges, you must first create the EnableAttributeBasedSecurity dynamic system setting.

- Adding and modifying roles

  If you are a super administrator, you can add and modify roles. Users within a role share defined responsibilities such as starting a pipeline.

- Providing access to Dev Studio to a role

  Deployment Manager provides a dedicated portal from which you can access features. From within Deployment Manager, when you configure pipeline details, you can open, modify, and create repositories and authentication profiles in Dev Studio if you have permissions to use the Dev Studio portal.

- Adding users and specifying their roles

  If you are a super administrator or application administrator, you can add users to Deployment Manager and specify their roles. Only super administrators can create other super administrators or application administrators who can access one or more applications. Application administrators can create other application administrators for the

applications that they manage.

- Modifying user roles and privileges

  Super administrators can give other users super administrative privileges or assign them as application administrators to any application. Application administrators can assign other users as application administrators for the applications that they manage.

- Modifying your user details and password

  You can modify your own user details, such as first and last name, and you can change your password.

- Deleting users

  If you are a super administrator or application administrator, you can delete users for the applications that you manage.

Using roles and privileges by creating a dynamic system setting

You can create roles that have specific privileges and then assign users to those roles to manager Deployment Manager users. To use roles and privileges, you must first create the *EnableAttributeBasedSecurity* dynamic system setting.

Do the following steps:

1. In the header of Dev Studio, click **Create SysAdmin Dynamic System Settings** .
2. In the **Short Description** field, enter a short description.
3. In the **Owning Ruleset** field, enter Pega-RulesEngine.
4. In the **Setting Purpose** field, enter EnableAttributeBasedSecurity.
5. Click **Create and open**.
6. On the **Settings** tab, in the value field, enter true.
7. Click **Save**.

Adding and modifying roles

If you are a super administrator, you can add and modify roles. Users within a role share defined responsibilities such as starting a pipeline.

If you are a super administrator, add or modify a role by doing the following steps:

1. In the navigation pane of Deployment Manager click **Users**, and then click **Roles and privileges**.
2. Do one of the following actions:
   - To add a role, click **Add role**.
   - To modify a role, click a role, and then click **Edit**.
3. In the **Add role** or **Edit role** dialog box, in the **Name** field, enter a name for the role.
4. Select the privileges that you want to assign to the role.
5. Click **Submit**.

Providing access to Dev Studio to a role

Deployment Manager provides a dedicated portal from which you can access features. From within Deployment Manager, when you configure pipeline details, you can open, modify, and create repositories and authentication profiles in Dev Studio if you have permissions to use the Dev Studio portal.

To provide access to the Dev Studio portal for a role, complete the following steps:

1. In the navigation pane of Deployment Manager, click **Users**, and then click **Roles and privileges**.
2. Do one of the following actions:
   - To add a role, click **Add role**.
   - To modify a role, click **Edit**.
3. In the **Add role** or **Edit Role** dialog box, in the **Name** field, enter a name for the role.
4. Click **Access to Dev Studio**.
5. Click **Submit**.

Result: If you specify Dev Studio as a default portal for the *PegaDeploymentManager:Administrators* access group, all the users that you add in the Deployment Manager portal can access Dev Studio.

Adding users and specifying their roles

If you are a super administrator or application administrator, you can add users to Deployment Manager and specify their roles. Only super administrators can create other super administrators or application administrators who can access one or more applications. Application administrators can create other application administrators for the applications that they manage.

To add users, do the following steps:

1. In the navigation pane of Deployment Manager click **Users**, and then click **People**.
2. On the **People** page, click **Add user**.
3. In the **Add user** dialog box, click the **User** field, and do one of the following actions:
   - Press the Down arrow key and select the user that you want to add.
   - Enter an email address.
4. Click **Add**.
5. From the **Role** list, select the role to assign to the user.
6. If you selected the App admin role or a custom role, in the **Applications** field, enter the application name that the user can access.
7. Click **Send invite** to send an email, which contains the user name and a randomly generated password for the user to log in to Deployment Manager with, to the user.

Modifying user roles and privileges

Super administrators can give other users super administrative privileges or assign them as application administrators to any application. Application administrators can assign other users as application administrators for the applications that they manage.

To modify user roles and privileges, do the following steps:

1. In the navigation pane of Deployment Manager click **Users**, and then click **People**.
2. On the **People** page, click the user.
3. In the **Roles and privileges** section, modify the user role and applications that they can access, as appropriate.
4. Click **Save**.

Modifying your user details and password

You can modify your own user details, such as first and last name, and you can change your password.

To update your information, do the following steps:

1. In the navigation pane of Deployment Manager click **Users**, and then click **People**.
2. On the **People** page, click your user name.
3. In the **Personal details** section, modify your name, email address, and phone number, as appropriate.
4. To change your password:
   a. Click **Update password.**
   b. In the **Change operator ID** dialog box, enter your new password, reenter it to confirm it, and then click **Submit**.
5. Click **Save**.

Deleting users

If you are a super administrator or application administrator, you can delete users for the applications that you manage.

To delete users, do the following steps:

1. In the navigation panel of Deployment Manager, click **Users**, and then click **People**.
2. On the **People** page, click the **Delete icon** for the user that you want to delete.

Understanding Deployment Manager notifications

You can enable notifications to receive updates about the events that occur in your pipeline. For example, you can choose to receive emails about whether unit tests failed or succeeded. You can receive notifications in the Deployment Manager notifications gadget, through email, or both. By default, all notifications are enabled for users who are configured in Deployment Manager.

By default, all notifications are enabled for users who are configured in Deployment Manager.

- Viewing and updating email accounts for notifications

  Receiving email notifications requires that an email account is configured on the orchestration server. You can view and update your email settings in Deployment Manager.

- Creating custom Deployment Manager notification channels

  You can extend Deployment Manager notification capabilities by creating custom notification channels. For example, you can send text messages to mobile devices when tasks start, stop, and are unsuccessful.

- Managing notifications

  Enable and receive notifications so that you can remain informed about important tasks in your pipeline. For example,

you can receive emails when certain tasks fail.

Viewing and updating email accounts for notifications

Receiving email notifications requires that an email account is configured on the orchestration server. You can view and update your email settings in Deployment Manager.

Changing your email settings requires access to Dev Studio, so your user role must have permission to access Dev Studio. For more information, see Understanding roles and users.

1. In the navigation pane of Deployment Manager click **Settings Email configuration** .
2. To update your email settings, perform the following steps:
   a. At the top of the **Settings: Email configuration** page, click **Dev Studio**.
   b. In the **Edit Email Account** form, configure and save the email account that you want to use to receive notifications.
   c. In the bottom left corner of Dev Studio, click **Back to Deployment Manager** to return to the Deployment Manager portal.
   d. Click the **refresh** icon to refresh your email configuration.

Creating custom Deployment Manager notification channels

You can extend Deployment Manager notification capabilities by creating custom notification channels. For example, you can send text messages to mobile devices when tasks start, stop, and are unsuccessful.

To create a custom notification channel, complete the following steps:

1. On the orchestration server, in Pega Platform, create a custom notification channel.
   For more information, see Adding a custom notification channel.
2. Add the application ruleset, which contains the channel that you created, to the Deployment Manager application.
   a. In the header of Dev Studio, click **Deployment Manager**, and then click **Definition**.
   b. On the Edit Application rule form, in the **Application rulesets** section, click **Add ruleset**.
   c. Press the Down arrow key and select the ruleset and version that contains the custom notification channel.
   d. Save the rule form.
3. Enable the channel that you created on the appropriate notifications by saving the notification in the application ruleset that contains the channel. **For example:** If you want to use the Mobile channel for the *pyStartDeployment* notification, save the *pyStartDeployment* notification in the application ruleset that contains the Mobile channel.
4. Enable the channel on the notification.
   a. Open the notification by clicking **Records Notifications** .
   b. Click the **Channels** tab.
   c. On the **Channel configurations** page, select the channel that you want to use.
   d. Save the rule form.

- Understanding custom Deployment Manager notification channels

  When notifications are enabled, you can receive notifications about the events that occur in your pipeline, such as when tasks start or stop. You can receive notifications through email, the Deployment Manager notifications gadget, or both. You can also create custom notification channels to meet application requirements such as sending notifications as phone text messages or as push notifications on mobile devices.

Understanding custom Deployment Manager notification channels

When notifications are enabled, you can receive notifications about the events that occur in your pipeline, such as when tasks start or stop. You can receive notifications through email, the Deployment Manager notifications gadget, or both. You can also create custom notification channels to meet application requirements such as sending notifications as phone text messages or as push notifications on mobile devices.

Deployment Manager provides the following notifications to which you can add channels:

- *pyAbortDeployment*
- *pyTaskFailure*
- *pyTaskFailure*
- *pyTaskCompletion*
- *pyStartDeployment*
- *pyStageCompletion*
- *pySchemaChange*
- *pyDeploymentCompletion*
- *pyAgedUpdateActionTaken*
- *pyAgedUpdateActionRequired*

Managing notifications

Enable and receive notifications so that you can remain informed about important tasks in your pipeline. For example, you

can receive emails when certain tasks fail.

To enable notifications and select the notifications that you want to receive, do the following steps:

1. In the navigation pane of Deployment Manager click your profile icon.
2. Click **Notification preferences**.
3. Select the events for which you want to receive notifications.
4. Specify how you want to receive notifications.
5. Click **Submit**.

Configuring an application pipeline

When you add a pipeline, you specify merge criteria and configure stages and steps in the continuous delivery workflow. For example, you can specify that a branch must be peer-reviewed before it can be merged, and you can specify that Pega unit tests must be run after a branch is merged and is in the QA stage of the pipeline.

You can create multiple pipelines for one version of an application. For example, you can use multiple pipelines in the following scenarios:

- To move a deployment to production separately from the rest of the pipeline. You can then create a pipeline that has only a production stage or development and production stages.
- To use parallel development and hotfix life cycles for your application.

- Adding a pipeline on Pega Cloud Services

  If you are using Pega Cloud Services, when you add a pipeline, you specify details such as the application name and version for the pipeline. Many fields are populated by default, such as the URL of your development system and product rule name and version.

- Adding a pipeline on-premises

  When you add a pipeline on premises, you define all the stages and tasks that you want to do on each system. For example, if you are using branches, you can start a build when a branch is merged. If you are using a QA system, you can run test tasks to validate application data.

- Modifying an application pipeline

  You can modify the details of your pipeline, such as configuring tasks, updating the repositories that the pipeline uses, and modifying the URLs of the systems in your environment. You cannot modify information if your pipeline is running.

Adding a pipeline on Pega Cloud Services

If you are using Pega Cloud Services, when you add a pipeline, you specify details such as the application name and version for the pipeline. Many fields are populated by default, such as the URL of your development system and product rule name and version.

To add a pipeline on Pega Cloud Services, do the following steps:

1. In the navigation pane, click **Pipelines Application pipelines** .
2. Click **New**.
3. Specify the details of the application for which you are creating the pipeline.
   a. To change the URL of your development system, which is populated by default with your development system URL, in the **Development environment** field, press the Down arrow key and select the URL.
   This is the system on which the product rule that defines the application package that moves through the repository is located.
   b. In the **Application** field, press the Down arrow key and select the name of the application.
   c. In the **Version** field, press the Down arrow key and select the application version.
   d. Click the **Access group** field and select the access group for which pipeline tasks are run.
   This access group must be present on all the candidate systems and have at least the sysadmin4 role. Ensure that the access group is correctly pointing to the application name and version that is configured in the pipeline.
   e. In the **Pipeline** name field, enter a unique name for the pipeline.
4. If you are using a separate product rule to manage test cases, to deploy a test case, in the **Application test cases** section, select the **Deploy test applications** check box; then, complete the following steps:
   a. In the **Test application** field, enter the name of the test application.
   b. In the **Version** field, enter the version of the test case product rule.
   c. In the **Access group** field, enter the access group for which test cases are run.
   d. In the **Product rule** field, enter the name of the test case product rule.
   e. From the **Deploy until** field, select the pipeline stage until the test case product rule will be deployed.
   Note: When you use separate product rules for test cases and run a pipeline, the Run Pega unit tests, Enable test coverage, and Verify test coverage tasks are run for the access group that is specified in this section.

   For the Run Pega scenario tests task, the user name that you provide should belong to the access group that is associated with the test application.

5. To change product rule that defines the contents of the application, the **Product** rule field, enter the name of the product rule that defines the contents of the application, which is populated by default with the application name.
6. To change the product rule version, in **Version** field, enter the version, which is populated by default with the application version.
7. Click **Create**. **Result:** The system adds tasks, which you cannot delete, to the pipeline that are required to successfully run a workflow, for example, Deploy and Generate Artifact. For Pega Cloud Services, it also adds mandatory tasks that must be run on the pipeline, for example, the Check guardrail compliance task and Verify security checklist task.
8. Add tasks that you want to perform on your pipeline, such as Pega unit testing.
   For more information, see Modifying stages and tasks in the pipeline.
9. Run diagnostics to verify that your pipeline is configured correctly.
   For more information, see Diagnosing a pipeline.

Adding a pipeline on premises

When you add a pipeline on premises, you define all the stages and tasks that you want to do on each system. For example, if you are using branches, you can start a build when a branch is merged. If you are using a QA system, you can run test tasks to validate application data.

To add a pipeline on premises, complete the following steps:

1. Click **Pipelines Application pipelines** .
2. Click **New**.
3. Specify the details of the application for which you are creating the pipeline.
   a. In the **Development environment** field, enter the URL of the development system.
      This is the system on which the product rule that defines the application package that moves through the repository is located.
   b. In the **Application** field, press the Down arrow key and select the name of the application.
   c. In the **Version** field, press the Down arrow key and select the application version.
   d. In the **Access group** field, press the Down arrow key and select the access group for which pipeline tasks are run.
      This access group must be present on all the candidate systems and have at least the sysadmin4 role.
   e. In the **Pipeline name** field, enter a unique name for the pipeline.
   f. In the **Product rule** field, enter the name of the product rule that defines the contents of the application.
   g. In the **Version** field, enter the product rule version.
4. If you are using a separate product rule to manage test cases, in the **Application test cases** section, to deploy a test case, select the **Deploy test applications** check box; then, complete the following steps:
   a. In the **Test application** field, enter the name of the test application.
   b. In the **Version** field, enter the version of the test case product rule.
   c. In the **Access group** field, enter the access group for which test cases are run. Ensure that the access group is correctly pointing to the application name and version that is configured in the pipeline.
   d. In the **Product rule** field, enter the name of the test case product rule.
   e. From the **Deploy until** field, select the pipeline stage until which the test case product rule will be deployed.
      When you use separate product rules for test cases and run a pipeline, the Run Pega unit tests, Enable test coverage, and Verify test coverage tasks are run for the access group that is specified in this section.

      For the Run Pega scenario tests task, the user name that you provide should belong to the access group that is associated with the test application.

5. To configure dependent applications, click **Dependencies**.
   a. Click **Add**.
   b. In the **Application name** field, press the Down arrow key and select the application name.
   c. In the **Pipeline** field, press the Down arrow key and select the pipeline.
   d. In the **Deployment** field, press the Down arrow key and select the deployment that contains the production-ready artifact of the dependent application.
      If you want the latest artifact of the dependent application to be automatically populated, select **latest**.
   For more information about dependent applications, see Managing application dependencies.
   For more information on updating existing dependencies, see Updating application dependencies in Deployment Manager.
   a. Click **Next**.
6. In the **Environment details** section, in the **Stages** section, specify the URL of each candidate system and the authentication profile that each system uses to communicate with the orchestration system.
   a. In the **Environments** field for the system, press the Down arrow key and select the URL of the system.
   b. If you are using your own authentication profiles, in the **Authentication** field for the system, press the Down arrow key and select the authentication profile that you want to communicate from the orchestration server to the system.
   By default, the fields are populated with the DMAppAdmin authentication profile.
7. In the **Artifact management** section, specify the development and production repositories through which the product rule that contains application contents moves through the pipeline.
8. In the **Development repository** field, press the Down arrow key and select the development repository.
9. In the **Production repository** field, press the Down arrow key and select the production repository.
10. In the **External orchestration server** section, if you are using a Jenkins step in a pipeline, specify the Jenkins details.

a. In the **URL** field, enter the URL of the Jenkins server.
b. In the **Authentication profile** field, press the Down arrow key and select the authentication profile on the orchestration server that specifies the Jenkins credentials to use for Jenkins jobs.

11. Click **Next**.
12. Specify whether you are using branches in your application.
    - If you are not using branches, click the **No** radio button, and then go to step 14.
    - If you are using branches, go to step 13.
13. To specify branch options, do the following steps:
    a. Click the **Yes** radio button.
    b. Do one of the following options:

    - To merge branches into the highest existing ruleset in the application, click **Highest existing ruleset**.

    - To merge branches into a new ruleset, click **New ruleset**.

    a. In the **Password** field, enter the password that locks the rulesets on the development system.
14. Click **Next**. **Result:** The system adds tasks, which you cannot delete, to the pipeline that are required to successfully run a workflow, for example, Deploy and Generate Artifact. The system also adds other tasks to enforce best practices such as Check guardrail compliance and Verify security checklist.
15. To specify that a branch must meet a compliance score before it can be merged:
    a. In the **Merge criteria** pane, click **Add task**.
    b. From the **Task** list, select **Check guardrail compliance**.
    c. In the **Weighted compliance score** field, enter the minimum required compliance score.
    d. Click **Submit**.

    For more information about compliance scores, see Compliance score logic
16. To specify that a branch must be reviewed:
    a. In the **Merge criteria** pane, click **Add task**.
    b. From the **Task** list, select **Check review status**.
    c. Click **Submit**.

    For more information about branch reviews, see Branch reviews.
17. To run Pega unit tests on the branches for the pipeline application or for an application that is associated with an access group before it can be merged:
    a. In the **Merge criteria** pane, click **Add task**.
    b. From the **Task** list, select **Pega unit testing**.
    c. To run all the Pega unit tests for an application that is associated with an access group, in the **Access Group** field, enter the access group.
    d. Click **Submit**.
    For more information about creating Pega unit tests, see Creating Pega unit test cases.

    Result:

    When you use separate product rules for test cases and run a pipeline, the Run Pega unit tests, Enable test coverage, and Verify test coverage tasks are run for the access group that is specified in the Application test cases section.

    For the Run Pega scenario tests task, the user name that you provide should belong to the access group that is associated with the test application.

18. To start a deployment automatically when a branch is merged, click the **Trigger deployment on merge check box**. Do not select this check box if you want to manually start deployments.

    For more information, see Manually starting a deployment in Deployment Manager.
19. Clear a check box for a deployment life cycle stage to skip it.
20. In the **Continuous Deployment** section, specify the tasks to be performed during each stage of the pipeline. See the following topics for more information:
    - Running Pega unit tests by adding the Run Pega unit tests task
    - Running Jenkins steps by adding the Run Jenkins step task
    - Specifying that an application meet a compliance score by adding the Check guardrail compliance score task
    - Ensuring that the Application Security Checklist is completed by adding the Verify security checklist task
    - Starting test coverage by adding the Enable test coverage task
    - Stopping test coverage by adding the Validate test coverage task
    - Running Pega scenario tests by adding the Run Pega scenario tests task
    - Refreshing application quality by adding the Refresh application quality task
    - Modifying the Approve for production task
21. Clear the **Production ready** check box if you do not want to generate an application package, which is sent to the production repository.
    You cannot clear this check box if you are using a production stage in the life cycle.
22. Click **Finish**.
23. Run diagnostics to verify that your pipeline is configured correctly.
    For more information, see Diagnosing a pipeline.

- Adding the Pega unit testing task

  If you use Pega unit tests to validate application data, add the Pega unit testing task on the pipeline stage where you want to run it. For example, you can run Pega unit tests on a QA system.

- Adding the Jenkins task

  If you are using Jenkins to perform tasks in your pipeline, you can add the Jenkins task to the stage on which you want it to run.

- Adding the manual step task

  Use manual steps so that users must take an action before a pipeline deployment can continue. Users can either accept the task to continue the deployment or reject the task to stop it.

- Adding the Check guardrail compliance score task

  You can use the Check guardrail compliance score task so that an application must meet a compliance score for the deployment to continue. The default value is 95, which you can modify.

- Starting another pipeline by adding the Trigger deployment task

  You can start another pipeline by adding the Trigger deployment task to a stage in your current pipeline. By starting another pipeline from a current pipeline, you can add more stages to your pipeline.

- Adding the Verify security checklist task

  For your pipeline to comply with security best practices, you can add a task so that to ensure that all the steps in Application Security Checklist are performed.

- Starting test coverage by adding the Enable test coverage task

  Add the Enable test coverage task to start test coverage. Starting and stopping test coverage generates a report that identifies the executable rules in your application that are either covered or not covered by tests. As a best practice, to ensure application quality, you should test all the rules in your application for which testing is supported.

- Stopping test coverage by adding the Validate test coverage task

  Add this task to stop a test coverage session. Starting and stopping test coverage generates a report that identifies the executable rules in your application that are either covered or not covered by tests. As a best practice, to ensure application quality, you should test all the rules in your application for which testing is supported.

- Running Pega scenario tests by adding the Run Pega scenario tests task

  If you are using Pega scenario tasks, you can run them in your pipeline by using the Run Pega scenario tests task. Deployment Manager supports Selenium 3.141.59.

- Refreshing application quality by adding the Refresh application quality task

  To refresh the Application Quality dashboard, which provides information about the health of your application, on the candidate system, add the Refresh application quality task. You can refresh the dashboard after running Pega unit tests, checking guardrail compliance, running Pega scenario tests, and starting or stopping test coverage.

- Modifying the Approve for production task

  The Approve for production task is added to the stage before production. Use this task if you want a user to approve application changes before those changes are send to production.

Running Pega unit tests by adding the Run Pega unit tests task

If you use Pega unit tests to validate application data, add the Pega unit testing task on the pipeline stage where you want to run it. For example, you can run Pega unit tests on a QA system.

When you use separate product rules for test cases and run a pipeline, the Pega unit testing task is run for the access group that is specified in the **Application test cases** section, which you configure when you add or modify a pipeline.

To run Pega unit tests for either the pipeline application or for an application that is associated with an access group, do the following steps:

1. Do one of the following actions:
   - Click a task, click the **More** icon, and then click either **Add task above** or **Add task below**.
   - Click **Add task** in the stage.
2. In the task list, click **Pega unit testing**.
3. Do one of the following actions:

   - To run all the Pega unit tests that are in a Pega unit suite for the pipeline application, in the **Test Suite ID** field, enter the pxInsName of the test suite.

     You can find this value in the XML document that comprises the test suite by clicking, in Dev Studio, **Actions** **XML** on the **Edit Test Suite** form. If you do not specify a test suite, all the Pega unit tests for the pipeline application are run.

- To run all the Pega unit tests for an application that is associated with an access group, in the **Access Group** field, enter the access group.

  For more information about creating Pega unit tests, see Creating Pega unit test cases.

4. Click **Submit**.
5. Continue configuring your pipeline. For more information, see one of the following topics:
   - Adding a pipeline on premises
   - Modifying stages and tasks in the pipeline

Running Jenkins steps by adding the Run Jenkins step task

If you are using Jenkins to perform tasks in your pipeline, you can add the Run Jenkins step to the stage on which you want it to run. If you have configured the Jenkins *OrchestratorURL* and *PipelineID* parameters, when this task fails, the pipeline stops running. For more information about configuring these parameters, see Configuring Jenkins.

To add this task, do the following steps:

1. Do one of the following actions:
   - Click a task, click the **More** icon, and then click either **Add task above** or **Add task below**.
   - Click **Add task** in the stage.
2. In the **Task** list, click **Run Jenkins step**.
3. In the **Job name** field, enter the name of the Jenkins job (which is the name of the Jenkins deployment) that you want to run.
4. In the **Token** field, enter the Jenkins authentication token.
5. In the **Parameters** field, enter parameters, if any, to send to the Jenkins job.
6. Click **Submit**.
7. Continue configuring your pipeline. For more information, see one of the following topics:
   - Adding a pipeline on premises
   - Modifying stages and tasks in the pipeline

Continuing or stopping a deployment by adding the Perform manual step task

Use manual steps so that users must take an action before a pipeline deployment can continue. Users can either accept the task to continue the deployment or reject the task to stop it.

To add a manual step that a user must perform in the pipeline, do the following steps:

1. Do one of the following actions:
   - Click a task, click the **More** icon, and then click either **Add task above** or **Add task below**.
   - Click **Add task** in the stage.
2. In the **Task** list, click **Perform manual step**.
3. In the **Job name** field, enter text that describes the action that you want the user to take.
4. In the **Assigned to** field, press the Down arrow key and select the operator ID to assign the task to.
5. Click **Submit**.
6. Continue configuring your pipeline. For more information, see one of the following topics:
   - Adding a pipeline on premises
   - Modifying stages and tasks in the pipeline

Specifying that an application meet a compliance score by adding the Check guardrail compliance score task

You can use the Check guardrail compliance score task so that an application must meet a compliance score for the deployment to continue. The default value is 97, which you can modify.

To specify that an application must meet a compliance score, do the following steps:

1. Do one of the following actions:
   - Click a task, click the **More** icon, and then click either **Add task above** or **Add task below**.
   - Click **Add task** in the stage.
2. In the **Task** list, click **Check guardrail compliance**.
3. In the **Weighted compliance score** field, enter the minimum required compliance score.
4. Continue configuring your pipeline. For more information, see one of the following topics:
   - Adding a pipeline on premises
   - Modifying stages and tasks in the pipeline

Starting another pipeline by adding the Trigger deployment task

You can start another pipeline by adding the Trigger deployment task to a stage in your current pipeline. By starting another pipeline from a current pipeline, you can add more stages to your pipeline.

To add the Trigger deployment task to a stage in your pipeline, perform the following steps:

1. Do one of the following actions:
    - Click a task, click the **More** icon, and then click either **Add task above** or **Add task below**.
    - Click **Add task** in the stage.
2. In the **Task** list, click **Trigger deployment**.
3. In the **Application name** field, press the Down arrow key, and then select the application that you want to deploy.
4. In the **Pipeline name** field, press the Down arrow key and, then select the pipeline that you want to start.
5. If you want to deploy the artifact that you are deploying in the current pipeline, select the **Deploy current artifact** check box. Otherwise, a new application is deployed on the pipeline.
6. Click **Submit**.

Ensuring that the Application Security Checklist is completed by adding the Verify security checklist task

For your pipeline to comply with security best practices, you can add a task to ensure that all the steps in the Application Security Checklist are performed. For customers on Pega Platform 8.4 and above, a new Security Checklist API is available to provide an automated security configuration assessment. Both candidate and orchestrator environments should be on or above Deployment Manager 4.8 to utilize this functionality.

Before you begin: You must log in to the system for which this task is configured, and then mark all the tasks in the Application Security checklist as completed for the pipeline application. For more information about completing the checklist, see Preparing your application for secure deployment. To add the Verify security checklist task, do the following steps:

1. Do one of the following actions:
    - Click a task, click the **More** icon, and then click either **Add task above** or **Add task below**.
    - Click **Add task** in the stage.
2. In the **Task** list, click **Verify Security checklist**.
3. Click **Submit**.
4. Continue configuring your pipeline. For more information, see one of the following topics:
    - Adding a pipeline on premises
    - Modifying stages and tasks in the pipeline

Starting test coverage by adding the Enable test coverage task

Add the Enable test coverage task to start test coverage. Starting and stopping test coverage generates a report that identifies the executable rules in your application that are either covered or not covered by tests. As a best practice, to ensure application quality, you should test all the rules in your application for which testing is supported.

For more information about application-level coverage reports, see Generating an application-level test coverage report.

When you use separate product rules for test cases and run a pipeline, the Enable test coverage task is run for the access group that is specified in the Application test cases section, which you configure when you add or modify a pipeline.

To add this task, complete the following steps:

1. Do one of the following actions:
    - Click a task, click the **More** icon, and then click either **Add task above** or **Add task below**.
    - Click **Add task** in the stage.
2. In the **Task** list, click **Enable test coverage**.
3. Select the **Start a new session** check box to start a test coverage session every time that the pipeline runs the deployment. If you do not select this check box, if a test coverage session is already running, the pipeline pauses and returns an error.
4. Click **Submit**.
5. Continue configuring your pipeline. For more information, see one of the following topics:
    - Adding a pipeline on premises
    - Modifying stages and tasks in the pipeline

- Adding a pipeline on premises
- Adding a pipeline on Pega Cloud Services
- Modifying application details

Stopping test coverage by adding the Validate test coverage task

Add this task to stop a test coverage session. Starting and stopping test coverage generates a report that identifies the executable rules in your application that are either covered or not covered by tests. As a best practice, to ensure application quality, you should test all the rules in your application for which testing is supported.

For more information about application-level coverage reports, see Generating an application-level test coverage report.

When you use separate product rules for test cases and run a pipeline, the Validate test coverage task is run for the access

group that is specified in the Application test cases section, which you configure when you add or modify a pipeline.

1. Add this task below the Enable test coverage task by doing one of the following actions:
   - Click a task, click the **More** icon, and then click either **Add task above** or **Add task below**.
   - Click **Add task** in the stage.
2. In the **Task** list, click **Validate test coverage**.
3. Click **Submit**.
4. Continue configuring your pipeline. For more information, see one of the following topics:
   - Adding a pipeline on premises
   - Modifying stages and tasks in the pipeline

Running Pega scenario tests by adding the Run Pega scenario tests task

If you are using Pega scenario tasks, you can run them in your pipeline by using the Run Pega scenario tests task. Deployment Manager supports Selenium 3.141.59.

To add the Run Pega scenario tests task, do the following steps:

1. Do one of the following actions:
   - Click a task, click the **More** icon, and then click either **Add task above** or **Add task below**.
   - Click **Add task** in the stage.
2. In the **Task** list, click **Run Pega scenario tests**.
3. In the **User name** field, enter the user name for the Pega Platform instance on which you are running scenario tests. **Note:** For the Run Pega scenario tests task, if you are using a separate product rule for a test application, the user name that you provide should belong to the access group that is associated with the test application.
4. In the **Password** field, enter the Pega Platform password.
5. From the **Test Service Provider** field, select the browser that you are using to run the scenario tests in the pipeline.
6. Do one of the following actions:
   - If you selected CrossBrowserTesting, BrowserStack, or SauceLabs, go to step 7.
   - If you selected Standalone, go to step 8.
7. If you selected CrossBrowserTesting, BrowserStack, or SauceLabs:
   a. In the **Provider auth name** field, enter the auth name that you use to log in to the test service provider.
   b. In the **Provider auth** key field, enter the key for the test service provider.
   c. Go to step 9.
8. If you selected Standalone, in the Provider URL field, enter the URL of the Selenium Standalone Server by using one of the following:
   a. Hub hostname and port: Use the format Hubhostname:port.
   b. IP address: Enclose the IP address in double quotation marks.
9. In the **Browser** field, enter the browser that you are using to record scenario tests.
10. In the **Browserversion** field, enter the browser version.
11. In the **Platform** field, enter the development platform that you are using to record tests.
12. In the **Screen resolution** field, enter the resolution at which are recording scenario tests.
13. Click **Submit**.
14. Continue configuring your pipeline. For more information, see one of the following topics:
    - Adding a pipeline on premises
    - Modifying stages and tasks in the pipeline

Refreshing application quality by adding the Refresh application quality task

To refresh the Application Quality dashboard, which provides information about the health of your application, on the candidate system, add the Refresh application quality task. You can refresh the dashboard after running Pega unit tests, checking guardrail compliance, running Pega scenario tests, and starting or stopping test coverage.

To add this task, complete the following steps:

1. Do one of the following actions:
   - Click a task, click the **More** icon, and then click either **Add task above** or **Add task below**.
   - Click **Add task** in the stage.
2. In the **Task** list, click **Refresh application** quality.
3. Click **Submit**.
4. Continue configuring your pipeline. For more information, see one of the following topics:
   - Adding a pipeline on premises
   - Modifying stages and tasks in the pipeline

Modifying the Approve for production task

The Approve for production task is added to the stage before production. Use this task if you want a user to approve application changes before those changes are send to production.

To modify the Approve for production task, do the following steps:

1. Click the **Info** icon.
2. In the **Job name** field, enter a name for the task.
3. In the **Assign to** field, press the Down arrow key and select the user who approves the application for production. An email is sent to this user, who can approve or reject application changes from within the email.
4. Click **Submit**.
5. Continue configuring your pipeline. For more information, see one of the following topics:
    - Adding a pipeline on premises
    - Modifying stages and tasks in the pipeline

Modifying an application pipeline

You can modify the details of your pipeline, such as configuring tasks, updating the repositories that the pipeline uses, and modifying the URLs of the systems in your environment. You cannot modify information if your pipeline is running.

- Modifying application details

  You can modify application details, such as the product rule that defines the content of the application that moves through the pipeline.

- Modifying URLs and authentication profiles

  You can modify the URLs of your development and candidate systems and the authentication profiles that are used to communicate between those systems and the orchestration server.

- Modifying repositories

  You can modify the development and production repositories through which the product rule that contains application contents moves through the pipeline. All the generated artifacts are archived in the development repository, and all the production-ready artifacts are archived in the production repository.

- Configuring Jenkins server information for running Jenkins jobs

  If you are using a Run Jenkins step, configure Jenkins server information so that you can run Jenkins jobs.

- Modifying merge options for branches

  If you are using branches in your application, specify options for merging branches into the base application.

- Modifying stages and tasks in the pipeline

  You can modify the stages and the tasks that are performed in each stage of the pipeline. For example, you can skip a stage or add tasks such as Pega unit testing to be done on the QA stage.

Modifying application details

You can modify application details, such as the product rule that defines the content of the application that moves through the pipeline.

Do the following steps:

1. If the pipeline is not open, in the navigation pane of Deployment Manager, click **Pipelines** and then click the name of the pipeline.
2. Click **Actions Pipeline settings** .
3. Click **Application details**.
4. In the **Development environment** field, enter the URL of the development system, which is the system on which the product rule that defines the application package that moves through the repository is located.
5. In the **Version** field, press the Down arrow key and select the application version.
6. In the **Product rule** field, enter the product rule that defines the contents of the application.
7. In the **Version** field, enter the product rule version.
8. If you are using a separate product rule to manage test cases, in the **Application test cases** section, complete the following steps:
    a. To deploy test cases, select the **Deploy test applications** check box.
    b. In the Test application field, enter the name of the test application.
    c. In the **Version** field, enter the version of the test case product rule.
    d. In the **Access group** field, enter the access group for which test cases are run.
    e. In the **Product rule** field, enter the name of the test case product rule.
    f. From the **Deploy until** field, select the pipeline stage until which the test case product rule will be deployed.
    Note: When you use separate product rules for test cases and run a pipeline, the Run Pega unit tests, Enable test coverage, and Verify test coverage tasks are run for the access group that is specified in this section.

    For the Run Pega scenario tests task, the user name that you provide should belong to the access group that is associated with the test application.
9. If the application depends on other applications, in the **Dependencies** section, add those applications.

a. Click **Add**.
b. In the **Application name** field, press the Down arrow key and select the application name.
c. In the **Application version** field, press the Down arrow key and select the application version.
d. In the **Repository name** field, press the Down arrow key and select the repository that contains the production-ready artifact of the dependent application.
If you want the latest artifact of the dependent application to be automatically populated, ensure that the repository that contains the production-ready artifact of the dependent application is configured to support file updates.
e. In the **Artifact name** field, press the Down arrow key and select the artifact.
For more information about dependent applications, see Listing product dependencies.
10. Click **Save**.
11. Run diagnostics to verify that your pipeline is configured correctly.
For more information, see Diagnosing a pipeline.

Modifying URLs and authentication profiles

You can modify the URLs of your development and candidate systems and the authentication profiles that are used to communicate between those systems and the orchestration server.

Do the following steps:

1. If the pipeline is not open, in the navigation pane of Deployment Manager, click **Pipelines** and then click the name of the pipeline.
2. Click **Actions Pipeline settings** .
3. Click **Deployment stages**.
4. In the **Environments** field for the system, press the Down arrow key and select the URL of the system.
   - VPN and firewalls that add the orchestrator to a list of trusted IP ranges must support two-way communication.
   - Environments with URL redirects are not supported, and the URL used for the environment should not have redirection enabled.
5. In the **Authentication** field for the system, press the Down arrow key and select the authentication profile that you want to communicate from the orchestration server to the system.
6. Click **Save**.
7. Run diagnostics to verify that your pipeline is configured correctly.
For more information, see Diagnosing a pipeline.

Modifying repositories

You can modify the development and production repositories through which the product rule that contains application contents moves through the pipeline. All the generated artifacts are archived in the development repository, and all the production-ready artifacts are archived in the production repository.

You do not need to configure repositories if you are using Pega Cloud Services; you can use different repositories other than the default ones that are provided.

1. If the pipeline is not open, in the navigation pane of Deployment Manager, click **Pipelines** and then click the name of the pipeline.
2. Click **Actions Pipeline settings** .
3. Click **Artifact Management**.
4. If you are using Deployment Manager on premises, or on Pega Cloud Services with default repositories, complete the following tasks:
   a. In the **Application repository** section, in the **Development repository** field, press the Down arrow key and select the development repository
   b. In the **Production repository** field, press the Down arrow key and select the production repository.
5. If you are using Deployment Manager on Pega Cloud Services and want to use different repositories other than the default repositories, complete the following tasks:
   a. In the **Artifact repository** section, click **Yes**.
   b. In the **Development repository** field, press the Down arrow key and select the development repository.
   c. In the **Production repository** field, press the Down arrow key and select the production repository.
6. Click **Save**.
7. Run diagnostics to verify that your pipeline is configured correctly.
For more information, see Diagnosing a pipeline.

Configuring Jenkins server information for running Jenkins jobs

If you are using a Run Jenkins step, configure Jenkins server information so that you can run Jenkins jobs.

1. If the pipeline is not open, in the navigation pane of Deployment Manager, click **Pipelines** and then click the name of the pipeline.
2. Click **Actions Pipeline settings** .
3. Click **External orchestration server**.
4. In the **URL** field, enter the URL of the Jenkins server.
5. In the **Authentication** profile field, press the Down arrow key and select the authentication profile on the

orchestration server that specifies the Jenkins credentials to use for Jenkins jobs.
6. Run diagnostics to verify that your pipeline is configured correctly.
For more information, see Diagnosing a pipeline.

Modifying merge options for branches

If you are using branches in your application, specify options for merging branches into the base application.

Do the following steps:

1. If the pipeline is not open, in the navigation pane of Deployment Manager, click **Pipelines** and then click the name of the pipeline.
2. Click **Actions Pipeline settings** .
3. Click **Merge policy**.
4. If you are not using branches, click the **No** radio button, and then go to step 6.
5. If you are using branches, do the following actions:
    a. Click **Yes**.
    b. Do one of the following actions:
    - To merge branches into the highest existing ruleset in the application, click **Highest existing ruleset**.
    - To merge branches into a new ruleset, click **New ruleset**.
    a. In the **Password** field, enter the password that locks the rulesets on the development system.
6. Click **Save**.
7. Run diagnostics to verify that your pipeline is configured correctly.
For more information, see Diagnosing a pipeline.

Modifying stages and tasks in the pipeline

You can modify the stages and the tasks that are performed in each stage of the pipeline. For example, you can skip a stage or add tasks such as Pega unit testing to be done on the QA stage.

Do the following steps:

1. If the pipeline is not open, in the navigation pane of Deployment Manager, click **Pipelines** and then click the name of the pipeline.
2. Click **Pipeline model**.
3. To specify that a branch must meet a compliance score before it can be merged:
    a. In the **Merge criteria** pane, click **Add task**.
    b. From the **Task** list, select **Check guardrail compliance**.
    c. In the **Weighted compliance score** field, enter the minimum required compliance score.
    d. Click **Submit**.
For more information about compliance scores, see Compliance score logic.
4. To specify that a branch must be reviewed before it can be merged:
    a. In the **Merge criteria** pane, click **Add task**.
    b. From the **Task** list, select **Check review status**.
    c. Click **Submit**.
For more information about branch reviews, see Branch reviews.
5. To run Pega unit tests on the branches for the pipeline application or for an application that is associated with an access group before it can be merged:
    a. In the **Merge criteria** pane, click **Add task**.
    b. From the **Task** list, select **Pega unit testing**.
    c. To run all the Pega unit tests for an application that is associated with an access group, in the **Access Group** field, enter the access group.
    d. Click **Submit**.
For more information about creating Pega unit tests, see Creating Pega unit test cases
6. To start a deployment automatically when a branch is merged, select the **Trigger deployment on merge** check box. Do not select this check box if you want to manually start a deployment.
For more information, see Manually starting a deployment in Deployment Manager.
7. Clear a check box for a deployment life cycle stage to skip it.
8. In the **Continuous Deployment** section, specify the tasks to be performed during each stage of the pipeline. See the following topics for more information:
    - Running Pega unit tests by adding the Run Pega unit tests task
    - Running Jenkins steps by adding the Run Jenkins step task
    - Specifying that an application meet a compliance score by adding the Check guardrail compliance score task
    - Ensuring that the Application Security Checklist is completed by adding the Verify security checklist task
    - Starting test coverage by adding the Enable test coverage task
    - Stopping test coverage by adding the Validate test coverage task
    - Running Pega scenario tests by adding the Run Pega scenario tests task
    - Refreshing application quality by adding the Refresh application quality task
    - Modifying the Approve for production task
9. Clear the **Production ready** check box if you do not want to generate an application package, which is sent to the production repository. You cannot clear this check box if you are using a production stage in the life cycle.
10. Click **Finish**.

11. Run diagnostics to verify that your pipeline is configured correctly.
    For more information, see Diagnosing a pipeline.

Accessing systems in your pipeline

You can open the systems in your pipeline and log in to the Pega Platform instances on each system. For example, you can access the system on which the QA stage is installed.

1. If the pipeline is not open, in the navigation pane of Deployment Manager, click **Pipelines** and then click the name of the pipeline.
2. Click the pop-out arrow for the system that you want to open.

Filtering pipelines in the dashboard

You can filter the pipelines that the dashboard displays by application name, version, and pipeline deployment status. By filtering pipelines, the dashboard displays only the information that is relevant to you.

To filter the display of pipelines, perform the following steps:

1. In the navigation pane of Deployment Manager click **Pipelines Application pipelines** .
2. At the top of the dashboard, in the **View** lists, select the information with which you want to filter the display of pipelines, and then click **Apply**.

Viewing merge requests

You can view the status of the merge requests for a pipeline to gain more visibility into the status of your pipeline. For example, you can see whether a branch was merged in a deployment and when it was merged.

To view merge requests, do the following steps:

1. If the pipeline is not open, in the navigation pane of Deployment Manager, click **Pipelines** and then click the name of the pipeline.
2. In the **Development** stage, click ***X*** **Merges in queue** to view all the branches that are in the queue or for which merge is in progress.
3. In the **Merge requests ready for deployment** dialog box, click **View all merge requests** to view all the branches that are merged into the pipeline.

Viewing deployment reports for a specific deployment

Deployment reports provide information about a specific deployment. You can view information such as the number of tasks that you configured on a deployment that have been completed and when each task started and ended. If there were schema changes on the deployment, the report displays the schema changes.

Do the following steps:

1. If the pipeline is not open, in the navigation pane of Deployment Manager, click **Pipelines** and then click the name of the pipeline.
2. Perform one of the following actions:
    - To view the report for the current deployment, click the **More** icon, and then click **View report**.
    - To view the report for a previous deployment, expand the **Deployment History** pane and click **Reports** for the appropriate deployment.

- Viewing reports for all deployments

  Reports provide a variety of information about all the deployments in your pipeline. For example, you can view the frequency of new deployments to production.

- Understanding schema changes in application packages

Viewing reports for all deployments

Reports provide a variety of information about all the deployments in your pipeline. For example, you can view the frequency of new deployments to production.

You can view the following key performance indicators (KPI):

- Deployment Success – Percentage of deployments that are successfully deployed to production
- Deployment Frequency – Frequency of new deployments to production
- Deployment Speed – Average time taken to deploy to production
- Start frequency – Frequency at which new deployments are triggered
- Failure rate – Average number of failures per deployment
- Merges per day – Average number of branches that are successfully merged per day

Do the following steps:

1. Open the pipeline by doing one of the following actions:
   - If the pipeline open, click **Actions View report** .
   - If a pipeline is not open, in the navigation pane, click **Reports**. Next, in the **Pipeline** field, press the Down arrow key and select the name of the pipeline for which to view the report.
2. From the list that appears in the top right of the **Reports** page, select whether you want to view reports for all deployments, the last 20 deployments, or the last 50 deployments.

Starting deployments

You can start deployments in a number of ways. For example, you can start a deployment manually if you are not using branches, by submitting a branch into the Merge Branches wizard, or by publishing application changes in App Studio to create a patch version of your application. Your user role determines if you can start a deployment.

- Manually starting a deployment

  You can start a deployment manually if you are not using branches and are working directly in rulesets. You can also start a deployment manually if you do not want deployments to start automatically when branches are merged.

- Understanding application changes made in App Studio

  You can publish application changes that you make in App Studio to the pipeline. Publishing your changes creates a patch version of the application and starts a deployment. For example, you can change a life cycle, data model, or user interface elements in a screen and submit those changes to systems in the pipeline.

- Starting a deployment as you merge branches from the development environment

  In either a branch-based or distributed, branch-based environment, you can immediately start a deployment by submitting a branch into a pipeline in the Merge Branches wizard. The wizard displays the merge status of branches so that you do not need to open Deployment Manager to view it.

- Understanding roles and users

Manually starting a deployment in Deployment Manager

You can start a deployment manually if you are not using branches and are working directly in rulesets. You can also start a deployment manually if you do not want deployments to start automatically when branches are merged.

To start a deployment manually, do the following steps:

1. If you do not want deployments to start automatically when branches are merged:
   a. If the pipeline is not open, in the navigation pane, click **Pipelines Application pipelines** .
   b. Click **Pipeline model**.
   c. Select the **Trigger deployment on merge** check box.
2. In the navigation pane, click **Pipelines Application pipelines** , and then click **Start deployment** for the pipeline that you want to start.
3. In the **Start deployment** dialog box, start a new deployment or deploy an existing application package. If you are deploying an existing package, you can only select a production-ready deployment; development artifacts are not supported.

Choice	Action
Start a deployment and deploy a new application package	a. Click **Generate new artifact**. b. In the **Deployment name** field, enter the name of the deployment. c. Click **Submit**.
Start a deployment and deploy an application package that has been packaged in a previous production deployment	a. Click **Deploy an existing artifact**. b. In the **Deployment name** field, enter the name of the deployment. c. In the **Pipeline** field, press the Down arrow key and select the pipeline. d. In the **Deployment** field, press the Down arrow key and select the previous deployment of that pipeline. e. Click **Deploy an existing artifact**. f. In the **Deployment name** field, enter the name of the deployment. g. In the **Pipeline** field, press the Down arrow key and select the pipeline. h. In the **Deployment** field, press the Down arrow key and select the previous deployment of that pipeline.

Understanding application changes made in App Studio

You can publish application changes that you make in App Studio to the pipeline. Publishing your changes creates a patch version of the application and starts a deployment. For example, you can change a life cycle, data model, or user interface elements in a screen and submit those changes to systems in the pipeline.

Ensure the following items are properly configured before making application changes in App Studio.

- A product rule exists with the same name and version as the application being deployed. For more information see Creating a product rule by using the create menu
- A pipeline has been created in Deployment Manager for the application being deployed.
- There is at least one unlocked ruleset in the application.
- The users who will publish changes are logged into the application being deployed on the development system.
- The users who will publish changes have been granted a role in Deployment Manager that can start deployments.
- The RMURL dynamic system setting, which defines the URL of orchestration server, is configured on the system.

Your pipeline should have at least one quality assurance or staging stage with a manual task so that you do not deploy changes to production that have not been approved by stakeholders.

- Publishing application changes in App Studio

Publishing application changes in App Studio

When you publish an application to a stage, your rules are deployed immediately to that system. To allow stakeholders to inspect and verify changes before they are deployed the stage, configure a manual task in on the previous stage. When the pipeline runs, it is paused during a manual step that is assigned to a user, which allows stakeholders to review your changes before they approve the step and resume running the pipeline.

1. In App Studio, do one of the following actions:
   - Click **Turn editing on**, and then, in the navigation pane, click **Settings Versions** .
   - In the App Studio header, click **Publish**.

   The **Settings** page displays the stages that are enabled in the application pipeline in Deployment Manager. The available stages are, in order, quality assurance, staging, and production.
   It also displays the application versions that are on each system. The version numbers are taken from the number at the end of each application deployment name in Deployment Manager. For example, if a deployment has a name of "MyNewApp:01_01_75", the dialog box displays "v75".
2. Submit an application from development to quality assurance or staging in your pipeline by completing the following steps:
   a. Click either **Publish to QA** or **Publish to staging**.
   b. To add a comment, which will be published when you submit the application, add a comment in the **Publish confirmation** dialog box.
   c. If Agile Workbench has been configured, associate a bug or user story with the application, in the **Associated User stories/Bugs** field, press the Down arrow key and select the bug or user story.
   d. Click **OK**. **Result:** Each unlocked ruleset version in your application is locked and rolled to the next highest version and is packaged and imported into the system. The amount of time that publishing application changes takes depends on the size of your application.

      A new application is also copied from the application that is defined on the pipeline in Deployment Manager. The application patch version is updated to reflect the version of the new rulesets; for example, if the ruleset versions of the patch application are 01-01-15, the application version is updated to be 01.01.15. A new product rule is also created.

      In addition, this application is locked and cannot be unlocked. You can use this application to test specific patch versions of your application on quality assurance or staging systems. You can also use it to roll back a deployment.
3. Make changes to your application in the unlocked rulesets, which you can publish again into the pipeline. If an application is already on the system, it is overridden by the new version that you publish.
4. If you configured a manual step, request that stakeholders review and test your changes. After they communicate to you that they have completed testing, you can publish your changes to the next stage in the pipeline.
5. Publish the application to the next stage in the pipeline by clicking the link that is displayed.
   The name of the link is the **Job name** field of the manual task that is defined on the stage.
   Result: If you do not have a manual task defined, the application automatically moves to the next stage.

Starting a deployment as you merge branches from the development environment

In either a branch-based or distributed, branch-based environment, you can immediately start a deployment by submitting a branch into a pipeline in the Merge Branches wizard. The wizard displays the merge status of branches so that you do not need to open Deployment Manager to view it.

If you are using a separate product rule for a test application, after you start a deployment either by using the Merge Branches wizard, the branches of both the target and test applications are merged in the pipeline.

You can submit a branch to your application and start the continuous integration portion of the pipeline when the following criteria is met:

- You have created a pipeline for your application in Deployment Manager.
- You are merging a single branch.
- The RMURL dynamic system setting, which defines the URL of orchestration server, is configured on the system.
- All the rulesets in your branch belong to a single application that is associated with your pipeline. Therefore, your branch cannot contain rulesets that belong to different application layers.

- Configuring settings before using the Merge Branches wizard

  You can start a branch merge, which triggers a deployment, by using the Merge Branches wizard. You must configure certain settings before you can submit a branch to your application.

- Submitting a branch into an application by using the Merge Branches wizard

  You can start a branch merge, which triggers a deployment, by submitting a branch into an application in the Merge Branches wizard. By using the wizard to start merges, you can start a deployment without additional configuration.

Configuring settings before using the Merge Branches wizard

You can start a branch merge, which triggers a deployment, by using the Merge Branches wizard. You must configure certain settings before you can submit a branch to your application.

Before you begin: Before starting a branch merge, do the following tasks.

1. Check all rules into their base rulesets before you merge them.
2. Check if there are any potential conflicts to address before merging branches. For more information, see Viewing branch quality and branch contents.
3. As a best practice, lock a branch after development is complete so that no more changes can be made. For more information, see Locking a branch.

Submitting a branch into an application by using the Merge Branches wizard

You can start a branch merge, which triggers a deployment, by submitting a branch into an application in the Merge Branches wizard. By using the wizard to start merges, you can start a deployment without additional configuration.

To submit a branch into an application by using the Merge Branches wizard, perform the following steps:

1. In the navigation pane of Dev Studio,, click **App**, and then click **Branches**.
2. Right-click the branch and click **Merge**.
3. Click **Proceed**. **Result:** The wizard displays a message in the following scenarios:
   - If there are no pipelines that are configured for your application or there are no branches in the target application.
   - If the value for the RMURL dynamic system setting is not valid.
4. Click **Switch to standard merge** to switch to the Merge Branches wizard that you can use to merge branches into target rulesets. For more information, see Merging branches into target rulesets .
5. In the **Application pipelines** section, from the **Pipeline** list, select the application for which the pipeline is configured into which you want to merge branches.
6. In the **Merge Description** field, enter information that you want to capture about the merge.

   This information appears when you view deployment details.

7. In the **Associated User stories/bugs** field, press the Down arrow key, and then select the Agile Workbench user story or bug that you want to associate with this branch merge.
8. Click **Merge**.

Result:

The system queues the branch for merging, generates a case ID for the merge, and runs the continuous integration criteria that you specified.

If there are errors, and the merge is not successful, an email is sent to the operator ID of the release manager that is specified on the orchestration server.

The branch is stored in the development repository and, after the merge is completed, Deployment Manager deletes the branch from the development system. By storing branches in the development repository, Deployment Manager keeps a history, which you can view, of the branches in a centralized location.

If your development system is appropriately configured, you can rebase your development application to obtain the most recently committed rulesets after you merge your branches. For more information, see Rebasing rules to obtain latest versions.

Pausing and resuming deployments

When you pause a deployment, the pipeline completes the task that it is running, and stops the deployment at the next step. Your user role determines if you can pause a deployment.

To pause a deployment:

1. If the pipeline is not open, in the navigation pane of Deployment Manager, click **Pipelines** and then click the name of the pipeline.
2. Click the pipeline.
3. Click **Pause**.
4. Click the **Pause** button again to resume the deployment.

- Understanding roles and users

Stopping a deployment

If a user has privileges to abort, they can do so to prevent a deployment from moving through a pipeline.

1. If the pipeline is not open, in the navigation pane of Deployment Manager, click **Pipelines** and then click the name of the pipeline.
2. Click the **More** icon on the stage where the deployment is active, and then click **Abort**.

- Understanding roles and users

Managing a deployment that has errors

If a deployment has errors, the pipeline stops processing on it. You can perform actions such as rolling back the deployment or skipping the step on which the error occurred.

Do the following steps:

1. If the pipeline is not open, in the navigation pane of Deployment Manager, click **Pipelines** and then click the name of the pipeline.
2. Click the **More** icon, and then do one of the following actions:
    - To resume running the pipeline from the task, click **Resume from current task**.
    - To skip the step and continue running the pipeline, click **Skip current task and continue** .
    - To roll back to an earlier deployment, click **Rollback**.
        - Pega Platform 8.4 supports application-level rollback. To leverage this functionality, candidate and orchestrator environments must be on Deployment Manager 4.8 or above. For older versions of Deployment Manager (4.7.x and below) or users on Pega Platform 8.3 or below, rollback defaults to the system-level.

          For more information on rollback, see Rolling back a deployment.
    - To stop running the pipeline, click **Abort**.

Rolling back a deployment

A rollback can be performed on any deployment that encounters an error or fails for any reason. The rollback feature relies on restore points, which are automatically generated every time an import happens. Any change implemented after the import but before the next restore point will be lost when the rollback action is triggered.

Note: For Pega versions 8.3 and earlier the rollback execution impacts the entire system and is not scoped to the application. Be careful about using restore when there are multiple independent applications on the system.

For Pega versions 8.4 and later with PegaDevOpsFoundation 4.8 and later, this will be an application-level rollback with no impact on other applications. Refer to the Application-level rollback document for more information.

To perform a rollback, see Managing a deployment that has errors

Rolling back a successful deployment

If there is a situation where a successful deployment must be rolled back, perform the following steps:

1. Insert a manual step as the final task in the stage where rollback is required.
2. Assign the manual step to a user who will make the decision to approve/reject the deployment.
3. To rollback, reject the deployment via email or directly in Deployment Manager.

Performing ad hoc tasks

By using ad hoc tasks, you can independently perform package and deploy actions outside of the context of a deployment, which is useful for exporting artifacts from the production environment and importing them into a lower environment, or for packaging and deploying artifacts if a Deployment Manager pipeline is not available.

Note: Ad hoc tasks are restricted to Super Admin users within Deployment Manager, or any user that has access to the **Run ad hoc task** privilege in Deployment Manager. If import conflicts are encountered when using ad hoc tasks, the conflict will always be overwritten.

1. If a pipeline is not open, in the navigation pane of Deployment Manager, click **Pipelines Application pipelines** , and then click the name of a pipeline.
2. From the navigation banner, click **Actions Run ad hoc tasks** .
3. In the **Run ad hoc tasks** window, select an environment and an authentication profile.
4. Select a task that you want to run on an artifact:
    - To package an artifact, on the **Task** menu click **Package**, and enter the repository name, product name, and version that you want to package.
    - To deploy an artifact, on the **Task** menu click **Deploy**, and enter the repository name and artifact path for the package you want to deploy.
5. Select **Execute task**

Troubleshooting issues with your pipeline

Deployment Manager provides several features that help you troubleshoot and resolve issues with your pipeline.

You can:

- View deployment logs for information about the completion status of operations.
- Run diagnostics to verify that your environment is correctly configured.
- Stop all deployments that are running on a pipeline.
- Use a chatbot to obtain information about common issues.

- Viewing deployment logs

  View logs for a deployment to see the completion status of operations, for example, when a data simulation is moved to the simulation environment. You can change the logging level to control which events are displayed in the log.

- Diagnosing a pipeline

  You can diagnose your pipeline to troubleshoot issues and verify that your pipeline is configured properly.

- Stopping all deployments

  You can stop all the deployments on a pipeline at once to quickly troubleshoot issues and resolve failed pipelines.

- Obtaining information about common issues by using the chatbot

  Deployment Manager provides a chatbot that you can use to obtain information about common issues, such as connectivity between systems, Jenkins configuration, and branch merging. After you enter your search text, the chatbot provides you with relevant answers and links to more information.

Viewing deployment logs

View logs for a deployment to see the completion status of operations, for example, when a deployment moves from staging to production. When the Deploy task runs, the application package is imported in to the candidate system. By default, logs record all the new rule and data instances and all the updated rule and data instances that are in this application package. You can disable the logging of such rule and data types and can change the logging level to control which events are displayed in the log.

To view a deployment log, do the following steps:

1. In Dev Studio, on the appropriate candidate system, change the logging level to control which events the log displays. **For example:** For example, you can change logging levels of your deployment from INFO to DEBUG for troubleshooting purposes. For more information, see Logging Level Settings tool.
2. To disable logging of new and updated rule and data instances in imported application packages, perform the following steps:
    1. On the candidate system for which you want to disable reporting, in the navigation pane of Admin Studio, click **Resources Log categories** .
    2. On the **Log categories** page, for the *DeploymentManager.DeltaInstanceLogging* log level, click the **More** icon, and then click **Change logging level**.
    3. In the **Change pxBackgroundProcessing.Agents log level** dialog box, in the **Update log level of category to** list, select **OFF**.
    4. Click **Submit**.
3. If the pipeline is not open, in the navigation pane, click **Pipelines Application pipelines** .
4. Do one of the following actions:
    - To view the log for the current deployment, click the **More** icon, and then click **View logs**.
    - To view the log for a previous deployment, expand the **Deployment History** pane, and then click **Logs** for the deployment.

Diagnosing a pipeline

You can diagnose your pipeline to troubleshoot issues and verify that your pipeline is configured properly.

For example, you can determine if the target application and product rule are in the development environment, connectivity between systems and repositories is working, and pre-merge settings are correctly configured.

1. If the pipeline is not open, in the navigation pane of Deployment Manager, click **Pipelines** and then click the name of the pipeline.
2. Click **Actions Diagnose pipeline** .
3. In the **Diagnostics** window, review the errors, if any.
   Note: If the RMURL dynamic system setting is not configured, Deployment Manager displays a message that you can disregard if you are not using branches, because you do not need to configure the dynamic system setting.

Stopping all deployments

You can stop all the deployments on a pipeline at once to quickly troubleshoot issues and resolve failed pipelines.

Take the following steps to stop all deployments on a pipeline:

1. If the pipeline is not open, in the navigation pane of Deployment Manager, click **Pipelines** and then click the name of the pipeline.
2. Click **Actions Abort open deployments** .
3. In the **Abort open deployments** dialog box, enter a reason for stopping the deployments, and then click **OK**.

Obtaining information about common issues by using the chatbot

Deployment Manager provides a chatbot that you can use to obtain information about common issues, such as connectivity between systems, Jenkins configuration, and branch merging. After you enter your search text, the chatbot provides you with relevant answers and links to more information.

Before you begin: If the chatbot is disabled, enable it. For more information, see Enabling and disabling the chatbot. To use the Deployment Manager chatbot to help resolve issues, perform the following steps:

1. In the bottom right corner of the Deployment Manager portal, click the chatbot icon.
2. Do one of the following actions:
   a. Click the appropriate link from the list of issues that the chatbot displays.
   b. Enter text for which you want to receive more information, and then click **Enter**.
3. To clear the chatbot history, in the chatbot window, click the **More** icon, and then click **Clear chat history**.

- Enabling and disabling the chatbot

  Use the chatbot to obtain more information about common Deployment Manager issues, such as branch merging and pipeline configuration. You can disable and enable the chatbot. By default, the chatbot is enabled.

Enabling and disabling the chatbot

Use the chatbot to obtain more information about common Deployment Manager issues, such as branch merging and pipeline configuration. You can disable and enable the chatbot. By default, the chatbot is enabled.

Only super administrators can enable and disable the chatbot. For more information about user roles, see Understanding roles and users.

1. In the navigation pane, click **Settings General settings** .
2. Do one of the following actions:
   - To enable the chatbot, select the **Enable self-service Deployment Manager web chatbot** check box.
   - To disable the chatbot, clear the check box.
3. Click **Save**.
4. At the top of the **General Settings** page, click the **Page back** icon.
5. Click the **Refresh** icon to refresh Deployment Manager and apply your changes.

Understanding schema changes in application packages

If an application package that is to be deployed on candidate systems contains schema changes, the Pega Platform orchestration server checks the candidate system to verify that you have the required privileges to deploy the schema changes. One of the following results occurs:

- If you have the appropriate privileges, schema changes are automatically applied to the candidate system, the application package is deployed to the candidate system, and the pipeline continues.
- If you do not have the appropriate privileges, Deployment Manager generates an SQL file that lists the schema changes and sends it to your email address. It also creates a manual step, pausing the pipeline, so that you can apply the schema changes. After you complete the step, the pipeline continues. For more information about completing a

step, see Completing or rejecting a manual step.

You can also configure settings to automatically deploy schema changes so that you do not have to manually apply them if you do not have the required privileges. For more information, see Configuring settings to automatically deploy schema changes.

Your user role must have the appropriate permissions so that you can manage schema changes.

- Configuring settings to automatically apply schema changes

  You can configure settings to automatically deploy schema changes that are in an application package that is to be deployed on candidate systems. Configure these settings so that you do not have to apply schema changes if you do not have the privileges to deploy them.

- Understanding roles and users

Configuring settings to automatically apply schema changes

You can configure settings to automatically deploy schema changes that are in an application package that is to be deployed on candidate systems. Configure these settings so that you do not have to apply schema changes if you do not have the privileges to deploy them.

Do the following steps:

1. On the candidate system, in Pega Platform, set the *AutoDBSchemaChanges* dynamic system setting to true to enable schema changes at the system level.
   a. In Dev Studio, search for AutoDBSchemaChanges.
   b. In the dialog box that appears for the search results, click **AutoDBSchemaChanges**.
   c. On the **Settings** tab, in the **Value** field, enter true.
   d. Click **Save**.
2. Add the *SchemaImport* privilege to your access role to enable schema changes at the user level. For more information, see Specifying privileges for an Access of Role to Object rule.

Result: These settings are applied sequentially. If the *AutoDBSchemaChanges* dynamic system setting is set to false, you cannot deploy schema changes, even if you have the *SchemaImport* privilege.

For more information about the *database/AutoDBSchemaChanges* dynamic system setting, see Importing rules and data by using a direct connection to the database.

Schema changes are also attached to the deployment report for the pipeline.

- Viewing deployment reports for a specific deployment

Completing or rejecting a manual step

If a manual step is configured on a stage, the deployment pauses when it reaches the step, and you can either complete it or reject it if your role has the appropriate permissions. For example, if a user was assigned a task and completed it, you can complete the task in the pipeline to continue the deployment. Deployment Manager also sends you an email when there is a manual step in the pipeline. You can complete or reject a step either within the pipeline or through email.

Deployment Manager also generates a manual step if there are schema changes in the application package that the release manager must apply. For more information, see Schema changes in application packages.

To complete or reject a manual step within the deployment, do the following steps:

1. To complete or reject a manual step from within an email, click either **Accept** or **Reject**.
2. To complete or reject a manual step in the pipeline,
   a. If the pipeline is not open, in the navigation pane, click **Pipelines Application pipelines** , and then click the name of the pipeline.
   b. Accept or reject the step by doing one of the following actions:
   - To resolve the task so that the deployment continues through the pipeline, click **Complete**.
   - To reject the task so that the deployment does not proceed, click **Reject**.

- Understanding roles and users
- Continuing or stopping a deployment by adding the Perform manual step task

Managing aged updates

If your role has the appropriate permissions, you can manage aged updates in a number of ways, such as importing them, skipping the import, or manually deploying applications. Managing aged updates gives you more flexibility in how you deploy application changes.

Do the following steps:

1. If the pipeline is not open, in the navigation pane of Deployment Manager, click **Pipelines** and then click the name of the pipeline.
2. Click **View aged updates** to view a list of the rules and data instances, which are in the application package, that are older than the instances that are on the system.
3. Click the **More** icon and do one of the following actions:
    - To import the older rule and data instances that are in the application package into the system, which overwrites the newer versions that are on the system, click **Overwrite aged updates**.
    - To skip the import, click **Skip aged updates**.
    - To manually deploy the package from the Import wizard on the system, click **Deploy manually and resume**. Deployment Manager does not run the Deploy step on the stage.

- Understanding aged updates

  An aged update is a rule or data instance in an application package that is older than an instance that is on a system to which you want to deploy the application package. By being able to import aged updates, skip the import, or manually deploy your application changes, you now have more flexibility in determining the rules that you want in your application and how you want to deploy them.

Understanding aged updates

An aged update is a rule or data instance in an application package that is older than an instance that is on a system to which you want to deploy the application package. By being able to import aged updates, skip the import, or manually deploy your application changes, you now have more flexibility in determining the rules that you want in your application and how you want to deploy them.

For example, you can update a dynamic system setting on a quality assurance system, which has an application package that contains the older instance of the dynamic system setting. Before Deployment Manager deploys the package, the system detects that the version of the dynamic system setting on the system is newer than the version in the package and creates a manual step in the pipeline.

Archiving and activating pipelines

If your role has the appropriate permissions, you can archive inactive pipelines so that they are not displayed on the Deployment Manager landing page.

To archive or activate a pipeline, do the following steps:

1. In the navigation pane of Deployment Manager click **Pipelines Application Pipelines** .
2. To archive a pipeline, perform the following steps:
    a. Click the **More** icon, and then click **Archive** for the pipeline that you want to archive.
    b. In the **Archive pipeline** dialog box, click **Submit**.
3. To activate an archived pipeline, perform the following steps:
    a. Click **Pipelines Archived Pipelines** .
    b. Click **Activate** for the pipeline that you want to activate.
    c. In the **Activate pipeline** dialog box, click **Submit**.

- Understanding roles and users

Disabling and enabling a pipeline

If your role has the appropriate permissions, you can disable a pipeline on which errors continuously cause a deployment to fail. Disabling a pipeline prevents branch merging, but you can still view, edit, and stop deployments on a disabled pipeline.

To disable and enable a pipeline, perform the following steps:

1. In the navigation pane of Deployment Manager click **Pipelines Application pipelines** .
2. To disable a pipeline, perform the following steps:
    a. Click the **More** icon, and then click **Disable** for the pipeline that you want to disable.
    b. In the **Disable pipeline** dialog box, click **Submit**.
3. To enable a disabled pipeline, click the **More** icon, and then click **Enable**.

- Understanding roles and users

Deleting a pipeline

If your role has the appropriate permission, you can delete a pipeline. When you delete a pipeline, its associated application packages are not removed from the repositories that the pipeline is configured to use.

To delete a pipeline, do the following steps:

1. If the pipeline is not open, in the navigation pane of Deployment Manager, click **Pipelines** and then click the name of

the pipeline.
2. Click the **More** icon, and then click **Delete** for the pipeline that you want to delete.
3. In the **Delete pipeline** dialog box, click **Submit**.

- Understanding roles and users

Using data migration pipelines with Deployment Manager 4.8.x

Data migration tests provide you with significant insight into how the changes that you make to decision logic affect the results of your strategies. To ensure that your simulations are reliable enough to help you make important business decisions, you can deploy a sample of your production data to a dedicated data migration test environment.

When you use Deployment Manager 4.8.x in data migration pipelines, you automate exporting data from the production environment and into the simulation environment. Data migration pipelines also require the following:

- Pega Platform 8.3.x or 8.4.x
- Decision management
- Pega Marketing

For more information about data migration pipelines, see these articles on Pega Community:

- Deploying sample production data to a simulation environment for testing
- Creating simulation tests

For information about using all the Deployment Manager 4.8.x features, see Configuring and running pipelines with Deployment Manager 4.8.x.

- Installing, upgrading, and configuring Deployment Manager 4.8.x for data migration pipelines

  You can use Deployment Manager 4.6.x or later in data migration pipelines so that you can automatically export simulation data from a production system and import it into a simulation system. For more information about using Deployment Manager 4.8.x with data migration pipelines, see .

- Exporting and importing simulation data automatically with Deployment Manager

  Create and run data migration pipelines in Deployment Manager 4.8.x to automatically export simulation data from a production environment into a simulation environment in which you can test simulation data. You can also use Deployment Manager to monitor and obtain information about your simulations, for example, by running diagnostics to ensure that your environment configurations are correct and by and viewing reports that display key performance indicators (KPIs).

Installing, upgrading, and configuring Deployment Manager 4.8.x for data migration pipelines

You can use Deployment Manager 4.6.x or later in data migration pipelines so that you can automatically export simulation data from a production system and import it into a simulation system. For more information about using Deployment Manager 4.8.x with data migration pipelines, see Exporting and importing simulation data automatically with Deployment Manager.

To install, upgrade, and configure Deployment Manager on the simulation and production environments and on the orchestration server, perform the following steps:

1. Install or upgrade Deployment Manager.
   - For first-time installations or upgrades from Deployment Manager 3.2.1, install Deployment Manager on the candidate systems (production and simulation environments) and the orchestration server. Upgrading is done automatically, and you do not need to do post-upgrade steps.

     For more information, see Installing or upgrading to Deployment Manager 4.8.x.
2. For first-time installations, configure communication between the orchestration server and the candidate systems:
   a. Enable the default operators on each system.
   For more information, see Configuring authentication profiles.
   b. Configure the authentication profiles, which enable communication between systems, on each system. Deployment Manager provides default authentication profiles, or you can create your own.
   For more information, see Configuring authentication profiles.
3. To move the orchestration server to a different environment, migrate your pipelines to the new orchestration server, and then, on the new orchestration server, configure the URL of the new orchestration server. This URL is used to update the task status on the orchestration server and diagnostics checks.
   For more information, see step 2 in Configuring the orchestration server.

Exporting and importing simulation data automatically with Deployment Manager

Create and run data migration pipelines in Deployment Manager 4.8.x to automatically export simulation data from a production environment into a simulation environment in which you can test simulation data. You can also use Deployment Manager to monitor and obtain information about your simulations, for example, by running diagnostics to ensure that your environment configurations are correct and by and viewing reports that display key performance indicators (KPIs).

- Creating a pipeline

  Create a pipeline by defining the production and simulation environments and the application details for the pipeline. By using a data migration pipeline, you can export and import simulation data automatically.

- Modifying a pipeline

  You can change the URLs of your production and simulation environments. You can also change the application information for which you are creating the pipeline.

- Diagnosing a pipeline

  You can diagnose your pipeline to verify its configuration. For example, you can verify that the orchestration system can connect to the production and simulation environments.

- Scheduling a pipeline by creating a job scheduler rule

  You can schedule a data migration pipeline to run during a specified period of time by creating and running a job scheduler. The job scheduler runs a Deployment Manager activity (pzScheduleDataSyncPipeline) on the specified pipeline, based on your configuration, such as weekly or monthly.

- Starting a pipeline manually

  If you do not run a data migration pipeline based on a job scheduler, you can run it manually in Deployment Manager.

- Pausing a pipeline

  Pause a pipeline to stop processing the data migration. When you pause a data migration, the pipeline completes the current task and stops the data migration.

- Stopping a pipeline

  Stop a pipeline to stop data migrations from being exported and imported.

- Stopping and resuming a pipeline that has errors

  If a data migration has errors, the pipeline stops processing on it, and you can either resume or stop running the pipeline.

- Deleting a pipeline

  When you delete a pipeline, its associated application packages are not deleted from the pipeline repositories.

- Viewing data migration logs

  View the logs for a data migration to see the completion status of operations, for example, when a data migration moves to a new stage. You can change the logging level to control the events are displayed in the log. For example, you can change logging levels of your deployment from INFO to DEBUG for troubleshooting purposes. For more information, see Logging Level Settings tool.

- Viewing a report for a specific data migration

  You can view a report for a specific data migration to gain more visibility into data migrations on a pipeline.

- Viewing reports for all data migrations in a pipeline

  Reports provide a variety of information about all the data migrations in your pipeline so that you can gain more visibility into data migration processing. For example, you can view the average time taken to complete data migrations.

Creating a pipeline

Create a pipeline by defining the production and simulation environments and the application details for the pipeline. By using a data migration pipeline, you can export and import simulation data automatically.

Do the following steps:

1. If the pipeline is not open, in the navigation pane of Deployment Manager, click **Pipelines Data migration pipelines** , and then click the name of the pipeline.
2. Click **New**.
3. On the **Environment Details** page, if you are using Deployment Manager on-premises, configure environment

details.
This information is automatically populated if you are using Deployment in Pega Cloud Services environments, but you can change it.
   a. In the **Environment** fields, enter the URLs of the production and simulation environments.
   b. If you are using your own authentication profiles, from the **Auth profile** lists, select the authentication profiles that you want the orchestration server to use to communicate with the production and simulation environments.
   c. Click **Next**.
4. On the **Application details** page, specify the application information for which you are creating the pipeline.
   a. From the **Application** list, select the name of the application.
   b. From the **Version** list, select the application version.
   c. From the **Access group** list, select the access group for which you want to run pipeline tasks. This access group must be present on the production and simulation environments and have at least the *sysadmin4* role.
   d. In the **Name of the pipeline** field, enter the pipeline name.
   e. Click **Next**.

   Result: The **Pipeline** page displays the stages and tasks, which you cannot delete, that are in the pipeline.
5. Click **Finish**.

Modifying a pipeline

You can change the URLs of your production and simulation environments. You can also change the application information for which you are creating the pipeline.

1. If the pipeline is not open, in the navigation pane of Deployment Manager, click **Pipelines Data migration pipelines** , and then click the name of the pipeline.
2. Click **Action Settings** .
3. Modify environment details by doing the following:
   a. Click **Environment Details**.
   b. In the **Environment** fields, enter the URLs of the production and simulation environments.
4. To change the application information for which you are creating the pipeline, click **Application details**.
   a. From the **Version** list, select the application version.
   b. From the **Access group** list, select the access group for which you want to run pipeline tasks.
   This access group must be present on the production and simulation environments and have at least the *sysadmin4* role.
5. Click **Save**.

Diagnosing a pipeline

You can diagnose your pipeline to verify its configuration. For example, you can verify that the orchestration system can connect to the production and simulation environments.

1. If the pipeline is not open, in the navigation pane of Deployment Manager, click **Pipelines Data migration pipelines** , and then click the name of the pipeline.
2. Click **Actions Diagnose pipeline** .
3. In the **Diagnostics** window, review the errors, if any.

Scheduling a pipeline by creating a job scheduler rule

You can schedule a data migration pipeline to run during a specified period of time by creating and running a job scheduler. The job scheduler runs a Deployment Manager activity (*pzScheduleDataSyncPipeline*) on the specified pipeline, based on your configuration, such as weekly or monthly.

For more information about job scheduler rules, see Job Scheduler rules.

Do the following steps:

1. On the orchestration server, in the navigation panel of Dev Studio, click **Records SysAdmin Job Scheduler** .
2. On the **Create Job Scheduler** rule form, enter the label of the scheduler and select the ruleset into which to save the job scheduler.
3. Click **Create and open**.
4. On the **Edit Job Scheduler** rule form, on the **Definition** tab, from the **Runs on** list, configure the job scheduler to run on all or one nodes:
   - To run the job scheduler on all nodes in a cluster, click **All associated nodes**.
   - To run the job scheduler on only one node in a cluster, click **Any one associated node**.
5. From the **Schedule** list, select how often you want to start the job scheduler, and then specify the options for it.
6. Select the context for the activity resolution.
   - If you want to resolve the *pzScheduleDataSyncPipeline* activity in the context of Deployment Manager, go to step 7.
   - If you want to resolve the activity in the context that is specified in the System Runtime Context, go to step 8.
7. To resolve the *pzScheduleDataSyncPipeline* activity in the context of Deployment Manager:
   a. From the **Context** list, select **Specify access group**.
   b. In the **Access group** field, press the Down arrow key and select the access group that can access Deployment Manager.

    c. Go to step 9.
8. to resolve the activity in the context that is specified in the System Runtime Context:
    a. From the **Context** list, select **Use System Runtime Context**.
    b. Update the access group of the batch requestor type access group with the access group that can access Deployment Manager; in the header of Dev Studio, clicking **Configure System General** .
    c. On the **System:General** page, on the **Requestors** tab, click the **BATCH** requestor type.
    d. On the **Edit Requestor Type** rule form, on the **Definition** tab, in the **Access Group Name** field, press the Down arrow key and select the access group that can access Deployment Manager.
    e. Click **Save**.
9. On the **Job Schedule** rule form, in the **Class** field, press the Down arrow key and select **Pega-Pipeline-DataSync**.
10. In the **Activity** field, press the Down arrow key and select **pzScheduleDataSyncPipeline**.
11. Click the **Parameters** link that appears below the **Activity** field.
12. In the **Activity Parameters** dialog box, in the **Parameter value** field for the **PipelineName** parameter, enter the data migration pipeline that the job scheduler runs.
13. In the **Parameter** value field for the **ApplicationName** parameter, enter the application that the data migration pipeline is running.
14. Click **Submit**.
15. Save the Job schedule rule form.

Starting a pipeline manually

If you do not run a data migration pipeline based on a job scheduler, you can run it manually in Deployment Manager.

1. Do one of the following actions:
    - If the pipeline for which you want to run a data migration is open, click **Start data migration**.
    - If the pipeline is not open, click **Pipelines Data migration pipelines** , and then click **Start data migration**.
2. In the **Start data migration** dialog box, click **Yes**.

Pausing a pipeline

Pause a pipeline to stop processing the data migration. When you pause a data migration, the pipeline completes the current task and stops the data migration.

1. If the pipeline is not open, in the navigation pane of Deployment Manager, click **Pipelines Data migration pipelines** , and then click the name of the pipeline.
2. Click **Pause**.

Stopping a pipeline

Stop a pipeline to stop data migrations from being exported and imported.

1. If the pipeline is not open, in the navigation pane of Deployment Manager, click **Pipelines Data migration pipelines** , and then click the name of the pipeline.
2. Click the **More** icon, and then click **Abort**.

Stopping and resuming a pipeline that has errors

If a data migration has errors, the pipeline stops processing on it, and you can either resume or stop running the pipeline.

1. If the pipeline is not open, in the navigation pane of Deployment Manager, click **Pipelines Data migration pipelines** , and then click the name of the pipeline.
2. Click the **More** icon, and then do one of the following:
    - To resume running the pipeline from the task, click **Start data migration pipeline**.
    - To stop running the pipeline, click **Abort**.

Deleting a pipeline

When you delete a pipeline, its associated application packages are not deleted from the pipeline repositories.

1. If the pipeline is not open, in the navigation pane of Deployment Manager, click **Pipelines Data migration pipelines** , and then click the name of the pipeline.
2. Click the **Delete** icon for the pipeline that you want to delete.
3. Click **Submit**.

Viewing data migration logs

View the logs for a data migration to see the completion status of operations, for example, when a data migration moves to a new stage. You can change the logging level to control the events are displayed in the log. For example, you can change logging levels of your deployment from INFO to DEBUG for troubleshooting purposes. For more information, see Logging Level Settings tool.

1. If the pipeline is not open, in the navigation pane of Deployment Manager, click **Pipelines Data migration pipelines** , and then click the name of the pipeline.
2. Do one of the following actions:
    - To view the log for the current data migration, click the **More** icon, and then click **View logs**.
    - To view the log for a previous data migration, expand the **Deployment History** pane and click **Logs** for the appropriate deployment.

Viewing a report for a specific data migration

You can view a report for a specific data migration to gain more visibility into data migrations on a pipeline.

1. If the pipeline is not open, in the navigation pane of Deployment Manager, click **Pipelines Data migration pipelines** , and then click the name of the pipeline.
2. Perform one of the following actions:
    - To view the report for the current deployment, click the **More** icon, and then click **View report**.
    - To view the report for a previous deployment, expand the **Deployment History** pane and click **Reports** for the appropriate deployment.

Viewing reports for all data migrations in a pipeline

Reports provide a variety of information about all the data migrations in your pipeline so that you can gain more visibility into data migration processing. For example, you can view the average time taken to complete data migrations.

You can view the following key performance indicators (KPI):

- Data migration success – Percentage of successfully completed data migrations
- Data migration frequency – Frequency of new deployments to production
- Data migration speed – Average time taken to complete data migrations
- Start frequency – Frequency at which new data migrations are triggered
- Failure rate – Average number of failures per data migration

To view reports, do the following tasks:

1. Do one of the following actions:
    - If the pipeline is open, click **Actions >View report**.
    - If a pipeline is not open, in the navigation pane, click **Reports**. Next, in the **Pipeline** field, press the Down arrow key and select the name of the pipeline for which to view the report.
2. From the list that appears in the top right of the **Reports** page, select whether you want to view reports for all deployments, the last 20 deployments, or the last 50 deployments.

Deployment Manager 3.4.x

Use Deployment Manager to configure and run continuous integration and delivery (CI/CD) workflows for your Pega applications from within Pega Platform. You can create a standardized deployment process so that you can deploy predictable, high-quality releases without using third-party tools.

With Deployment Manager, you can fully automate your CI/CD workflows, including branch merging, application package generation, artifact management, and package promotion to different stages in the workflow.

Deployment Manager 3.4.x is compatible with Pega 7.4. You can download it for Pega Platform from the Deployment Manager Pega Exchange page.**Note:** These topics describe the features for the latest version of Deployment Manager 3.4.x.

Note: Each customer Virtual Private Cloud (VPC) on Pega Cloud Services has a dedicated orchestrator instance to use Deployment Manager. You do not need to install Deployment Manager to use with your Pega Cloud Services application.

- Installing, upgrading, and configuring Deployment Manager 3.4.x

  Use Deployment Manager to create continuous integration and delivery (CI/CD) pipelines, which automate tasks and allow you to quickly deploy high-quality software to production.

- Configuring and running pipelines with Deployment Manager 3.4.x

  Use Deployment Manager to create continuous integration and delivery (CI/CD) pipelines, which automate tasks so that you can quickly deploy high-quality software to production.

Installing, upgrading, and configuring Deployment Manager 3.4.x

Use Deployment Manager to create continuous integration and delivery (CI/CD) pipelines, which automate tasks and allow you to quickly deploy high-quality software to production.

Note: This document describes the features for the latest version of Deployment Manager 3.4.x.

- Installing Deployment Manager 3.4.x

  Install Deployment Manager 3.4.x on-premises. Each customer virtual private cloud (VPC) on Pega Cloud has a dedicated orchestrator instance to use Deployment Manager. You do not need to install Deployment Manager to use it with your Pega Cloud application. If you are upgrading from an earlier release to Deployment Manager 3.4.x, contact Pegasystems Global Customer Support (GCS) to request a new version.

- Upgrading to Deployment Manager 3.4.x

  After you install Deployment Manager 3.4.x, you must do post-upgrade steps. Before you upgrade, ensure that no deployments are running, have errors, or are paused.

- Configuring systems in the pipeline

  Complete the following tasks to set up a pipeline for all supported CI/CD workflows. If you are using branches, you must configure additional settings after you perform the required steps.

- Configuring the development system

  After you configure the orchestration server and all your candidate systems, configure additional settings so that you can create pipelines if you are using branches in a distributed or non-distributed branch-based environment.

- Configuring additional settings (optional)

  As part of your pipeline, you can optionally send email notifications to users or configure Jenkins if you are using a Jenkins task.

Installing Deployment Manager 3.4.x

Install Deployment Manager 3.4.x on-premises. Each customer virtual private cloud (VPC) on Pega Cloud has a dedicated orchestrator instance to use Deployment Manager. You do not need to install Deployment Manager to use it with your Pega Cloud application. If you are upgrading from an earlier release to Deployment Manager 3.4.x, contact Pegasystems Global Customer Support (GCS) to request a new version.

If you are using Deployment Manager on premises, complete the following steps to install it:

Note: If you are upgrading from Deployment Manger 3.2.1, after you import files on premises or Deployment Manager 3.4.x is deployed on Pega Cloud Services, finish the upgrade immediately so that your pipelines work in Deployment Manager 3.4.x.

1. Install Pega 7.4 on all systems in the CI/CD pipeline.
2. Browse to the Deployment Manager Pega Marketplace page, and then download the **DeploymentManager03.0240x.zip** file to your local disk on each system.
3. Extract the **DeploymentManager03.0240x.zip** file.
4. Use the Import wizard to import files into the appropriate systems. For more information about the Import wizard, see Importing a file by using the Import wizard.
5. On the orchestration server, import the following files:
   - **PegaDevOpsFoundation_03.04.0x.zip**
   - **PegaDeploymentManager_03.04.0x.zip**
6. On the development, QA, staging, and production systems, import the **PegaDevOpsFoundation_03.04.0x.zip** file.
7. If you are using a distributed development, on the remote development system, import the **PegaDevOpsFoundation_03.04.0x**.zip file.
8. Do one of the following actions
   - If you are upgrading to Deployment Manager 3.4.x, perform the upgrade. For more information, see Upgrading to Deployment Manager 3.4.x.
   - If you are not upgrading Deployment Manager 3.4.x, continue the installation procedure. For more information, see Configuring the orchestration server.

Upgrading to Deployment Manager 3.4.x

After you install Deployment Manager 3.4.x, you must do post-upgrade steps. Before you upgrade, ensure that no deployments are running, have errors, or are paused.

To upgrade to Deployment Manager 3.4.x either on Pega Cloud Services or on premises, perform the following steps:

1. Enable default operators and configure authentication profiles on the orchestration server and candidate systems. For more information, see Configuring authentication profiles on the orchestration server and candidate systems.
2. On each candidate system, add the PegaDevOpsFoundation application to your application stack.
   a. In the Designer Studio header, click the name of your application, and then click **Definition**.
   b. In the **Built on application** section, click **Add application**.
   c. In the **Name** field, press the Down arrow key and select **PegaDevOpsFoundation**.
   d. In the **Version** field, press the Down arrow key and select the version of Deployment Manager that you are using.
   e. Click **Save**.

If you are upgrading from Deployment Manager 3.2.1, you do not need to do the rest of the steps in this procedure or the required steps in the remainder of this document. If you are upgrading from earlier releases and have pipelines configured, complete this procedure.

Note: If you are upgrading from Deployment Manager 3.2.1, you do not need to do the rest of the steps in this procedure or the required configuration steps. If you are upgrading from earlier releases and have pipelines configured, complete this procedure.

3. On the orchestration server, log in to the release management application.
4. In Designer Studio, search for *pxUpdatePipeline*, and then click the activity in the dialog box that displays the results.
5. Click **Actions Run** .
6. In the dialog box that is displayed, click **Run**.
7. Modify the current release management application so that it is built on PegaDeploymentManager:03-04-01.
    a. In the Designer Studio header, click the name of your application, and then click **Definition**.
    b. In the **Edit Application** rule form, on the **Definition** tab, in the **Built on application** section, for the PegaDeploymentManager application, press the Down arrow key and select **03.04.01**.
    c. Click **Save**.
8. Merge rulesets to the PipelineData ruleset.
    a. Click **Designer Studio System Refactor Rulesets** .
    b. Click **Copy/Merge RuleSet**.
    c. Click the **Merge Source RuleSet(s) to Target RuleSet** radio button.
    d. Click the **RuleSet Versions** radio button.
    e. In the **Available Source Ruleset(s)** section, select the first open ruleset version that appears in the list, and then click the **Move** icon.
    Note: All your current pipelines are stored in the first open ruleset. If you modified this ruleset after you created the application, select all the ruleset versions that contain pipeline data.
9. In the target **RuleSet/Information** section, in the **Name** field, press the Down arrow key and select **Pipeline Data**.
10. In the **Version** field, enter 01-01-01.
11. For the **Delete Source RuleSet(s) upon completion of merge?** option, click **No**.
12. Click **Next**.
13. Click **Merge** to merge your pipelines to the PipelineData:01-01-01 ruleset.
14. Click **Done**.

Result: Your pipelines are migrated to the Pega Deployment Manager application. Log out of the orchestration server and log back in to it with the DMReleaseAdmin operator ID and the password that you specified for it. **Note:** You do not need to perform any of the required configuration procedures.

Configuring systems in the pipeline

Complete the following tasks to set up a pipeline for all supported CI/CD workflows. If you are using branches, you must configure additional settings after you perform the required steps.

To configure systems in the pipeline, do the following steps:

1. Configuring authentication profiles on the orchestration server and candidate systems
2. Configuring the orchestration server
3. Configuring candidate systems
4. Configuring repositories on the orchestration server and candidate systems

- Configuring authentication profiles on the orchestration server and candidate systems

  Deployment Manager provides default operator IDs and authentication profiles. You must enable the default operator IDs and configure the authentication profiles that the orchestration server uses to communicate with the candidate systems.

- Configuring the orchestration server

  The orchestration server is the system on which release managers configure and manage CI/CD pipelines. Configure it before you use it in your pipeline.

- Configuring candidate systems

  Configure each system that is used for the development, QA, staging, and production stage in the pipeline.

- Configuring repositories on the orchestration server and candidate systems

  If you are using Deployment Manager on-premises, create repositories on the orchestration server and all candidate systems to move your application between all the systems in the pipeline. You can use a supported repository type that is provided in Pega Platform, or you can create a custom repository type.

Configuring authentication profiles on the orchestration server and candidate systems

Deployment Manager provides default operator IDs and authentication profiles. You must enable the default operator IDs and configure the authentication profiles that the orchestration server uses to communicate with the candidate systems.

Configure the default authentication profile by doing these steps:

1. On the orchestration server, enable the DMReleaseAdmin operator ID and specify its password.
    a. Log in to the orchestration server with administrator@pega.com/install.
    b. In Designer Studio, click **Records Organization Operator ID** , and then click **DMReleaseAdmin**.
    c. In the Designer Studio header, click the operator ID initials, and then click **Operator**.
    d. On the Edit Operator ID rule form, click the **Security** tab.
    e. Clear the **Disable Operator** check box.
    f. Click **Save**.
    g. Click **Update password**.
    h. In the **Change Operator ID Password** dialog box, enter a password, reenter it to confirm it, and then click **Submit**.
    i. Clear the **Force password change on next login** check box if you do not want to change the password for the DMReleaseAdmin operator ID the next time that you log in.
    j. Log out of the orchestration server.
2. On each candidate system, update the DMReleaseAdmin authentication profile to use the new password. All candidate systems use this authentication profile to communicate with the orchestration server about the status of the tasks in the pipeline.
    a. Log in to each candidate system with the DMAppAdmin user name and the password that you specified.
    b. Click **Records Security Authentication Profile** .
    c. Click **DMReleaseAdmin**.
    d. On the **Edit Authentication Profile** rule form, click **Set password**.
    e. In the **Password** dialog box, enter the password, and then click **Submit**.
    f. Save the rule form.
3. On each candidate system, which includes the development, QA, staging, and production systems, enable the DMAppAdmin operator ID. If you want to create your own operator IDs, ensure that they point to the PegaDevOpsFoundation application.
    a. Log in to each candidate system with administrator@pega.com/install.
    b. In Designer Studio, click **Records Organization Operator ID** , and then click **DMAppAdmin**.
    c. In the Designer Studio header, click the operator ID initials, and then click **Operator**.
    d. On the **Edit Operator ID** rule form, click the **Security** tab.
    e. Clear the **Disable Operator** check box.
    f. Click **Save**.
    g. Click **Update password**.
    h. In the **Change Operator ID Password** dialog box, enter a password, reenter it to confirm it, and then click **Submit**.
    i. Clear the **Force password change on next login** check box if you do not want to change the password for the DMReleaseAdmin operator ID the next time that you log in.
    j. Log out of each candidate system.
4. On the orchestration server, modify the DMAppAdmin authentication profile to use the new password. The orchestration server uses this authentication profile to communicate with candidate systems so that it can run tasks in the pipeline.
    a. Log in to the orchestration server with the DMAppAdmin user name and the password that you specified.
    b. Click **Records Security Authentication Profile** .
    c. Click **DMAppAdmin**.
    d. On the Edit Authentication Profile rule form, click **Set password**.
    e. In the **Password** dialog box, enter the password, and then click **Submit**.
    f. Save the rule form.
5. Do one of the following actions:
    a. If you are upgrading to Deployment Manager 3.4.x, resume the upgrade procedure from step 2. For more information, see Upgrading to Deployment Manager 3.4.x.
    b. If you are not upgrading, continue the installation procedure. For more information, see Configuring the orchestration server.

- Understanding default operator IDs and authentication profiles

  When you install Deployment Manager on all the systems in your pipeline, default applications, operator IDs, and authentication profiles that communicate between the orchestration server and candidate systems are also installed.

Understanding default operator IDs and authentication profiles

When you install Deployment Manager on all the systems in your pipeline, default applications, operator IDs, and authentication profiles that communicate between the orchestration server and candidate systems are also installed.

On the orchestration server, the following items are installed:

- The Pega Deployment Manager application.
- The DMReleaseAdmin operator ID, which release managers use to log in to the Pega Deployment Manager application. You must enable this operator ID and specify its password.
- The DMAppAdmin authentication profile. You must update this authentication profile to use the password that you

specified for the DMAppAdmin operator ID, which is configured on all the candidate systems.

On all the candidate systems, the following items are installed:

- The PegaDevOpsFoundation application.
- The DMAppAdmin operator ID, which points to the PegaDevOpsFoundation application. You must enable this operator ID and specify its password.
- The DMReleaseAdmin authentication profile. You must update this authentication profile to use the password that you specified for the DMReleaseAdmin operator ID, which is configured on the orchestration server.

Note: The DMReleaseAdmin and DMAppAdmin operator IDs do not have default passwords.

Configuring the orchestration server

The orchestration server is the system on which release managers configure and manage CI/CD pipelines. Configure it before you use it in your pipeline.

To configure the orchestration server, complete the following tasks:

1. If your system is not configured for HTTPS, verify that TLS/SSL settings are not enabled on the api and cicd service packages.
   a. Click **Records Integration-Resources Service Package** .
   b. Click **api**.
   c. On the **Context** tab, verify that the **Require TLS/SSL for REST services in this package** check box is cleared.
   d. Click **Records Integration-Resources Service Package** .
   e. Click **cicd**.
   f. On the **Context** tab, verify that the **Require TLS/SSL for REST services in this package** check box is cleared.
2. Configure the candidate systems in your pipeline.
   For more information, see Configuring candidate systems.

Configuring candidate systems

Configure each system that is used for the development, QA, staging, and production stage in the pipeline.

Do the following steps:

1. On each candidate system, add the PegaDevOpsFoundation application to your application stack.
   a. In the Designer Studio header, click the name of your application, and then click **Definition**.
   b. In the **Built on application** section, click **Add application**.
   c. In the **Name** field, press the Down arrow key and select **PegaDevOpsFoundation**.
   d. In the **Version** field, press the Down arrow key and select the version of Deployment Manager that you are using.
   e. Click **Save**.
2. If your system is not configured for HTTPS, verify that TLS/SSL settings are not enabled on the api and cicd service packages.
   a. Click **Records Integration-Resources Service Package** .
   b. Click **api**.
   c. On the **Context** tab, verify that the **Require TLS/SSL for REST services in this package** check box is cleared.
   d. Click **RecordsIntegration-ResourcesService Package**.
   e. Click **cicd**.
   f. On the **Context** tab, verify that the Require TLS/SSL for REST services in this package check box is cleared.
3. To use a product rule for your target application, test application, or both, other than the default rules that are created by the New Application wizard, on the development system, create product rules that define the test application package and the target application package that will be moved through repositories in the pipeline.

   For more information, see *Product rules: Completing the Create, Save As, or Specialization form*.

   When you use the New Application wizard, a default product rule is created that has the same name as your application.
4. Configure repositories through which to move artifacts in your pipeline.
   For more information, see Configuring repositories on the orchestration server and candidate systems.

Configuring repositories on the orchestration server and candidate systems

If you are using Deployment Manager on-premises, create repositories on the orchestration server and all candidate systems to move your application between all the systems in the pipeline. You can use a supported repository type that is provided in Pega Platform, or you can create a custom repository type.

If you are using Deployment Manager on Pega Cloud Services, default repositories are provided. If you want to use repositories other than the ones provided, you can create your own.

For more information about creating a supported repository type, see Creating a repository connection.

For more information about creating a custom repository type, see Creating and using custom repository types for Deployment Manager.

When you create repositories, note the following information:

- The Pega repository type is not supported.
- Ensure that each repository has the same name on all systems.
- When you create JFrog Artifactory repositories, ensure that you create a Generic package type in JFrog Artifactory. Also, when you create the authentication profile for the repository on Pega Platform, you must select the **Preemptive authentication** check box.

After you configure a pipeline, you can verify that the repository connects to the URL of the development and production repositories by clicking **Test Connectivity** on the **Repository** rule form.

Configuring the development system

After you configure the orchestration server and all your candidate systems, configure additional settings so that you can create pipelines if you are using branches in a distributed or non-distributed branch-based environment.

To configure the development system, complete the following steps:

1. On the development system (in nondistributed environment) or the main development system (in a distributed environment), create a dynamic system setting to define the URL of the orchestration server, even if the orchestration server and the development system are the same system.
   a. Click **Create Records SysAdmin Dynamic System Settings** .
   b. In the **Owning Ruleset** field, enter Pega-DevOps-Foundation.
   c. In the **Setting Purpose** field, enter RMURL.
   d. Click **Create and open**.
   e. On the **Settings** tab, in the **Value** field, enter the URL of the orchestration server. Use this format: http://*hostname:port*/prweb/PRRestService.
   f. Click **Save**.
2. On either the development system (in a non-distributed environment) or the remote development system (in a distributed environment), use the New Application wizard to create a new development application that developers will log in to.
   This application allows development teams to maintain a list of development branches without modifying the definition of the target application.
3. On either the development system or remote development system, add the target application of the pipeline as a built-on application layer of the development application.
   a. Log in to the application.
   b. In the Designer Studio header, click the name of your application, and then click **Definition**.
   c. In the *Built-on application* section, click *Add application*.
   d. In the *Name* field, press the Down arrow key and select the name of the target application.
   e. In the **Version** field, press the Down arrow key and select the target application version.
   f. Click *Save*.
4. On either the development system or remote development system, lock the application rulesets to prevent developers from making changes to rules after branches have been merged.
   a. In the Designer Studio header, click the name of your application, and then click **Definition**.
   b. In the **Application rulesets** section, click the **Open** icon for each ruleset that you want to lock.
   c. Click **Lock and Save**.
5. To publish branches to a development system to start a branch to merge, configure a Pega repository.
   Merge branches by using the Merge Branch wizard. However, you can publish a branch to the remote development system to start a deployment. Publishing a branch when you have multiple pipelines per application is not supported.
   a. On either the development system or remote development system, in Designer Studio, enable Pega repository types.
      For more information, see *Enabling the Pega repository type*.
   b. Create a new Pega repository type. For more information, see *Creating a repository connection*.
   c. In the **Host ID** field, enter the URL of the development system.
   d. Ensure that the default access group of the operator that is configured for the authentication profile of this repository points to the pipeline application on the development system (in a nondistributed environment) or source development system (in a distributed environment).

Configuring additional settings (optional)

As part of your pipeline, you can optionally send email notifications to users or configure Jenkins if you are using a Jenkins task.

- Configuring email notifications on the orchestration server

You can optionally configure email notifications on the orchestration server. For example, users can receive emails when pre-merge criteria are not met and the system cannot create a deployment.

- Configuring Jenkins

  If you are using a Jenkins task in your pipeline, configure Jenkins.

Configuring email notifications on the orchestration server

You can optionally configure email notifications on the orchestration server. For example, users can receive emails when pre-merge criteria are not met and the system cannot create a deployment.

To configure the orchestration server to send emails, complete the following steps:

1. Use the Email wizard to configure an email account and listener by clicking **Designer Studio Integration Email Email Wizard** .
   This email account sends notifications to users when events occur, for example, if there are merge conflicts. For detailed information, see the procedure for “Configuring an email account that receives email and creates or manages work” in *Entering email information in the Email wizard*
2. From the **What would you like to do?** list, select **Receive an email and create/manage a work object**.
3. From the **What is the class of your work type?** list, select **Pega-Pipeline-CD**.
4. From the **What is your starting flow name?** list, select **NewWork**.
5. From the **What is your organization?** list, select the organization that is associated with the work item.
6. In the **What Ruleset?** field, select the ruleset that contains the generated email service rule.
   This ruleset applies to the work class.
7. In the **What RuleSet Version?** field, select the version of the ruleset for the generated email service rule.
8. Click **Next** to configure the email listener.
9. In the **Email Account Name** field, enter Pega-Pipeline-CD, which is the name of the email account that the listener references for incoming and outgoing email.
10. In the **Email Listener Name** field, enter the name of the email listener.
    Begin the name with a letter, and use only letters, numbers, the ampersand character (&), and hyphens.
11. In the **Folder Name** field, enter the name of the email folder that the listener monitors.
    Typically, this folder is INBOX.
12. In the **Service Package** field, enter the name of the service package to be deployed.
    Begin the name with a letter, and use only letters, numbers, and hyphens to form an identifier.
13. In the **Service Class** field, enter the service class name.
14. In the **Requestor User ID** field, press the Down arrow key, and select the operator ID of the release manager operator.
15. In the **Requestor Password** field, enter the password for the release manager operator.
16. In the **Requestor User ID** field, enter the operator ID that the email service uses when it runs.
17. In the **Password** field, enter the password for the operator ID.
18. Click **Next** to continue the wizard and configure the service package.
    For more information, see *Configuring the service package in the Email wizard*.
19. After you complete the wizard, enable the listener that you created in the Email Wizard.
    For more information, see *Starting a listener*.

- Understanding email notifications

  Emails are also preconfigured with information about each notification type. For example, when a deployment failure occurs, the email that is sent provides information, such as the pipeline name and URL of the system on which the deployment failure occurred.

Understanding email notifications

Emails are also preconfigured with information about each notification type. For example, when a deployment failure occurs, the email that is sent provides information, such as the pipeline name and URL of the system on which the deployment failure occurred.

Preconfigured emails are sent in the following scenarios:

- Deployment start – When a deployment starts, an email is sent to the release manager and, if you are using branches, to the operator who started a deployment.
- Deployment step failure – If any step in the deployment process is unsuccessful, the deployment pauses. An email is sent to the release manager and, if you are using branches, to the operator who started the branch merge.
- Deployment step completion – When a step in a deployment process is completed, an email is sent to the release manager and, if you are using branches, to the operator who started the branch merge.
- Deployment completion – When a deployment is successfully completed, an email is sent to the release manager and, if you are using branches, to the operator who started the branch merge.
- Stage completion – When a stage in a deployment process is completed, an email is sent to the release manager and, if you are using branches, to the operator who started the branch merge.
- Stage failure – If a stage fails to be completed, an email is sent to the release manager and, if you are using branches, to the operator who started the branch merge.
- Manual tasks requiring approval – When a manual task requires email approval from a user, an email is sent to the

user, who can approve or reject the task from the email.
- Stopped deployment – When a deployment is stopped, an email is sent to the release manager and, if you are using branches, to the operator who started the branch merge.
- Pega unit testing failure – If a Pega unit test cannot successfully run on a step in the deployment, an email is sent to the release manager and, if you are using branches, to the operator who started the branch merge.
- Pega unit testing success – If a Pega unit test is successfully run on a step in the deployment, an email is sent to the release manager and, if you are using branches, to the operator who started the branch merge.
- Schema changes required – If you do not have the required schema privileges to deploy the changes on application packages that require those changes, an email is sent to the operator who started the deployment.
- Guardrail compliance score failure – If you are using the Check guardrail compliance task, and the compliance score is less than the score that is specified in the task, an email with the score is sent to the release manager.
- Guardrail compliance score success – If you are using the Check guardrail compliance task, and the task is successful, an email with the score is sent to the release manager.
- Approve for production – If you are using the Approve for production task, which requires approval from a user before application changes are deployed to production, an email is sent to the user. The user can reject or approve the changes.
- Verify security checklist failure – If you are using the Verify security checklist task, which requires that all tasks be completed in the Application Security Checklist to ensure that the pipeline complies with security best practices, the release manager receives an email.
- Verify security checklist success – If you are using the Verify security checklist task, which requires that all tasks be completed in the Application Security Checklist to ensure that the pipeline complies with security best practices, the release manager receives an email.

Configuring Jenkins

If you are using a Jenkins task in your pipeline, configure Jenkins.

Do the following steps:

1. On the orchestration server, create an authentication profile that uses Jenkins credentials.
    a. Click **Create Security Authentication profile** .
    a. Enter a name, and then click **Create and open**.
    b. In the **User name** field, enter the user name of the Jenkins user.
    c. Click **Set password**, enter the Jenkins password, and then click **Submit**.
    d. Click the **Preemptive authentication** check box.
    e. Click **Save**.
2. Because the Jenkins task does not support Cross-Site Request Forgery (CSRF), disable it by completing the following steps:
    a. In Jenkins, click **Manage Jenkins**.
    b. Click **Configure Global Security**.
    c. In the **CSRF Protection** section, clear the **Prevent Cross Site Request Forgery exploits** check box.
    d. Click **Save**.
3. Install the Post build task plug-in.
4. Install the curl command on the Jenkins server.
5. Create a new freestyle project.
6. On the **General** tab, select the **This project is parameterized** check box.
7. Add the BuildID and CallBackURL parameters.
    a. Click **Add parameter**, and then select **String parameter**.
    b. In the **String** field, enter BuildID.
    c. Click **Add parameter**, and then select **String parameter**.
    d. In the **String** field, enter CallBackURL.
8. In the **Build Triggers** section, select the **Trigger builds remotely** check box.
9. In the **Authentication Token** field, select the token that you want to use when you start Jenkins jobs remotely.
10. In the **Build Environment** section, select the **Use Secret text(s) or file(s)** check box.
11. In the **Bindings** section, do the following actions:
    a. Click **Add**, and then select **User name and password (conjoined)**.
    b. In the **Variable** field, enter RMCREDENTIALS
    c. .In the **Credentials** field, click **Specific credentials**.
    d. Click **Add**, and then select **Jenkins**.
    e. In the **Add credentials** dialog box, in the **Username** field, enter the operator ID of the release manager operator that is configured on the orchestration server.
    f. In the **Password** field, enter the password.
    g. Click **Save**.
12. Configure information in the **Post-Build Actions** section, depending on your operating system:
    - If Jenkins is running on Microsoft Windows, go to step 13.
    - If Jenkins is running on Linux, go to step 14.
13. If Jenkins is running on Microsoft Windows, add the following post-build tasks:
    a. Click **Add post-build** action, and then select **Post build task**.
    b. In the **Log text** field, enter a unique string that for the message that is displayed in the build console output when a build fails, for example BUILD FAILURE**.**
    c. In the **Script** field, enter curl --user %RMCREDENTIALS% -H "Content-Type: application/json" -X POST --data "{\"jobName\":\"%JOB_NAME%\",\"buildNumber\":\"%BUILD_NUMBER%\",\"pyStatusValue\":\"FAIL\",\"pyID\":\"%BuildID%\"}" "%CallBackURL%"**.**

d. Click **Add another task**.
e. In the **Log text** field, enter a unique string that for the message that is displayed in the build console output when a build is successful, for example BUILD SUCCESS**.**
f. In the **Script** field, enter curl --user %RMCREDENTIALS% -H "Content-Type: application/json" -X POST --data "{\"jobName\":\"%JOB_NAME%\",\"buildNumber\":\"%BUILD_NUMBER%\",\"pyStatusValue\":\"SUCCESS\",\"pyID\":\"%BuildID%\"}" "%CallBackURL%"
g. Click **Save**.

14. If Jenkins is running on Linux, add the following post-build tasks. Use the dollar sign ($) instead of the percent sign (%) to access the environment variables:
a. Click **Add post-build action**, and then select **Post build task**.
b. In the **Log text** field, enter a unique string that for the message that is displayed in the build console output when a build fails, for example BUILD FAILURE**.**
c. In the **Script** field, enter curl --user $RMCREDENTIALS -H "Content-Type: application/json" -X POST --data "{\"jobName\":\"$JOB_NAME\",\"buildNumber\":\"$BUILD_NUMBER\",\"pyStatusValue\":\"FAIL\",\"pyID\":\"$BuildID\"}" "$CallBackURL"
d. Click **Add another task**.
e. In the **Log text** field, enter a unique string that for the message that is displayed in the build console output when a build is successful, for example BUILD SUCCESS**.**
f. In the **Script** field, enter curl --user $RMCREDENTIALS -H "Content-Type: application/json" -X POST --data "{\"jobName\":\"$JOB_NAME\",\"buildNumber\":\"$BUILD_NUMBER\",\"pyStatusValue\":\"SUCCESS\",\"pyID\":\"$BuildID\"}" "$CallBackURL"
g. Click **Save**.

Configuring and running pipelines with Deployment Manager 3.4.x

Use Deployment Manager to create continuous integration and delivery (CI/CD) pipelines, which automate tasks so that you can quickly deploy high-quality software to production.

Use Deployment Manager to create continuous integration and delivery (CI/CD) pipelines, which automate tasks and allow you to quickly deploy high-quality software to production.

On the orchestration server, release managers use the DevOps landing page to configure CI/CD pipelines for their Pega Platform applications. The landing page displays all the running and queued application deployments, branches that are to be merged, and reports that provide information about your DevOps environment such as key performance indicators (KPIs).

Note: These topics describe the features for the latest version of Deployment Manager 3.4.x.

- Configuring an application pipeline

  When you add a pipeline, you specify merge criteria and configure stages and steps in the continuous delivery workflow. For example, you can specify that a branch must be peer-reviewed before it can be merged, and you can specify that Pega unit tests must be run after a branch is merged and is in the QA stage of the pipeline.

- Manually starting a deployment in Deployment Manager

  You can start a deployment manually if you are not using branches and are working directly in rulesets. You can also start a deployment manually if you do not want deployments to start automatically when branches are merged.

- Starting a deployment in a distributed, branch-based environment

  If you are using Deployment Manager in a distributed, branch-based environment and using multiple pipelines per application, first export the branch to the source development system, and then merge it.

- Completing or rejecting a manual step

  If a manual step is configured on a stage, the deployment pauses when it reaches the step, and you can either complete it or reject it. For example, if a user was assigned a task and completed it, you can complete the task to continue the deployment. Deployment Manager also sends you an email when there is a manual step in the pipeline. You can complete or reject a step either within the pipeline or through email.

- Managing aged updates

  If your role has the appropriate permissions, you can manage aged updates in a number of ways, such as importing them, skipping the import, or manually deploying applications. Managing aged updates gives you more flexibility in how you deploy application changes.

- Configuring settings to automatically apply schema changes

  You can configure settings to automatically deploy schema changes that are in an application package that is to be deployed on candidate systems. Configure these settings so that you do not have to apply schema changes if you do not have the privileges to deploy them.

- Pausing and resuming deployments

  When you pause a deployment, the pipeline completes the task that it is running, and stops the deployment at the next step. Your user role determines if you can pause a deployment.

- Stopping a deployment

  If a user has privileges to abort, they can do so to prevent a deployment from moving through a pipeline.

- Managing a deployment that has errors

  If a deployment has errors, the pipeline stops processing on it. You can perform actions such as rolling back the deployment or skipping the step on which the error occurred.

- Viewing branch status

  You can view the status of all the branches that are in your pipeline. For example, you can see whether a branch was merged in a deployment and when it was merged.

- Viewing deployment logs

  View logs for a deployment to see the completion status of operations, for example, when a deployment moves from staging to production. When the Deploy task runs, the application package is imported in to the candidate system. By default, logs record all the new rule and data instances and all the updated rule and data instances that are in this application package. You can disable the logging of such rule and data types and can change the logging level to control which events are displayed in the log.

- Viewing deployment reports for a specific deployment

  Deployment reports provide information about a specific deployment. You can view information such as the number of tasks that you configured on a deployment that have been completed and when each task started and ended. If there were schema changes on the deployment, the report displays the schema changes.

- Viewing reports for all deployments

  Reports provide a variety of information about all the deployments in your pipeline. For example, you can view the frequency of new deployments to production.

- Deleting a pipeline

  If your role has the appropriate permission, you can delete a pipeline. When you delete a pipeline, its associated application packages are not removed from the repositories that the pipeline is configured to use.

- Viewing, downloading, and deleting application packages

  You can view, download, and delete application packages in repositories that are on the orchestration server. If you are using Deployment Manager on Pega Cloud Services, application packages that you have deployed to cloud repositories are stored on Pega Cloud Services. To manage your cloud storage space, you can download and permanently delete the packages.

Configuring an application pipeline

When you add a pipeline, you specify merge criteria and configure stages and steps in the continuous delivery workflow. For example, you can specify that a branch must be peer-reviewed before it can be merged, and you can specify that Pega unit tests must be run after a branch is merged and is in the QA stage of the pipeline.

You can create multiple pipelines for one version of an application. For example, you can use multiple pipelines in the following scenarios:

- To move a deployment to production separately from the rest of the pipeline. You can then create a pipeline that has only a production stage or development and production stages.
- To use parallel development and hotfix life cycles for your application.

- Adding a pipeline on Pega Cloud Services

  If you are using Pega Cloud Services, when you add a pipeline, you specify details such as the application name and version for the pipeline. Many fields are populated by default, such as the URL of your development system and product rule name and version.

- Adding a pipeline on premises

  When you add a pipeline on premises, you define all the stages and tasks that you want to do on each system. For example, if you are using branches, you can start a build when a branch is merged. If you are using a QA system, you can run test tasks to validate application data.

- Modifying application details

  You can modify application details, such as the product rule that defines the content of the application that moves through the pipeline.

- Modifying URLs and authentication profiles

You can modify the URLs of your development and candidate systems and the authentication profiles that are used to communicate between those systems and the orchestration server.

- Modifying repositories

  You can modify the development and production repositories through which the product rule that contains application contents moves through the pipeline. All the generated artifacts are archived in the development repository, and all the production-ready artifacts are archived in the production repository.

- Configuring Jenkins server information

  If you are using a Jenkins step, specify details about the Jenkins server such as its URL.

- Specifying merge options for branches

  If you are using branches in your application, specify options for merging branches into the base application.

- Modifying stages and tasks in the pipeline

  You can modify the stages and the tasks that are performed in each stage of the pipeline. For example, you can skip a stage or add tasks such as Pega unit testing to be done on the QA stage.

Adding a pipeline on Pega Cloud Services

If you are using Pega Cloud Services, when you add a pipeline, you specify details such as the application name and version for the pipeline. Many fields are populated by default, such as the URL of your development system and product rule name and version.

To add a pipeline on Pega Cloud Services, do the following steps:

1. In the Designer Studio footer, click **Deployment Manager**.
2. Click **Add pipeline**.
3. Specify the details of the application for which you are creating the pipeline.
   a. To change the URL of your development system, which is populated by default with your development system URL, in the **Development environment** field, press the Down arrow key and select the URL.
   This is the system on which the product rule that defines the application package that moves through the repository is located.
   b. In the **Application** field, press the Down arrow key and select the name of the application.
   c. In the **Version** field, press the Down arrow key and select the application version.
   d. Click the **Access group** field and select the access group for which pipeline tasks are run.
   This access group must be present on all the candidate systems and have at least the sysadmin4 role. Ensure that the access group is correctly pointing to the application name and version that is configured in the pipeline.
   e. In the **Pipeline** name field, enter a unique name for the pipeline.
4. Click **Create**. **Result:** The system adds tasks, which you cannot delete, to the pipeline that are required to successfully run a workflow, for example, Deploy and Generate Artifact. For Pega Cloud Services, it also adds mandatory tasks that must be run on the pipeline, for example, the Check guardrail compliance task and Verify security checklist task.
5. Add tasks that you want to perform on your pipeline, such as Pega unit testing. For more information, see Modifying stages and tasks in the pipeline.

Adding a pipeline on-premises

When you add a pipeline on premises, you define all the stages and tasks that you want to do on each system. For example, if you are using branches, you can start a build when a branch is merged. If you are using a QA system, you can run test tasks to validate application data.

To add a pipeline on premises, complete the following steps:

1. In the Designer Studio footer, click **Deployment Manager**.
2. Click **Add pipeline**.
3. Specify the details of the application for which you are creating the pipeline.
   a. In the **Development environment** field, enter the URL of the development system. This is the system on which the product rule that defines the application package that moves through the repository is located.
   b. In the **Application** field, press the Down arrow key and select the name of the application.
   c. In the **Version** field, press the Down arrow key and select the application version.
   d. In the **Access group** field, press the Down arrow key and select the access group for which pipeline tasks are run.
   This access group must be present on all the candidate systems and have at least the sysadmin4 role.
   e. In the **Pipeline name** field, enter a unique name for the pipeline.
   f. In the **Product rule** field, enter the name of the product rule that defines the contents of the application.
   g. In the **Version** field, enter the product rule version.
4. To configure dependent applications, click **Dependencies**.
   a. Click **Add**.
   b. In the **Application name** field, press the Down arrow key and select the application name.

c. In the **Application version** field, press the Down arrow key and select the application version.
d. In the **Repository name** field, press the Down arrow key and select the repository that contains the production-ready artifact of the dependent application.
If you want the latest artifact of the dependent application to be automatically populated, ensure that the repository that contains the production-ready artifact of the dependent application is configured to support file updates.
e. In the **Artifact** name field, press the Down arrow key and select the artifact.

For more information about dependent applications, see Product rules: Listing product dependencies for Pega-supplied applications.

a. Click **Next**.

5. In the **Environment details** section, in the **Stages** section, specify the URL of each candidate system and the authentication profile that each system uses to communicate with the orchestration system.
   a. In the **Environments** field for the system, press the Down arrow key and select the URL of the system.
   b. If you are using your own authentication profiles, in the **Authentication** field for the system, press the Down arrow key and select the authentication profile that you want to communicate from the orchestration server to the system.

   By default, the fields are populated with the DMAppAdmin authentication profile.
6. In the **Artifact management** section, specify the development and production repositories through which the product rule that contains application contents moves through the pipeline.
7. In the **Development repository** field, press the Down arrow key and select the development repository.
8. In the **Production repository** field, press the Down arrow key and select the production repository.
9. In the **External orchestration server** section, if you are using a Jenkins step in a pipeline, specify the Jenkins details.
   a. In the **URL** field, enter the URL of the Jenkins server.
   b. In the **Authentication profile** field, press the Down arrow key and select the authentication profile on the orchestration server that specifies the Jenkins credentials to use for Jenkins jobs.
10. Click **Next**.
11. If you are using branches in your application, in the **Merge policy** section, specify merge options. Do one of the following actions:
    - To merge branches into the highest existing ruleset in the application, click **Highest existing ruleset**.
    - To merge branches into a new ruleset, click **New ruleset**.
12. In the **Password** field, enter the password that locks the rulesets on the development system.
13. Click **Next**. **Result:** The system adds tasks, which you cannot delete, to the pipeline that are required to successfully run a workflow, for example, Deploy and Generate Artifact. The system also adds other tasks to enforce best practices such as Check guardrail compliance and Verify security checklist.
14. To specify that a branch must meet a compliance score before it can be merged:
    a. In the **Merge criteria** pane, click **Add task**.
    b. From the **Task** list, select **Check guardrail compliance**.
    c. In the **Weighted compliance score** field, enter the minimum required compliance score.
    d. Click **Submit**.

    For more information about compliance scores, see Compliance score logic
15. To specify that a branch must be reviewed before it can be merged:
    a. In the **Merge criteria** pane, click **Add task**.
    b. From the **Task** list, select **Check review status**.
    c. Click **Submit**.

    For more information about branch reviews, see Branch reviews.
16. To start a deployment automatically when a branch is merged, click the **Trigger deployment on merge check box**. Do not select this check box if you want to manually start deployments.
    For more information, see Manually starting a deployment.
17. Clear a check box for a deployment life cycle stage to skip it.
18. In the **Continuous Deployment** section, specify the tasks to be performed during each stage of the pipeline. See the following topics for more information:
    - Adding the Pega unit testing task
    - Adding the Jenkins task
    - Adding the Check guardrail compliance score task
    - Adding the Verify security checklist task
    - Modifying the Approve for production task
19. Click **Finish**.

- Running Pega unit tests by adding the Run Pega unit tests task

  If you use Pega unit tests to validate application data, add the Pega unit testing task on the pipeline stage where you want to run it. For example, you can run Pega unit tests on a QA system.

- Running Jenkins steps by adding the Run Jenkins step task

  If you are using Jenkins to perform tasks in your pipeline, you can add the Run Jenkins step to the stage on which you want it to run. If you have configured the Jenkins OrchestratorURL and PipelineID parameters, when this task fails, the pipeline stops running. For more information about configuring these parameters, see .

- Continuing or stopping a deployment by adding the Perform manual step task

  Use manual steps so that users must take an action before a pipeline deployment can continue. Users can either accept the task to continue the deployment or reject the task to stop it.

- Specifying that an application meet a compliance score by adding the Check guardrail compliance score task

  You can use the Check guardrail compliance score task so that an application must meet a compliance score for the deployment to continue. The default value is 97, which you can modify.

- Ensuring that the Application Security Checklist is completed by adding the Verify security checklist task

  For your pipeline to comply with security best practices, you can add a task to ensure that all the steps in the Application Security Checklist are performed. For customers on Pega Platform 8.4 and above, a new Security Checklist API is available to provide an automated security configuration assessment. Both candidate and orchestrator environments should be on or above Deployment Manager 4.8 to utilize this functionality.

- Modifying the Approve for production task

  The Approve for production task is added to the stage before production. Use this task if you want a user to approve application changes before those changes are send to production.

Adding the Pega unit testing task

If you use Pega unit tests to validate application data, add the Pega unit testing task on the pipeline stage where you want to run it. For example, you can run Pega unit tests on a QA system.

To run Pega unit tests for either the pipeline application or for an application that is associated with an access group, do the following steps:

1. Do one of the following actions:
   - Click a manually added task, click the **More** icon, and then click either **Add task above** or **Add task below**.
   - Click **Add task** in the stage.
2. Select **Pega unit testing** from the **Task** list.
3. Do one of the following actions:

   - To run all the Pega unit tests that are in a Pega unit suite for the pipeline application, in the **Test Suite ID** field, enter the *pxInsName* of the test suite.

     You can find this value in the XML document that comprises the test suite by clicking, in Designer Studio, **Actions XML** on the **Edit Test Suite** form. If you do not specify a test suite, all the Pega unit tests for the pipeline application are run.

   - To run all the Pega unit tests for an application that is associated with an access group, in the **Access Group** field, enter the access group.

     For more information about creating Pega unit tests, see Creating Pega unit test cases.

4. Click **Submit**.
5. Continue configuring your pipeline. For more information, see one of the following topics:
   - Adding a pipeline on-premises
   - Modifying stages and tasks in the pipeline

Adding the Jenkins task

If you are using Jenkins to perform tasks in your pipeline, you can add the Jenkins task to the stage on which you want it to run.

Do the following steps.

1. Do one of the following actions:
   - Click a manually added task, click the **More** icon, and then click either **Add task above** or **Add task below**.
   - Click **Add task** in the stage.
2. In the **Job name** field, enter the name of the Jenkins job (which is the name of the Jenkins deployment) that you want to run.
3. In the **Token** field, enter the Jenkins authentication token.
4. In the **Parameters** field, enter parameters, if any, to send to the Jenkins job.
5. Click **Submit**.
6. Continue configuring your pipeline. For more information, see one of the following topics:
   - Adding a pipeline on premises
   - Modifying stages and tasks in the pipeline

Adding the manual step task

Use manual steps so that users must take an action before a pipeline deployment can continue. Users can either accept the task to continue the deployment or reject the task to stop it.

To add a manual step that a user must perform in the pipeline, do the following steps:

1. Do one of the following actions:
   - Click a manually added task, click the **More** icon, and then click either **Add task above** or **Add task below**.
   - Click **Add task** in the stage.
2. From the **Task** list, select **Manual**.
3. In the **Job name** field, enter text that describes the action that you want the user to take.
4. In the **Assigned to** field, press the Down arrow key and select the operator ID to assign the task to.
5. Click **Submit**.
6. Continue configuring your pipeline. For more information, see one of the following topics:
   - Adding a pipeline on-premises
   - Modifying stages and tasks in the pipeline

Adding the Check guardrail compliance score task

You can use the Check guardrail compliance score task so that an application must meet a compliance score for the deployment to continue. The default value is 95, which you can modify.

To specify that an application must meet a compliance score, do the following steps:

1. Do one of the following actions:
   - Click a manually added task, click the **More** icon, and then click either **Add task above** or **Add task below**.
   - Click **Add task** in the stage.
2. From the **Task** list, select **Check guardrail compliance**.
3. In the **Weighted compliance score** field, enter the minimum required compliance score.
4. Continue configuring your pipeline. For more information, see one of the following topics:
   - Adding a pipeline on premises
   - Modifying stages and tasks in the pipeline

Adding the Verify security checklist task

For your pipeline to comply with security best practices, you can add a task so that to ensure that all the steps in Application Security Checklist are performed.

You must log in to the system for which this task is configured, and then mark all the tasks in the Application Security checklist as completed for the pipeline application. For more information about completing the checklist, see Preparing your application for secure deployment.

Do the following steps:

1. Do one of the following actions:
   - Click a manually added task, click the **More** icon, and then click either **Add task above** or **Add task below**.
   - Click **Add task** in the stage.
2. From the **Task** list, select **Verify Security checklist**.
3. Click **Submit**.
4. Continue configuring your pipeline. For more information, see one of the following topics:
   - Adding a pipeline on-premises
   - Modifying stages and tasks in the pipeline

Modifying the Approve for production task

The Approve for production task is added to the stage before production. Use this task if you want a user to approve application changes before those changes are send to production.

Do the following steps:

1. Click the **Info** icon.
2. In the **Job name** field, enter a name for the task.
3. In the **Assign to** field, press the Down arrow key and select the user who approves the application for production. An email is sent to this user, who can approve or reject application changes from within the email.
4. Click **Submit**.
5. Continue configuring your pipeline. For more information, see one of the following topics:
   - Adding a pipeline on-premises
   - Modifying stages and tasks in the pipeline

Modifying application details

You can modify application details, such as the product rule that defines the content of the application that moves through the pipeline.

Do the following steps:

1. If the pipeline is not open, in the navigation pane, click **Pipelines** , and then click the name of the pipeline.
2. Click **Actions Application details** .

3. In the **Development environment** field, enter the URL of the development system, which is the system on which the product rule that defines the application package that moves through the repository is located.
4. In the **Version** field, press the Down arrow key and select the application version.
5. In the **Product rule** field, enter the product rule that defines the contents of the application.
6. In the **Version** field, enter the product rule version.
7. If you are using a separate product rule to manage test cases, in the **Application test cases** section, complete the following steps:
    a. To deploy test cases, select the **Deploy test applications** check box.
    b. In the Test application field, enter the name of the test application.
    c. In the **Version** field, enter the version of the test case product rule.
    d. In the **Access group** field, enter the access group for which test cases are run.
    e. In the **Product rule** field, enter the name of the test case product rule.
    f. From the **Deploy until** field, select the pipeline stage until which the test case product rule will be deployed.
    Note: When you use separate product rules for test cases and run a pipeline, the Run Pega unit tests, Enable test coverage, and Verify test coverage tasks are run for the access group that is specified in this section.

    For the Run Pega scenario tests task, the user name that you provide should belong to the access group that is associated with the test application.

8. If the application depends on other applications, in the **Dependencies** section, add those applications.
    a. Click **Add**.
    b. In the **Application name** field, press the Down arrow key and select the application name.
    c. In the **Application version** field, press the Down arrow key and select the application version.
    d. In the **Repository name** field, press the Down arrow key and select the repository that contains the production-ready artifact of the dependent application. If you want the latest artifact of the dependent application to be automatically populated, ensure that the repository that contains the production-ready artifact of the dependent application is configured to support file updates.
    e. In the **Artifact name** field, press the Down arrow key and select the artifact.

    For more information about dependent applications, see Listing product dependencies.
9. Click **Save**.

Modifying URLs and authentication profiles

You can modify the URLs of your development and candidate systems and the authentication profiles that are used to communicate between those systems and the orchestration server.

Do the following steps:

1. If the pipeline is not open, in the navigation pane, click **Pipelines** , and then click the name of the pipeline.
2. Click **Actions Environment Details** .
3. In the **Environments** field for the system, press the Down arrow key and select the URL of the system.
4. In the **Authentication** field for the system, press the Down arrow key and select the authentication profile that you want to communicate from the orchestration server to the system.
5. Click **Save**.

Modifying repositories

You can modify the development and production repositories through which the product rule that contains application contents moves through the pipeline. All the generated artifacts are archived in the Development repository, and all the production-ready artifacts are archived in the Production repository.

You do not need to configure repositories if you are using Pega Cloud Services; you can use different repositories other than the default ones that are provided.

Do the following steps:

1. If the pipeline is not open, in the navigation pane, click **Pipelines** , and then click the name of the pipeline.
2. Click **Actions Artifact Management** .
3. If you are using Deployment Manager on premises, or on Pega Cloud Services with default repositories, complete the following tasks:
    a. In the **Application repository** section, in the **Development repository** field, press the Down arrow key and select the development repository
    b. In the **Production repository** field, press the Down arrow key and select the production repository.
4. If you are using Deployment Manager on Pega Cloud Services and want to use different repositories other than the default repositories, complete the following tasks:
    a. In the **Artifact repository** section, click **Yes**.
    b. In the **Development repository** field, press the Down arrow key and select the development repository.
    c. In the **Production repository** field, press the Down arrow key and select the production repository.
5. Click **Save**.

Configuring Jenkins server information

If you are using a Jenkins step, specify details about the Jenkins server such as its URL.

Do the following steps:

1. If the pipeline is not open, in the navigation pane, click **Pipelines** , and then click the name of the pipeline.
2. Click **Actions External orchestration server** .
3. Click the **Jenkins** icon, and then click **OK**.
4. In the **URL** field, enter the URL of the Jenkins server.
5. In the **Authentication** profile field, press the Down arrow key and select the authentication profile on the orchestration server that specifies the Jenkins credentials to use for Jenkins jobs.
6. Click **Save**.

Specifying merge options for branches

If you are using branches in your application, specify options for merging branches into the base application.

Do the following steps:

1. If the pipeline is not open, in the navigation pane, click **Pipelines** , and then click the name of the pipeline.
2. Click **Actions Merge policy** .
3. Do one of the following actions:
   - To merge branches into the highest existing ruleset in the application, click **Highest existing ruleset**.
   - To merge branches into a new ruleset, click **New ruleset**.
   a. In the **Password** field, enter the password that locks the rulesets on the development system.
4. Click **Save**.

Modifying stages and tasks in the pipeline

You can modify the stages and the tasks that are performed in each stage of the pipeline. For example, you can skip a stage or add tasks such as Pega unit testing to be done on the QA stage.

Do the following steps:

1. If the pipeline is not open, in the navigation pane, click **Pipelines** , and then click the name of the pipeline.
2. Click **Pipeline model**.
3. To specify that a branch must meet a compliance score before it can be merged:
   a. In the **Merge criteria** pane, click **Add task**.
   b. From the **Task** list, select **Check guardrail compliance**.
   c. In the **Weighted compliance score** field, enter the minimum required compliance score.
   d. Click **Submit**.
   For more information about compliance scores, see Compliance score logic.
4. To specify that a branch must be reviewed before it can be merged:
   a. In the **Merge criteria** pane, click **Add task**.
   b. From the **Task** list, select **Check review status**.
   c. Click **Submit**.
   For more information about branch reviews, see Branch reviews.
5. To start a deployment automatically when a branch is merged, select the **Trigger deployment on merge** check box. Do not select this check box if you want to manually start a deployment.
   For more information, see Manually starting a deployment.
6. Clear a check box for a deployment life cycle stage to skip it.
7. In the Continuous Deployment section, specify the tasks to be performed during each stage of the pipeline. See the following topics for more information:
   - Adding the Pega unit testing task
   - Adding the Jenkins task
   - Adding the Check guardrail compliance score task
   - Adding the Verify security checklist task
   - Modifying the Approve for production task
8. Click **Finish**.

Manually starting a deployment

You can start a deployment manually if you are not using branches and are working directly in rulesets. You can also start a deployment manually if you do not want deployments to start automatically when branches are merged.

Do the following steps:

1. If you do not want deployments to start automatically when branches are merged:
   1. If the pipeline is not open, in the navigation pane, click **Pipelines Application pipelines**
   2. Click **Pipeline model**.
   3. Select the **Trigger deployment on merge** check box.
2. Do one of the following actions:
   - If the pipeline that you want to start is open, click **Start deployment**.

    - Click **Pipelines** , and then click **Start deployment** for the pipeline that you want to start.
3. In the **Start deployment** dialog box, start a new deployment or deploy an existing application by completing one of the following actions:
    - To deploy a new application package, go to step 3.
    - To deploy an application that is on a cloud repository, go to step 4.
4. To start a deployment and deploy a new application package, do the following steps:
    a. Click **Generate new artifact**.
    a. In the **Deployment name** field, enter the name of the deployment.
    b. Click **Deploy**.
    c. Go to step 5.
5. To start a deployment and deploy an application package that is on a cloud repository, do the following steps:
    a. Click **Deploy an existing artifact**.
    b. In the **Deployment name** field, enter the name of the deployment.
    c. In the **Select a repository** field, press the Down arrow key and select the repository.
    d. In the **Select an artifact** field, press the Down arrow key and select the application package.
6. Click **Deploy**.

Starting a deployment in a distributed, branch-based environment

If you are using Deployment Manager in a distributed, branch-based environment and using multiple pipelines per application, first export the branch to the source development system, and then merge it.

Do the following steps:

1. On the remote development system, package the branch. For more information, see Packaging a branch.
2. Export the branch.
3. On the source development system, import the branch by using the Import wizard. For more information, see Importing a file by using the Import wizard.
4. On the source development system, start a deployment by using the Merge Branches wizard. For more information, see Submitting a branch into a pipeline.
   If you are using one pipeline per application, you can publish a branch to start the merge. For more information, see Publishing a branch to a repository.

Completing or rejecting a manual step

If a manual step is configured on a stage, the deployment pauses when it reaches the step, and you can either complete it or reject it. For example, if a user was assigned a task and completed it, you can complete the task to continue the deployment. Deployment Manager also sends you an email when there is a manual step in the pipeline. You can complete or reject a step either within the pipeline or through email.

Deployment Manager also generates a manual step if there are schema changes in the application package that the release manager must apply. For more information, see Schema changes in application packages.

To complete or reject a manual step within the deployment, do the following steps:

1. To complete or reject a manual step from within an email, click either **Accept** or **Reject**.
2. To complete or reject a manual step in the pipeline,
    a. In the Designer Studio footer, click **Deployment Manager**.
    b. Click a pipeline.
    c. Accept or reject the step by doing one of the following actions:
    - To resolve the task so that the deployment continues through the pipeline, click **Complete**.
    - To reject the task so that the deployment does not proceed, click **Reject**.

Managing aged updates

You can manage aged updates in a number of ways such as importing them, skipping the import, or manually deploying applications. Managing aged updates gives you more flexibility in how you deploy application changes.

Do the following steps::

1. In the Designer Studio footer, click **Deployment Manager**.
2. Click the pipeline.
3. Click **View aged updates** to view a list of the rules and data instances, which are in the application package, that are older than the instances that are on the system.
4. Click the **More** icon and do one of the following actions:
    - To import the older rule and data instances that are in the application package into the system, which overwrites the newer versions that are on the system, click **Overwrite aged updates**.
    - To skip the import, click **Skip aged updates**.
    - To manually deploy the package from the Import wizard on the system, click **Deploy manually and resume**. Deployment Manager does not run the Deploy step on the stage.

- Understanding aged updates

An aged update is a rule or data instance in an application package that is older than an instance that is on a system to which you want to deploy the application package. By being able to import aged updates, skip the import, or manually deploy your application changes, you now have more flexibility in determining the rules that you want in your application and how you want to deploy them.

Understanding aged updates

An aged update is a rule or data instance in an application package that is older than an instance that is on a system to which you want to deploy the application package. By being able to import aged updates, skip the import, or manually deploy your application changes, you now have more flexibility in determining the rules that you want in your application and how you want to deploy them.

For example, you can update a dynamic system setting on a quality assurance system, which has an application package that contains the older instance of the dynamic system setting. Before Deployment Manager deploys the package, the system detects that the version of the dynamic system setting on the system is newer than the version in the package and creates a manual step in the pipeline.

Configuring settings to automatically apply schema changes

You can configure settings to automatically deploy schema changes that are in an application package that is to be deployed on candidate systems. Configure these settings so that you do not have to apply schema changes if you do not have the privileges to deploy them.

Do the following steps:

1. On the candidate system, in Pega Platform, set the *AutoDBSchemaChanges* dynamic system setting to true to enable schema changes at the system level.
   a. In Designer Studio, search for *AutoDBSchemaChanges*.
   b. In the dialog box that appears for the search results, click **AutoDBSchemaChanges**.
   c. On the **Settings** tab, in the **Value** field, enter true.
   d. Click **Save**.
2. Add the SchemaImport privilege to your access role to enable schema changes at the user level. For more information, see Specifying privileges for an Access or Role to Object rule.

Result: These settings are applied sequentially. If the *AutoDBSchemaChanges* dynamic system setting is set to false, you cannot deploy schema changes, even if you have the SchemaImport privilege.

For more information about the database/AutoDBSchemaChanges dynamic system setting, see Importing rules and data by using a direct connection to the database.

- Understanding schema changes in application packages

  If an application package that is to be deployed on candidate systems contains schema changes, the Pega Platform orchestration server checks the candidate system to verify that you have the required privileges to deploy the schema changes. One of the following results occurs:

Understanding schema changes in application packages

If an application package that is to be deployed on candidate systems contains schema changes, the Pega Platform orchestration server checks the candidate system to verify that you have the required privileges to deploy the schema changes. One of the following results occurs:

- If you have the appropriate privileges, schema changes are automatically applied to the candidate system, the application package is deployed to the candidate system, and the pipeline continues.
- If you do not have the appropriate privileges, Deployment Manager generates an SQL file that lists the schema changes and sends it to your email address. It also creates a manual step, pausing the pipeline, so that you can apply the schema changes. After you complete the step, the pipeline continues. For more information about completing a step, see Completing or rejecting a manual step.

You can also configure settings to automatically deploy schema changes so that you do not have to manually apply them if you do not have the required privileges. For more information, see Configuring settings to automatically deploy schema changes.

Pausing a deployment

When you pause a deployment, the pipeline completes the task that it is running, and stops the deployment at the next step.

To pause a deployment:

1. If the pipeline is not open, in the navigation pane, click **Pipelines** .
2. Click the pipeline.

3. Click **Pause**.

Stopping a deployment

Stop a deployment to discontinue processing.

To stop a deployment:

1. If the pipeline is not open, in the navigation pane, click **Pipelines** .
2. Click the **More** icon, and then click **Abort**.

Performing actions on a deployment that has errors

If a deployment has errors, the pipeline stops processing on it. You can perform actions on it, such as rolling back the deployment or skipping the step on which the error occurred.

Do the following steps:

1. If the pipeline is not open, in the navigation pane, click **Pipelines** .
2. Click the **More** icon, and then do one of the following actions:
    - To resume running the pipeline from the task, click **Resume from current task**.
    - To skip the step and continue running the pipeline, click **Skip current task and continue** .
    - To roll back to an earlier deployment, click **Rollback**.
    - To stop running the pipeline, click **Abort**.

Viewing branch status

You can view the status of all the branches that are in your pipeline. For example, you can see whether a branch was merged in a deployment and when it was merged.

Do the following steps:

1. Click **Deployment Manager** in the Designer Studio footer.
2. Click a pipeline.
3. Click **Actions View branches** .

Viewing deployment logs

View logs for a deployment to see the completion status of operations, for example, when a data simulation is moved to the simulation environment. You can change the logging level to control which events are displayed in the log.

For example, you can change logging levels of your deployment from INFO to DEBUG for troubleshooting purposes. For more information, see Logging Level Settings tool.

Do the following steps:

1. .Click **Deployment Manager** in the Designer Studio footer.
2. Click a pipeline.
3. Click the **Gear** icon for the deployment for which you want to view the log file.
4. Click **View log**.

Viewing deployment reports for a specific deployment

Deployment reports provide information about a specific deployment. You can view information such as the number of tasks that you configured on a deployment that have been completed and when each task started and ended.

Do the following steps:

1. Click **Deployment Manager** in the Designer Studio footer.
2. Click a pipeline.
3. Click the **Gear** icon for the deployment for which you want to view the deployment report.
4. Click **View report**.

Viewing reports for all deployments

Reports provide a variety of information about all the deployments in your pipeline. For example, you can view the frequency of new deployments to production.

You can view the following key performance indicators (KPI):

- Deployment Success – Percentage of deployments that are successfully deployed to production
- Deployment Frequency – Frequency of new deployments to production

- Deployment Speed – Average time taken to deploy to production
- Start frequency – Frequency at which new deployments are triggered
- Failure rate – Average number of failures per deployment
- Merges per day – Average number of branches that are successfully merged per day

1. In the Designer Studio footer, click **Deployment Manager**.
2. Click a pipeline.
3. Click **Action Reports** .

Deleting an application pipeline

When you delete a pipeline, its associated application packages are not removed from the repositories that the pipeline is configured to use.

Do the following steps:

1. In the Designer Studio footer, click **Deployment Manager**.
2. Click the **Delete** icon for the pipeline that you want to delete.
3. Click **Submit**.

Viewing, downloading, and deleting application packages

You can view, download, and delete application packages in repositories that are on the orchestration server. If you are using Deployment Manager on Pega Cloud Services, application packages that you have deployed to cloud repositories are stored on Pega Cloud Services. To manage your cloud storage space, you can download and permanently delete the packages.

If you are using a separate product rule to manage a test application, the name of the product rule is the same as that of the product rule with _Tests appended to it.

Do the following steps:

1. In the Designer Studio footer, click **Deployment Manager**.
2. Click the pipeline for which you want to download or delete packages.
3. Click **Actions Browse artifacts** .
4. Click either **Development Repository** or **Production Repository**.
5. To download a package, click the package, and then save it to the appropriate location.
6. To delete a package, select the check boxes for the packages that you want to delete and then click **Delete**.

Automating deployment pipelines using Open DevOps solutions

Use DevOps practices such as continuous integration and continuous delivery to quickly move application changes from development, through testing, and to deployment. Use Pega Platform tools and common third-party tools to implement DevOps.

You can set up a continuous integration and delivery (CI/CD) pipeline that uses a Pega repository in which you can store and test software and a third-party automation server such as Jenkins that starts jobs and performs operations on your software. Use a CI/CD pipeline to quickly detect and resolve issues before deploying your application to a production environment.

For example, you can configure an automation server with REST services to automatically merge branches after you publish them to a Pega repository. You can also configure Jenkins to create branch reviews, run PegaUnit tests, and return the status of a merge.

- Downloading and installing prpcServiceUtils

  Download and install prpcServiceUtils so that you can use it with Jenkins deploy applications.

- Using prpcServiceUtils and Jenkins for automated application deployment

  You can use Jenkins to automate exporting and importing Pega Platform applications. Download the prpcServiceUtils command-line tool and configure Jenkins to export or import archives. You can use a single Jenkins build job to both export and import an application archive, or you can create separate jobs for each task.

- Running test cases and suites with the Execute Tests service

  You can use the Execute Tests service (REST API) to validate the quality of your code after every build is created by running unit test cases that are configured for the application.

Downloading and installing prpcServiceUtils

Download and install prpcServiceUtils so that you can use it with Jenkins deploy applications.

To download and install prpcServiceUtils, do the following steps:

1. Download the **prpcServiceUtils.zip** file onto the Jenkins server. The package is available for download on the Marketplace here.
2. Extract the files onto any location to which Jenkins has access.

- Configuring the Jenkins build environment

Using prpcServiceUtils and Jenkins for automated application deployment

You can use Jenkins to automate exporting and importing Pega Platform applications. Download the prpcServiceUtils command-line tool and configure Jenkins to export or import archives. You can use a single Jenkins build job to both export and import an application archive, or you can create separate jobs for each task.

For more information about prpcServiceUtils for service-enabled scripting, see Using service-enabled scripting and the prpcServiceUtils tool.

Ensure that your system includes the following items:

- Jenkins 1.651.1 or later
- Jenkins Plugins:
  - Ant Plugin
  - Environment Injector Plugin
  - Build with Parameters Plugin
- Ant version 1.9 or later
- JDK version 1.7 or later

- Configuring the Jenkins build environment

  Configure your build environment to call the prpcServiceUtils.bat or prpcServiceUtils.sh script and pass parameters to import or export the RAP.

- Configuring the Jenkins project

  Configure your build environment to call the prpcServiceUtils.bat or prpcServiceUtils.sh script and pass parameters to import or export the RAP:

- Configuring Jenkins in 5.1.x

  If you are using a Run Jenkins step task in your pipeline, configure Jenkins so that it can communicate with the orchestration server.

- Adding build steps to import or export the archive

  You can enter build steps to import an archive or export an archive, or you can do both in one job.

- Importing or exporting the archive by running the Jenkins job

  Run a Jenkins job to import or export the application archive.

Configuring the Jenkins build environment

Configure your build environment to call the prpcServiceUtils.bat or prpcServiceUtils.sh script and pass parameters to import or export the RAP.

To configure the Jenkins build environment, do the following steps:

1. Verify that the following Jenkins plugins are installed:
   - Ant Plugin
   - Environment Injector Plugin
   - Build with Parameters Plugin
2. Open a web browser and navigate to the Jenkins server.
3. Click **Manage Jenkins**.
4. Click **Configure System**.
5. Configure the PEGA_HOME environment variable:
   a. In the **Global properties** section, select **Environmental variables**.
   b. In the name field, enter PEGA_HOME.
   c. In the **value** field, enter the location where you extracted the prpcServiceUtils.zip file.
   d. Click **Add**.
6. Click **Apply**, and then click **Save**.

- Configuring the Jenkins project

Configuring the Jenkins project

Configure your build environment to call the prpcServiceUtils.bat or prpcServiceUtils.sh script and pass parameters to import or export the RAP:

1. Complete one of the following actions:
   - Create a project if you have not already done so.
   - Open an existing project.
2. Click **Configure**.
3. Select **This build is parameterized**.
4. Click **Add Parameter** and create the parameters that Jenkins passes to the prpcServiceUtils tool:
   - To import or export a RAP, create the following parameters:

     ### Product parameters

Parameter name	Type	Default value
productName	String	The name of the RAP rule used to generate the archive
productVersion	String	The version number of the RAP rule

   - To import or export an application, create the following parameters:

     ### Application parameters

Parameter name	Type	Default value
applicationName	String	The name of the application
applicationVersion	String	The version number of the application.

5. Select **Prepare an environment for the run**.
6. In the **Properties Content** section, set the following properties:
   - SystemName=$BUILD_TAG
   - Source Code Management=None
   - Inject environment variables to the build process=Properties Content
7. In the **Properties Content** section, set the *ImportExistingInstances* property to one of the following values. The default is unset:
   - override

     For rules - If a rule with the same key exists in the system, but the rule resolution properties differ (for example, ruleset or version), replace the existing rule with the imported rule.

     For work - If a work object with the same key exists but belongs to a different application (for example, it has a different class hierarchy but same classgroup name and same ID prefix), replace the existing work object with the imported work object.

   - skip

     For rules - If a rule with the same key exists in the system, and the rule resolution properties differ, do not replace the existing rule.

     For work - If a work object with the same key exists but belongs to a different application, do not replace the existing work object.

   - unset: The import will fail if keys already exist either for rule instances that have different rule resolution properties or for work objects that belong to a different applications that use the same classgroup name.

8. Set the artifact directory where exported logs and files are downloaded In the following format: ARTIFACTS_DIR=<var>path to artifact directory</var>
   The default is the logs directory.
   Note: You can also set the directory later by specifying -artifactsDir when you run the batch file.
9. In the **Properties Content** section, enter the static properties in the format ParameterName=Value.
   - Source properties for export:

     ### Source properties for export

Parameter name	Default value
SourceHost	The host name and port number of the Pega Platform server from which to export the archive file.

Parameter name	Default value
SourceUser	The operator name. This operator must have export privileges.
SourcePassword	The operator password.

- Target properties for import:

Target properties for import

Parameter name	Default value
TargetHost	The host name and port number of the Pega Platform server that contains the archive file to import.
TargetUser	The operator name. This operator must have import privileges.
TargetPassword	The operator password.

10. Click **Save**.

- Adding build steps to import or export the archive

Configuring Jenkins in 5.1.x

If you are using a Run Jenkins step task in your pipeline, configure Jenkins so that it can communicate with the orchestration server.

1. On the orchestration server, create an authentication profile that uses Jenkins credentials.
   - If you are using a version of Jenkins earlier than 2.17.6, create an authentication profile on the orchestration server that specifies the credentials to use.
     1. Click **Create Security Authentication Profile** .
     2. Enter a name, and then click **Create and open**.
     3. In the **User name** field, enter Jenkins user ID.
     4. Click **Set password**, enter the Jenkins password, and then click **Submit**.
     5. Click the **Preemptive authentication** check box.
     6. Click **Save**.
     7. Go to step 4.

     For more information about configuring authentication profiles, see Creating an authentication profile.
   - If you are using Jenkins 2.17.6 or later and want to use an API token for authentication, go to step 2.
   - If you are using Jenkins 2.17.6 or later and want to use a Crumb Issuer for authentication, go to step 3.
2. If you are using Jenkins version 2.17.6 or later and want to use an API token for authentication, do the following steps:
   a. Log in to the Jenkins server.
   b. Click **People**, click the user who is running the Jenkins job, and then click **Configure API token** .
   c. Generate the API token.
   d. Create an authentication profile on the orchestration server by clicking **Create Security Authentication Profile** .
   e. In the **User name** field, enter the Jenkins user ID.
   f. Click **Set password**, enter the API token that you generated, and then click **Submit**.
   g. Click the **Preemptive authentication** check box.
   h. Click **Save**.
   i. Go to step 4.

   For more information about configuring authentication profiles, see Creating an authentication profile.
3. If you are using Jenkins version 2.17.6 or later and want to use a Crumb Issuer for authentication, do the following steps:
   a. Log in to the Jenkins server.
   b. Click **Manage Jenkins Manage Plugins** and select the check box for the **Strict Crumb Issuer** plug-in.
   c. Click **Manage Jenkins Configure Global Security** .
   d. In the **CSRF protection** section, in the **Crumb Issuer** list, select **Strict Crumb Issuer**.
   e. Click **Advanced**, and then clear the **Check the session ID** check box.
   f. Click **Save**.
   g. Create an authentication profile on the orchestration server by clicking **Create Security Authentication Profile** .
   h. In the **User name** field, enter the Jenkins user ID.
   i. Click **Set password**, enter the Jenkins password, and then click **Submit**.
   j. Click the **Preemptive authentication** check box.
   k. Click **Save**.
   l. Go to step 4.

   For more information about configuring authentication profiles, see Creating an authentication profile.

The following steps describe one of the ways to configure Jenkins jobs for the integration with Deployment Manager to work (using Jenkins Free Style Project as the option). This can be done in multiple different ways (e.g. with workflow plugin and groovy, or scripts in Ant, Gradle, etc.).

4. Install the Post build task plug-in.
5. Install the curl and jq commands on the Jenkins server. As mentioned above, these can be replaced with any other scripts as well.
6. Create a new freestyle project.
7. On the **General** tab, select the **This project is parameterized** check box.
8. Add the DeploymentID and CallBackURL parameters.
    a. Click **Add parameter**, and then select **String parameter**.
    b. In the **String** field, enter DeploymentID.
    c. Click **Add parameter**, and then select **String parameter**.
    d. In the **String** field, enter CallBackURL.
9. To add parameters that you can use in Run Jenkins step tasks in the pipeline, click **Add parameter**, select **String parameter**, and enter the string of the parameter. The system automatically populates these values in Jenkins tasks. You can add any of the following strings:
    a. If you are configuring Jenkins tasks in a merge pipeline, add any of the following strings:
        - **DeploymentID**: Deployment ID of the pipeline on which the task is triggered.
        - **DeploymentNumber**: Sequence number of the deployment.
        - **TaskID**: TaskID of the Jenkins task.
        - **CallBackURL**: URL to post the status of task.
        - **OrchestratorURL**: URL on which the Jenkins task is configured.
        - **PipelineName**: Pipeline name on which the Jenkins task is configured.
        - **PipelineID**: ID of the pipeline on which the Jenkins task is configured.
        - **ApplicationName**: Application name for which the pipeline is configured.
        - **ApplicationVersion**: Application version for which the pipeline is configured.
        - **RepositoryName**: Repository to publish the merged branch.
        - **BranchName**: Name of the branch for which the merge Jenkins task is configured.
        - **BranchFilePath**: File path to the branch artifact.
    b. If you are configuring Jenkins tasks in a deployment pipeline, add any of the following strings:
        - **DeploymentID**: Deployment ID of the pipeline on which the task is triggered.
        - **DeploymentNumber**: Sequence number of the deployment.
        - **TaskID**: TaskID of the Jenkins task.
        - **CallBackURL**: URL to post the status of task.
        - **OrchestratorURL**: URL on which the Jenkins task is configured.
        - **PipelineName**: Pipeline name on which the Jenkins task is configured.
        - **PipelineID**: ID of the pipeline on which the Jenkins task is configured.
        - **RepositoryName**: Repository to publish the merged branch.
        - **DeploymentArtifactName**: Artifact name that the Deploy task uses on the stage on which the Jenkins task is configured.
        - **ArtifactPath**: Full path to the artifact that the Deploy task uses.
        - **CurrentStage**: Name of the stage on which the Jenkins task is configured.
        - **CurrentStageURL**: URL of the system on which the Jenkins task is configured.
10. In the **Build Triggers** section, select the **Trigger builds** remotely check box.
11. In the **Authentication Token** field, select the token that you want to use when you start Jenkins jobs remotely.
12. In the **Build Environment** section, select the **Use Secret text(s) or file(s)** check box.
13. In the **Bindings** section, do the following actions:
    a. Click **Add**, and then select **User name and password (separated)**.
    b. In the **Username Variable** field, enter client_id.
    c. In the **Password Variable** field, enter client_secret.
    d. In the **Credentials** field, click **Specific credentials**.
    e. Click **Add**, and then select **Jenkins**.
    f. In the **Add credentials** dialog box, in the **Username** field, enter the client id configured in the oAuth client credentials on Deployment Manager.
    g. In the **Password** field, enter the client secret.
    h. Optionally enter a description, and click **Save**.
14. Add post-build tasks by doing one of the following actions:
    a. If Jenkins is running on Microsoft Windows, go to step 15.
    b. If Jenkins is running on Linux, go to step 16.
15. If Jenkins is running on Microsoft Windows, add the following post-build tasks:
    a. Click **Add post-build** action, and then select **Post build task**.
    b. In the **Post-Build Actions** section, in the **Log text** field, enter a unique string for the message that is displayed in the build console output when a build fails, for example BUILD FAILURE**.**
    c. In the **Script** field, enter @echo off for /F %%I in ('curl --insecure -d "client_id=%%client_id%%&client_secret=%%client_secret%%&grant_type=client_credentials" https://10.224.203.11:8443/prweb/PRRestService/oauth2/v1/token ^| jq -r .access_token') do set token=%%~I curl --insecure -H "Content-Type: application/json" -H "Accept: application/json" -H "Authorization:Bearer %token%" -k -X PUT --data "{\"taskStatus\":\"Resolved-Rejected\",\"errors\": [{\"errorMessage\":\"Jenkins job failed. Check here for logs: %BUILD_URL%}],\"taskInfo\":{\"outputParameters\":[{\"name\": \"BuildNumber\",\"type\": \"Text\",\"value\": \"%BUILD_NUMBER%\"},{\"name\": \"JenkinsBuildURL\",\"type\": \"Text\",\"value\": \"%BUILD_URL%\"}]}}" %CallBackURL%.
    d. Click **Add another task**.
    e. In the **Post-Build Actions** section, in the **Log text** field, enter a unique string for the message that is displayed in the build console output when a build is successful, for example BUILD SUCCESS**.**
    f. In the **Script** field, enter @echo off for /F %%I in ('curl --insecure -d

"client_id=%%client_id%%&client_secret=%%client_secret%%&grant_type=client_credentials" https://10.224.203.11:8443/prweb/PRRestService/oauth2/v1/token ^| jq -r .access_token') do set token=%%~I curl --insecure -H "Content-Type: application/json" -H "Accept: application/json" -H "Authorization:Bearer %token%" -k -X PUT --data "{\"taskStatus\":\"Resolved-Completed\",\"taskInfo\":{\"outputParameters\":[{\"name\": \"BuildNumber\",\"type\": \"Text\",\"value\": \"%BUILD_NUMBER%\"},{\"name\": \"JenkinsBuildURL\",\"type\": \"Text\",\"value\": \"%BUILD_URL%\"}]}}" %CallBackURL%

   g. Click **Save**.
   h. Go to step 17.
16. If Jenkins is running on Linux, add the following post-build tasks. Use the dollar sign ($) instead of the percent sign (%) to access the environment variables:
   a. Click **Add post-build action**, and then select **Post build task**.
   b. In the **Log text** field, enter a unique string that for the message that is displayed in the build console output when a build fails, for example BUILD FAILURE**.**
   c. In the **Script** field, enter export token=`curl --insecure -d "client_id=${client_id}&client_secret=${client_secret}&grant_type=client_credentials" https://10.224.203.11:8443/prweb/PRRestService/oauth2/v1/token | jq -r .access_token` curl --insecure -H "Content-Type: application/json" -H "Accept: application/json" -H "Authorization:Bearer ${token}" -k -X PUT --data "{\"taskStatus\":\"Resolved-Rejected\",\"errors\":[{\"errorMessage\":\"Jenkins job failed. Check here for logs: ${BUILD_URL}\"}],\"taskInfo\":{\"outputParameters\":[{\"name\": \"BuildNumber\",\"type\": \"Text\",\"value\": \"${BUILD_NUMBER}\"},{\"name\": \"JenkinsBuildURL\",\"type\": \"Text\",\"value\": \"${BUILD_URL}\"}]}}" $CallBackURL
   d. Click **Add another task**.
   e. In the **Log text** field, enter a unique string that for the message that is displayed in the build console output when a build is successful, for example BUILD SUCCESS**.**
   f. In the **Script** field, enter export token=`curl --insecure -d "client_id=${client_id}&client_secret=${client_secret}&grant_type=client_credentials" https://10.224.203.11:8443/prweb/PRRestService/oauth2/v1/token | jq -r .access_token` curl --insecure -H "Content-Type: application/json" -H "Accept: application/json" -H "Authorization:Bearer ${token}" -k -X PUT --data "{\"taskStatus\":\"Resolved-Completed\",\"taskInfo\":{\"outputParameters\":[{\"name\": \"BuildNumber\",\"type\": \"Text\",\"value\": \"${BUILD_NUMBER}\"},{\"name\": \"JenkinsBuildURL\",\"type\": \"Text\",\"value\": \"${BUILD_URL}\"}]}}" $CallBackURL
   g. Click **Save**.
   h. Go to step 17.
17. To stop a pipeline deployment if a Jenkins build fails, add a post-build script:
   a. Click **Add post-build action**, and then select **Post build task**.
   b. In the **Log text** field, enter a unique string for the message that is displayed in the build console output when a build fails, for example JENKINS BUILD FAILURE**.**
   c. In the **Script** field, enter (for Windows) for /F %%I in ('curl --insecure -d "client_id=%%client_id%%&client_secret=%%client_secret%%&grant_type=client_credentials" https://10.224.203.11:8443/prweb/PRRestService/oauth2/v1/token ^| jq -r .access_token') do set token=%%~I curl --insecure -H "Content-Type: application/json" -H "Accept: application/json" -H "Authorization:Bearer %token%" --data "{\"reasonForAbort\":\"Jenkins task failed\"}" -k -X PUT %OrchestratorURL%/PRRestService/DeploymentManager/v1/deployments/%DeploymentID%/abort
   d. In the **Script** field, enter (for Linux) export token=`curl --insecure -d "client_id=${client_id}&client_secret=${client_secret}&grant_type=client_credentials" https://10.224.203.11:8443/prweb/PRRestService/oauth2/v1/token | jq -r .access_token` curl --insecure -H "Content-Type: application/json" -H "Accept: application/json" -H "Authorization:Bearer ${token}" --data "{\"reasonForAbort\":\"Jenkins task failed\"}" -k -X PUT ${OrchestratorURL}/PRRestService/DeploymentManager/v1/deployments/${DeploymentID}/abort
   e. Click **Save**.

Adding build steps to import or export the archive

You can enter build steps to import an archive or export an archive, or you can do both in one job.

- Adding export build steps
- Adding import build steps

- Adding export build steps
- Adding import build steps

Adding export build steps

To add export steps to your build job, do the following steps:

1. Add an Invoke Ant build step:
   a. In the **Build** section, click **Add build step** and select **Invoke Ant**.
   b. n the **Targets** field, enter exportprops.
   c. In the **Build File** field, enter the path to the build file:
   - On Windows, enter the following path: $PEGA_HOME\samples\Jenkins-build.xml
   - On UNIX, enter the following path: $PEGA_HOME/scripts/samples/jenkins/Jenkins-build.xml
2. Add a build step to run either prpcServiceUtils.bat or prpcServiceUtils.sh.
   - If you are on Windows, go to step 3.
   - If you are on UNIX, go to step 4.
3. On Windows, create an Execute Windows batch command build step:
   a. In the **Build** section, click **Add build step** and select **Execute Windows batch command**.
   b. In the **Command** field, enter the following command: %PEGA_HOME%\scripts\utils\prpcServiceUtils.bat export --connPropFile %WORKSPACE%\%SystemName%_export.properties --artifactsDir %WORKSPACE%
4. On UNIX, create an Execute Shell batch command build step:
   a. In the **Build** section, click **Add build step** and select **Execute Shell batch command**.
   b. In the **Command** field, enter the following command: $PEGA_HOME/scripts/utils/prpcServiceUtils.sh export --connPropFile

$WORKSPACE/${SystemName}_export.properties --artifactsDir $WORKSPACE

- Adding import build steps
- Adding build steps to import or export the archive

Adding import build steps

To add import build steps to your build job, do the following steps:

1. Add an Invoke Ant build step:
   a. In the **Build** section, click **Add build step** and select **Invoke Ant**.
   b. n the **Targets** field, enter importprops.
   c. In the **Build File** field, enter the path to the build file:
   - On Windows, enter the following path: $PEGA_HOME\samples\Jenkins-build.xml
   - On UNIX, enter the following path: $PEGA_HOME/scripts/samples/jenkins/Jenkins-build.xml
2. Add a build step to run either prpcServiceUtils.bat or prpcServiceUtils.sh.
   - If you are on Windows, go to step 3.
   - If you are on UNIX, go to step 4.
3. On Windows, create an Execute Windows batch command build step:
   a. In the **Build** section, click **Add build step** and select **Execute Windows batch command**.
   b. In the **Command** field, enter the following command:
   %PEGA_HOME%\scripts\utils\prpcServiceUtils.bat import --connPropFile %WORKSPACE%\%SystemName%_import.properties --artifactsDir %WORKSPACE%
4. On UNIX, create an Execute Shell batch command build step:
   a. In the **Build** section, click **Add build step** and select **Execute Shell batch command**.
   b. In the **Command** field, enter the following command:
   $PEGA_HOME/scripts/utils/prpcServiceUtils.sh import --connPropFile $WORKSPACE/${SystemName}_import.properties --artifactsDir $WORKSPACE

- Adding export build steps
- Adding build steps to import or export the archive

Importing or exporting the archive by running the Jenkins job

Run a Jenkins job to import or export the application archive.

Do the following steps:

1. In Jenkins, click **Build with Parameters**.
2. When the parameter values are displayed, verify the default settings and edit any values.
3. Set the artifact directory where exported logs and files are downloaded in the following format: -artifactsDir=<var>path to artifact directory</var>
   The default directory is the logs directory.
4. Click **Build**.

Running test cases and suites with the Execute Tests service

You can use the Execute Tests service (REST API) to validate the quality of your code after every build is created by running unit test cases that are configured for the application.

A continuous integration (CI) tool, such as Jenkins, calls the service, which runs all the unit test cases or test suites in your application and returns the results in xUnit format. The continuous integration tool interprets the results and, if the tests are not successful, you can correct errors before you deploy your application.

When you use Jenkins, you can also use the Execute Tests service to run unit tests after you merge a branch on a remote system of record and start a job. For more information, see Remotely starting automation jobs to perform branch operations and run unit tests.

The service comprises the following information:

- Service name: Pega unit Rule-Test-Unit-Case pzExecuteTests
- Service package: Pega unit
- End point: http://<yourapplicationURL>/prweb/PRRestService/Pega unit/Rule-Test-Unit-Case/pzExecuteTests

You can quarantine a test case by marking it **Disabled**. A disabled test case is not run by the Execute Tests service. Test case quarantines prevent noncritical tests from running if they are causing failures so that the service can continue to run.

- Request parameters

  The Execute Tests service takes certain string request parameters.

- Response

  The service returns the test results in an XML file in xUnit format and stores them in the location that you specified in

the LocationOfResults request parameter.

- Configuring your default access group

  When you run the Execute Tests service, you can specify the access group that is associated with the application for which you want to run all unit test cases or a test suite. If you do not specify an access group or application name and version, the service runs the unit test cases or test suite for the default access group that is configured for your Pega Platform operator ID.

- Configuring your build environment

  Configure your build environment so that it can call the Execute Tests service and run all the unit test cases or a test suite in your application. Your configuration depends on the external validation engine that you use.

- Running tests and verifying results

  After you configure your validation engine, run the service and verify the test results. Your test suites and test cases must be checked in so that you can run them.

- Test failures

  Test cases and suites that are run using Execute tests services can fail for a few reasons.

Request parameters

The Execute Tests service takes certain string request parameters.

The strings are:

- ApplicationInformation – Optional. The name and version of the application for which you want to run Pega unit test cases. You can pass it instead of the AccessGroup parameter.
  - If you pass only this parameter, the service runs all the test cases in the application.
  - If you do not pass this parameter, the service runs all the test cases in the application that are associated with the default access group that is configured for your operator.

Use the format ApplicationInformation=<application_name:application_version>.

- AccessGroup – Optional. The access group that is associated with the application for which you want to run Pega unit test cases. You can pass it instead of the ApplicationInformation parameter.
  - If you pass this parameter, the service runs all the test cases in the application that are associated with this access group.
  - If you do not pass this parameter, the service runs all the test cases in the application that are associated with the default access group that is configured for your operator.

Use the format AccessGroup=<access_group_name:access_group_user>.

- TestSuiteID – The *pxInsName* of the test suite that you want to run. You can find this value in the XML document that comprises the test suite by clicking Actions > XML on the Edit Test Suite form. You can run one test suite at a time. When you use this parameter, all the test cases in the test suite are run, but no other test cases in your application are run. This parameter is required for Pega unit test suites. If test suites share the same name among applications:
  - If you pass the ApplicationInformation or AccessGroup parameter with the TestSuiteID parameter, the service runs the test suite in the application that you specified.
  - If you do not pass the ApplicationInformation parameter or the AccessGroup parameter with the TestSuiteID parameter, the system runs the test suite in the application that is associated with the default access group.

Use the format TestSuiteID=<*pxInsName*>.

- LocationOfResults – The location where the service stores the XML file that contains the test results. This parameter is optional for test cases and test suites.
- RunWithCoverage – Determines whether the application-level test coverage report is generated after the Execute Tests service runs all relevant test cases or the selected test suite. For more information, see Generating an application-level test coverage report.
  - If you set the parameter to False, the application-level test coverage report is not generated. This is the default behavior.
  - If you set the parameter to True, and application-level coverage is not running, the Execute Tests service starts application-level coverage mode, runs all unit tests, stops coverage mode, and generates the application-level coverage report. This report is displayed on the test coverage landing page in the Application level section.
  - If you set the parameter to True, and application-level coverage is already running, the Execute Tests service returns an error.

Response

The service returns the test results in an XML file in xUnit format and stores them in the location that you specified in the LocationOfResults request parameter.

The output is similar to the following example:

<test-case errors="2" failures="0" label="Purchase order transformation with a bad element in the output expected" name="report-bad-element-name" skip="0" tests="7"> <nodes expected="/" result="/"><nodes xmlns:purchase="urn:acme-purchase-order" expected="/purchase:order[1]" result="/purchase-order[1]"><error type="Local name comparison">Expected "order" but was "purchase-order"</error><error type="Namespace URI comparison">Expected "urn:acme-purchase-order" but was ""</error></nodes></nodes><sysout>This text is captured by the report</sysout><syserr/></test-case>

Configuring your default access group

When you run the Execute Tests service, you can specify the access group that is associated with the application for which you want to run all unit test cases or a test suite. If you do not specify an access group or application name and version, the service runs the unit test cases or test suite for the default access group that is configured for your Pega Platform operator ID.

1. In the navigation pane of Dev Studio, click the **Operator** menu, and then click **Operator**.
2. In the **Application Access** section, select your default access group.
3. Selecting default access group configuration
4. Click **Save**.

Configuring your build environment

Configure your build environment so that it can call the Execute Tests service and run all the unit test cases or a test suite in your application. Your configuration depends on the external validation engine that you use.

For example, the following procedure describes how to configure the Jenkins server to call the service.

1. Open a web browser and go to the location of the Jenkins server.
2. Install the HTTP request plug-in for Jenkins to call the service and the JUnit plug-in so that you can view reports in xUnit format.
    a. Click **Manage Jenkins**.
    b. Click **Manage Plugins**.
    c. On the **Available** tab, select the **HTTP Request Plugin** and the **JUnit Plugin** check boxes.
    d. Specify whether to install the plug-in without restarting Jenkins or to download the plug-in and install it after restarting Jenkins.
3. Configure the Pega Platform credentials for the operator who authenticates the Execute Tests service.
    a. Click **Credentials**, and then click **System**.
    b. Click the drop-down arrow next to the domain to which you want to add credentials, and click **Add** credentials.
    c. In the **Username** field, enter the operator ID that is used to authenticate the service. This operator should belong to the access group that is associated with the application for which you want to run test cases and test suites.
    d. In the **Password** field, enter the password.
    e. Click **OK**.
4. Configure the Jenkins URL that runs the service.
    a. Click **Manage Jenkins**, and then click **Configure System**.
    b. In the **Jenkins Location** section, in the **Jenkins URL** field, enter the URL of the Jenkins server.
    c. Click **Apply**, and then click **Save**.
5. Add a build step to be run after the project is built.
    a. Open an existing project or create a project.
    b. Click **Configure**.
    c. In the **Build** section, click **Add build step**, and select **HTTP Request** from the list.
    d. In the **HTTP Request** section, in the URL field, enter the endpoint of the service. Use one of the following formats:
        - http://<*your application URL*>/prweb/PRRestService/Pega unit/Rule-Test-Unit-Case/pzExecuteTests
        - http://<*your application URL*>/prweb/PRRestService/Pega unit/Rule-Test-Unit-Case/pzExecuteTests?AccessGroup=<access_group_name:accessgroup_group_users>
        - http://<*your application URL*>/prweb/PRRestService/Pega unit/Rule-Test-Unit-Case/pzExecuteTests?TestSuiteID=<pxInsName>
        - http://<*your application URL*>/prweb/PRRestService/Pega unit/Rule-Test-Unit-Case/pzExecuteTests?ApplicationInformation?=ApplicationInformation:<application_name:application_version>

        If you are using multiple parameters, separate them with the ampersand (&) character, for example, http://<your application URL>/prweb/PRRestService/Pega unit/Rule-Test-Unit-Case/pzExecuteTests?ApplicationInformation?=ApplicationInformation:<application_name:application_version>&TestSuiteID=<pxInsName>

6. From the **HTTP mode** list, select **POST**.
7. Click **Advanced**.
8. In the **Authorization** section, from the **Authenticate** list, select the Pega Platform operator ID that authenticates the service that you configured in step 3.
9. In the Response section, in the Output response to file field, enter the name of the XML file where Jenkins stores the output that it receives from the service. This field corresponds to the LocationOfResults request parameter. In the Post-build Actions section, from the Add post build section list, select Publish Junit test result report and enter **/*.xml

in the Test Report XML field. This setting configures the results in xUnit format, which provides information about test results, such as a graph of test results trends. These results are displayed on your project page in Jenkins.
10. Click **Apply**, and then click **Save**.

Running tests and verifying results

After you configure your validation engine, run the service and verify the test results. Your test suites and test cases must be checked in so that you can run them.

For example, in Jenkins, complete the following steps.

1. Open the project and click **Build Now**.
2. In the Build History pane, click the build that you ran.
3. On the next page, click **Test Result**.
4. In the **All Tests** section, click root. The results of all tests are displayed.
5. **Optional:** Expand a test result in the **All Failed Tests** section and view details about why the test was not successful.

Test failures

Test cases and suites that are run using Execute tests services can fail for a few reasons.

Reasons for failed tests:

- The operator does not have access to the location of the results.
- The access group that is passed by the service either does not exist or no access group is associated with the operator ID.
- The application name and version that are passed do not exist.
- An application is not associated with the access group that is passed by the service.
- No Pega unit test cases or test suites are in the application.
- The test suite *pxInsName* does not exist for the application name and version or for the access group that is passed by the service.

Testing Pega applications

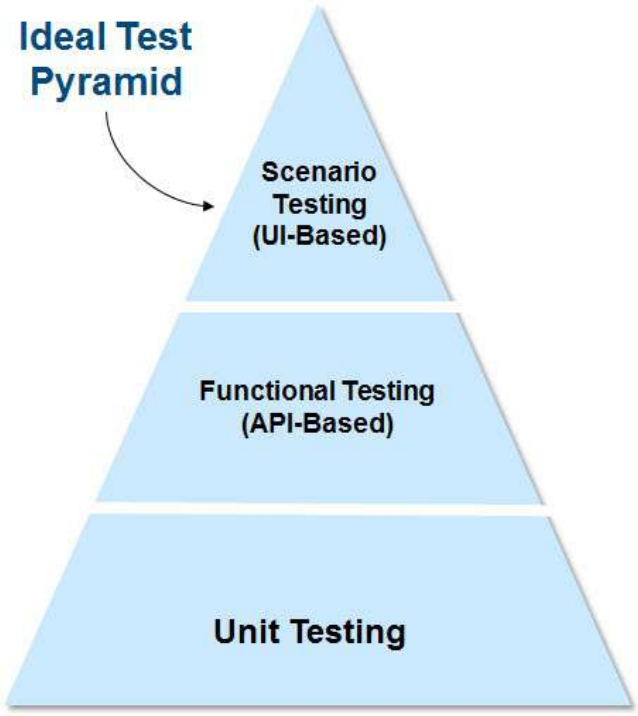
Having an effective automation test suite for your application in your continuous delivery DevOps pipeline ensures that the features and changes that you deliver to your customers are of high-quality and do not introduce regressions.

At a high level, the recommended test automation strategy for testing your Pega applications is as follows:

- Create your automation test suite based on industry best practices for test automation
- Build up your automation test suite by using Pega Platform capabilities and industry test solutions
- Run the right set of tests at different stages of your delivery pipeline
- Test early and test often

Industry best practices for test automation can be graphically shown as a test pyramid. Test types at the bottom of the pyramid are the least expensive to run, easiest to maintain, take the least amount of time to run, and should represent the greatest number of tests in the test suite. Test types at the top of the pyramid are the most expensive to run, hardest to maintain, take the most time to run, and should represent the least number of tests in the test suite. The higher up the pyramid you go, the higher the overall cost and the lower the benefits.

Ideal test pyramid



- PegaUnit testing

  Automated unit testing is a key stage of a continuous development and continuous integration model of application development. With continuous and thorough testing, issues are identified and fixed prior to releasing an application, which improves the application quality.

- Running scenario tests against a user interface

  Run scenario tests against a user interface to verify that the end-to-end scenarios are functioning correctly. The UI-based scenario testing tool allows you to focus on creating functional and useful tests, rather than writing complex code.

- Analyzing application quality metrics

  Quickly identify areas within your application that need improvement by viewing metrics related to your application's health on the Application Quality dashboard.

- Setting up for test automation

  Before you create Pega unit test cases and test suites, you must configure a test ruleset in which to store the tests.

PegaUnit testing

Automated unit testing is a key stage of a continuous development and continuous integration model of application development. With continuous and thorough testing, issues are identified and fixed prior to releasing an application, which improves the application quality.

Automated unit testing involves creating unit test cases for tests that are run against individual rules, grouping multiple test cases into test suites, running the tests, and viewing the results. When the tests run, the results are compared to the expected results that are defined in assertions.

- Understanding unit test cases

  A test case identifies one or more testable conditions (assertions) that are used to determine whether a rule returns an expected result. Reusable test cases support the continuous delivery model, providing a way to test rules on a recurring basis to identify the effects of new or modified rules.

- Grouping test cases into suites

  You can group related unit test cases or test suites into a test suite so that you can run multiple test cases and suites in a specified order. For example, you can run related test cases in a regression test suite when changes are made to application functionality.

- Setting up and cleaning the context for a test case or test suite

  You can set up the environment and conditions required for running a test case, determine how to clean up test data at the end of the test run, and set pages on which to automatically run rules.

- Viewing unit test reports

  View a graph with test pass rate trend data, a summary of Pega unit tests that were run, and an overview of Pega unit test compliance for currently included applications on the Reports tab on the Unit Testing landing page.

- Viewing unit tests without rules

  On the Application: Unit testing landing page you can display a list of unit tests that are not associated with any rule and export this list to an XLS or a PDF file. You should deactivate these unit tests because they will always fail.

- Understanding Pega Platform 7.2.2 and later behavior when switching between Pega unit testing and Automated Unit Testing features

  Beginning with Pega 7.2.2, you can use Pega unit testing to create test cases to validate the quality of your application by comparing the expected test output with results that are returned by running rules. In previous versions, you used Automated Unit Testing (AUT) to create test cases.

- Working with the deprecated AUT tool

  In older versions of Pega Platform, automated unit tests were created using the Automated Unit Testing (AUT) tool, which has since been replaced by PegaUnit testing. If you have automated unit tests that were created using AUT and they haven't been changed to PegaUnit test cases, then you can switch back to AUT to manage those tests.

Understanding unit test cases

A test case identifies one or more testable conditions (assertions) that are used to determine whether a rule returns an expected result. Reusable test cases support the continuous delivery model, providing a way to test rules on a recurring basis to identify the effects of new or modified rules.

You can run test cases whenever code changes are made that might affect existing functionality. For example, an account executive wants to ensure that a 10% discount is applied to all preferred customers. You create a test case that verifies that this discount is applied to all preferred customers in the database. The test case test fails if there are any preferred customers for which the 10% discount is not applied. You then add a new preferred customer to the database and run the test case to make sure that the customer is correctly configured to receive the discount and that the discount for other preferred customers is not affected.

Additionally, you can group related unit test cases into a test suite so that you can run multiple test cases and suites in a specified order. For example, you can run related test cases in a regression test suite when changes are made to application functionality. For more information about test suites, see Grouping test cases into suites.

After you create test cases and test suites, you can run them in a CI/CD pipeline for your application by using Deployment Manager or a third-party automation server such as Jenkins. For more information, see Using Deployment Manager for model-driven DevOps.

You can use unit test the following types of rules:

• Activities • Case types • Collections • Data pages • Data transforms • Decision tables • Decision trees	• Declare expressions • Flows • Map values • Report definitions • Strategies • When

Typically, you unit test a rule, and then convert it to a test case. For flow and case type rules, you record the test case.

- Viewing test coverage reports

  View a report that contains the results of test coverage sessions to determine which rules in your application are not covered with tests. You can improve the quality of your application by creating tests for all uncovered rules that are indicated in the reports.

- Creating unit test cases for flows and case types

  When you create a unit test case for a flow or case type, you run the flow or case type and enter data for assignments and decisions. The system records the data that you enter in a data transform, which is created after you save the test form. You can start recording at any time.

- Defining expected test results with assertions

  Use unit test cases to compare the expected output of a rule to the actual results returned by running the rule. To define the expected output, you configure assertions (test conditions) on the test cases that the test, when run,

compares to the results returned by the rule.

- Opening a unit test case

  You can view a list of the unit test cases that have been created for your application and select the one that you want to open.

- Running a unit test case

  Run a unit test case to validate rule functionality.

- Viewing test case results

  After you run a unit test case, you can view the results of the test run.

- Exporting a list of test cases

  You can export a list of all the unit test cases that are in your application or configured on a rule form.

- Managing unit tests and test suites

  On the Application: Unit testing landing page you can run unit test cases and test suites that are not marked as disabled, and view reports to check the quality of your application to identify rules that did not pass unit testing.

- Running test cases and suites with the Execute Tests service

Creating unit test cases for rules

For most rules, you can create a reusable test case by converting a unit test to a test case, configuring case details, and then defining expected test results with assertions (test conditions). When the test case runs, the test results are compared to the expected results defined for the rule's assertions. If the test results do not meet the defined assertions, then the test fails.

Before you begin: Unit test a rule and convert the test run into a test case. For more information, see Unit testing individual rules.

1. **Optional:** To modify the rule or class that is used for the test, in the upper-right corner of the **Definition** tab, click the **Gear** icon, select the rule or class and then click **Submit.**

   If you are testing a strategy rule, then the *componentName* and *pzRandomSeed* parameters are also displayed. If you change either of these parameters, the test case does not return the expected results.

   - *componentName* – The name of the component (for example, Switch) that you are testing.
   - *pzRandomSeed* – An internal parameter, which is the random seed for the Split and Champion Challenger shapes.
2. **Optional:** To prevent the test from being run as part of a test suite or from a REST service, on the **Definition** tab, select the **Disable** check box.
   The test case will be run only when you click **Actions Run** .
3. In the **Expected results** section, add assertions that define the expected results of the test. For more information about creating assertions, see Assertions.
4. On the **Setup & Cleanup** tab, configure the actions to perform and the objects and clipboard pages to be available before and after the test runs. You can also clean up the clipboard after the test is run by applying additional data transforms or activities. For more information, see Setting up your test environment.
5. Click **Save**.
6. In the **Details** dialog box, enter a label that identifies the test case. The test case identifier is generated based on the label and cannot be modified after it is saved.

Creating unit test cases for flows and case types

When you create a unit test case for a flow or case type, you run the flow or case type and enter data for assignments and decisions. The system records the data that you enter in a data transform, which is created after you save the test form. You can start recording at any time.

Certain conditions apply on the data that you can record for flow and case types. For information, see Data that you can record for flows and case types about the data that you can record.

Restriction: From Pega Platform version 8.5, running and recording a test case for a process that has a Create stage is no longer possible for new case types. If you have migrated from a version lower than 8.5, you can include the process in any stage of a migrated case type, and then run the test on the process.
Before you begin: Exclude properties in your work class from the test by modifying the *pxCapturePropertyIgnore* data transform. For more information, see Data that you can record for flows and case typesabout the data that you can record.

1. Open the flow or case type for which you want to record a test.
2. On the toolbar, click **Actions Record test case** .
   The system starts running the flow or case type.

3. Enter input as you step through the flow or case type.
4. Click **Create test case** in the lower-right of the screen to save the recording as a test case.
5. Click **Save**, enter a label that identifies the test case, and then click **Submit.**
6. **Optional:** To modify the rule or class that is used for the test, in the upper-right corner of the **Definition** tab, click the **Gear** icon, select the rule or class, and then click **Submit.**
7. **Optional:** To prevent the test from being run as a part of a test suite or from a REST service, on the **Definition** tab, select the **Disable** check box.
   The test case will be run only when you manually click **Actions Run** .
8. In the **Expected results** section, add assertions that define the expected results of the test. For more information about creating assertions, see Defining expected test results with assertions.
9. On the **Setup & Cleanup** tab, configure the actions to perform and the objects and clipboard pages to be available before and after the test runs. You can also clean up the clipboard after the test is run by applying additional data transforms or activities. For more information, see Setting up and cleaning the context for a test case or test suite.
10. Click **Save**.
11. Configure the unit test case. See Creating unit test cases for rules for more information.

Result:

After you save the test case, a data transform, which captures the input that you entered, is created and associated with the test case. You can edit this data transform to modify the test case input. The Edit test case form also displays the path of the flow or case type.

- Data that you can record for flows and case types

  When you create a unit test case for a flow or case type, the system records the data that you enter.

- Excluding work class properties from flows and case type tests

  Exclude properties in your work class from the test by modifying the pyDataCapturePropertyIgnores data transform.

- Creating unit test cases for rules
- Running a unit test case
- Viewing test case results
- Exporting a list of test cases

Data that you can record for flows and case types

When you create a unit test case for a flow or case type, the system records the data that you enter.

You can record the following type of information:

- Starter flows. Non-starter flows may be tested from a starter flow that calls on the non-starter flow.

- Subprocesses that are configured as part of a flow.
- The Assignment, Utility, and Approval shapes. For flows, assignments must be routed to the current operator so that the recording of the flow continues and the system captures data as part of the test case.
- Data that is captured on the *pyWorkPage*.

  When a flow or case type runs, a *pyWorkPage* is created on the clipboard and captures information such as data that you enter for assignments. It also captures information such as case ID, date and time that the case was created, and the latest case status.

  There are additional assertions that you can configure for flows and case types, including case status, assigned to, and attachment exists. For these assertions, the system compares expected values to the value that is recorded on the *pyWorkPage*.

  If you refresh or cancel recording the flow or case type, data that is on the *pyWorkPage* might not be accurate.

- Local actions and flow actions that are configured as part of the flow or case type.
- Child cases that are created and finish running before the flow or test case resumes running.

- All properties, excluding properties that begin with either px or pz.

- Creating unit test cases for flows and case types

Excluding work class properties from flows and case type tests

Exclude properties in your work class from the test by modifying the *pyDataCapturePropertyIgnores* data transform.

Some properties, like *.pyID*, are not processed when a unit test case is run. These properties vary for every test run. The *pxDataCapturePropertyIgnore* data transform displays the properties that unit tests do not process.

1. Open the data transform:
   a. In the navigation pane of Dev Studio, click **App Classes** , and enter Work- in the **Search** field.
   b. Expand **Data Model > Data Transform** and then click **pyDataCapturePropertyIgnores**.

2. Save the data transform to your *Work-* class and in your test ruleset.
3. For each property that you want to exclude in the data transform, do the following steps:
    a. On the **Definition** tab, click the **Add** icon.
    b. From the **Action** list, select **Set**.
    c. In the **Target** field, enter the property that you want to exclude.
    d. In the **Source** field, enter two double quotation marks, separated by a space: " ".
4. Save the data transform.

Defining expected test results with assertions

Use unit test cases to compare the expected output of a rule to the actual results returned by running the rule. To define the expected output, you configure assertions (test conditions) on the test cases that the test, when run, compares to the results returned by the rule.

When a test runs, it applies assertions in the order that you define them on the **Definition** tab of the test case. All assertions, except for run time assertions, must pass for the test to be successful.

For example, an account executive wants to ensure that a 10% discount is applied to all preferred customers. You can create a test case that verifies that this discount is applied to all preferred customers in the database. If the test does not pass, the results indicate where the 10% discount is not applied.

Note: On decision trees and decision rules, you cannot configure properties from a read-only data page or a data page that is a declarative target.

- Configuring activity status assertions

  You can verify that an activity returns the correct status when it runs by configuring an activity status assertion. You can also assert if an activity has an error and, if it does, what the message is so that you can validate that the message is correct.

- Configuring assigned to assertions

  For flows and case types, you can use the assigned to assertion to verify that an assignment is routed to the appropriate work queue or operator.

- Configuring attachment exists assertions

  For flows and case types, you can verify that the flow or case type has an attachment of type file or note (attached using the Attach Content shape) or email (attached using the Send Email shape) attached.

- Configuring case instance count assertions

  For flows and case types, you can verify the number of cases that were created when the case type or flow was run.

- Configuring case status assertions

  You can configure a case status assertion on a flow or case type to verify the status of the case.

- Configuring decision result assertions

  After you create a unit test case for a decision table or decision tree, the system generates a decision result assertion. This assertion displays the input values for testing the rule, and the result that is generated by the rule.

- Configuring expected run-time assertions

  You can create an assertion for the expected run time of the rule. The expected run-time assertion is less than or equal to an amount of time that you specify, in seconds.

- Configuring list assertions

  You can create list assertions for page lists on a rule to determine if either the expected result is anywhere in the list of results returned by the rule. Even if the order of results changes, the test will continue to work.

- Configuring page assertions

  Some rules, such as activities and data transforms, can create or remove pages from the system. You can create page assertions to determine whether or not a page exists after a unit test case runs. You can also assert if a property has an error and, if it does, what the message is so that you can validate that the message is correct.

- Configuring property assertions

  You can configure property assertions to validate that the actual values of properties returned by a rule are the expected values. You can also assert if a property has an error and, if it does, what the message is so that you can validate that the message is correct.

- Configuring result count assertions

You can configure assertions to compare the number of items returned in a page list, page group, value list, or value group on the rule to the result that you expect to see on the clipboard.

Configuring activity status assertions

You can verify that an activity returns the correct status when it runs by configuring an activity status assertion. You can also assert if an activity has an error and, if it does, what the message is so that you can validate that the message is correct.

Before you begin: Open the unit test case. For more information, see Opening a unit test case.

1. On the bottom of the **Definition** tab, click **Add expected result**.
2. In the **Assertion type** list, click **Activity status**.
3. In the **Value** list, click the status that you expect the activity to return when the test runs.
4. To validate the message that displays for the activity, select **Validate message**, select a **Comparator**, and then enter the message that you want to validate in the **Value** box.
5. **Optional:** To add a comment, click the **Add comment** icon, enter a comment, and then click **OK**.
6. Click **Save**.

- Defining expected test results with assertions

Configuring assigned to assertions

For flows and case types, you can use the assigned to assertion to verify that an assignment is routed to the appropriate work queue or operator.

If you have multiple assignments on a flow or test case, you can route each assignment to an operator ID or work queue. Clipboard pages are created for each assignment under the **pyWorkPage** page and capture the assignment details, including the operator ID or work queue to which the assignment was routed. The assigned to assertion compares the operator ID or work queue to the last assignment that is configured on the flow or case type, which depends on where you stop recording the flow or case type.

For example, your flow has a Customer Details assignment, which is routed to the operator ID johnsmith. It also has a subprocess with an Account Information assignment, which is routed to the account_processing work queue.

If you record only the Customer Details assignment, the assigned to value is johnsmith. If you also record the Account Information assignment, the assigned to value is account_processing.

Before you begin: Open the unit test case. For more information, see Opening a unit test case.

1. On the bottom of the **Definition** tab, click **Add expected result**.
2. From the **Assertion type** list, select **Assigned to**.
3. From the **Assigned to** list, select **Operator** or **Work queue**.
4. Select a comparator from the **Comparator** list.
5. In the **Value** field, press the Down Arrow key and select the operator ID or work queue.
6. **Optional:** To add a comment, click the **Add comment** icon, enter a comment, and click **OK**.
7. Click **Save**.

- Defining expected test results with assertions

Configuring attachment exists assertions

For flows and case types, you can verify that the flow or case type has an attachment of type file or note (attached using the Attach Content shape) or email (attached using the Send Email shape) attached.

Before you begin: Open the unit test case. For more information, see Opening a unit test case.

1. On the bottom of the **Definition** tab, click **Add expected result**.
2. From the **Assertion type** list, select **Attachment exists**.
3. From the **Attachment type** list, select one of the following options, and then provide the value for each field:
    - **File**: Select to specify that the attachment type is file, and then enter the following values:
        - **Description**: Enter the text that was provided as the description in the Attach Content shape.
        - **Name**: Enter the name of the file that was provided in the Attach Content shape.
    - **Note**: Select to specify that the attachment type is note, and then enter text that was entered as the note description in the Attach Content shape.
    - **Email**: Select to specify that the attachment type is an email, and then enter the email subject that was provided in the Send Email shape.
4. Repeat steps 1 through 3 to add additional attachment assertions.
5. **Optional:** To add a comment, click the **Add comment** icon, enter a comment, and click **OK**.
6. Click **Save**.

- Attachment exists assertions

  On case types and flows, you can test whether an attachment of type file or note, which were attached in the Attach Content shape, or email, which was attached using the Send Email shape, exists.

- Defining expected test results with assertions

Attachment exists assertions

On case types and flows, you can test whether an attachment of type file or note, which were attached in the Attach Content shape, or email, which was attached using the Send Email shape, exists.

If you have multiple attachments on a flow or test case that match the expected value of the assertion, the assertion runs for every attachment that exists. If the system finds an attachment that matches the assertion value, the assertion passes and iterates over all the attachments on the flow or case type. If no attachment exists, the assertion fails

The system compares the expected output on attachments that are recorded on the *pyWorkPage* page. For example, if a case type has a parent case that spins off a child case, and you record just the child case, the **pyWorkPage** page records attachments for only the child case and not the parent case, which is recorded on the *pyWorkCover* page.

In addition, if you create a test case from a parent case that generates a child case that is returned to the parent case after the child case runs, the *pyWorkPage* page records the attachments only on the parent case.

For example, your case has an Attach Content shape that attaches a Process immediately note in the first stage of the case type. In the third stage, your case has a Send Email shape that attaches an email with the subject Request approved. The assertion passes if you searched for either the Process immediately note or Request approved email subject.

- Configuring attachment exists assertions
- Defining expected test results with assertions

Configuring case instance count assertions

For flows and case types, you can verify the number of cases that were created when the case type or flow was run.

For example, a Job Application case type runs a child case that processes background checks. If you record the entire Job Applicant case type and the child case type, the number of case instances for Job Application case type is one, and the number of case instances of Background Check child case type is one.

If you do not run the run the Background Check child case type when you create the test case, the number of Background Check case instances is zero.

Before you begin: Open the unit test case. For more information, see Opening a unit test case.

1. On the bottom of the **Definition** tab, click **Add expected result**.
2. From the **Assertion type** list, select **Case instance count**.
3. In the **Of case type** field, do one of the following:
   - To select a case type from your work pool, press the Down Arrow key and select the case type.
   - Enter a case type that is not part of your work pool.
4. Select a comparator from the **Comparator** list.
5. In the **Value** field, enter the number of cases to compare against the output.
6. **Optional:** Click **Add** to add another case instance count assertion and repeat steps 4 through 6.
7. **Optional:** To add a comment, click the **Add comment** icon, enter a comment, and click **OK**.
8. Click **Save**.

- Defining expected test results with assertions

Configuring case status assertions

You can configure a case status assertion on a flow or case type to verify the status of the case.

If you have multiple assignments on a flow or case type, you can configure a case status on each assignment. The *pyWorkPage* on the clipboard captures the latest case status, which depends on where you stop recording the flow or case type.

For example, your flow has a Customer Details assignment, with the case status set as New. It also has a subflow with an Account Information assignment, with the case status set as Pending.

If you record only the Customer Details assignment, the case status, which is captured in the *.pyStatusWork* property on the *pyWorkPage*, is set to New. If you also record the Account Information assignment, the case status is set to Completed.

Before you begin: Open the unit test case. For more information, see Opening a unit test case.

1. On the bottom of the **Definition** tab, click **Add expected result**.
2. From the **Assertion type** list, select **Case status**.

3. Select the comparator from the **Comparator** list.
4. In the **Value** field, press the Down Arrow key and select the case status.
5. **Optional:** To add a comment, click the **Add comment** icon, enter a comment, and click **OK**.
6. Click **Save**.

- Defining expected test results with assertions

Configuring decision result assertions

After you create a unit test case for a decision table or decision tree, the system generates a decision result assertion. This assertion displays the input values for testing the rule, and the result that is generated by the rule.

You can manually update the input values, add properties, remove properties, and modify the default decision result if the test is modified.

Note: This assertion is supported on when rules, decision tables, and decision trees only.

Before you begin: Open the unit test case. For more information, see Opening a unit test case.

1. Click the **Definition** tab.
2. To add multiple input values and results to the assertion, or add other assertions, perform one of the following actions:
   Note: You can add multiple input values and results to this assertion but cannot add other assertion types to this test case. You can add other assertion types to this test case only if you have a single input and result entry for the assertion.
   - To add multiple input values and results to the assertion:
   a. Select the **Multiple input combinations** check box.
   b. Enter values for the input and result that you expect the assertion to generate when the test stops running.
   c. Click **Add** and enter values for each additional input and result that you want to test.
   - To use one input value and result, enter the values that you expect the assertion to generate when the test stops running. You can then add additional assertions to the test case.
3. To update the assertion to reflect properties that were added to the rule, click **Refresh**.
   Note: Refresh updates the assertion with properties that are added to the rule. If properties have been removed from the rule, then you need to manually remove the properties from the assertion.
4. Add or remove properties by clicking **Manage properties** and then entering the changes. You need to enter data for properties that were added to the rule. **Result:** The properties are reflected as unexpected results in test case results.
5. In the rule form, click **Save**. **Result:**

   The test case runs the decision tree or decision table with each input combination and compares the result with the expected decision result for that combination.

   Other decision result combinations or other configured assertions then run. If the expected result of any of the input combinations in the decision result assertion does not match the result that the rule returns, the assertion fails.

- Defining expected test results with assertions

Configuring expected run-time assertions

You can create an assertion for the expected run time of the rule. The expected run-time assertion is less than or equal to an amount of time that you specify, in seconds.

An actual run time that is significantly longer than the expected run time can indicate an issue. For example, if you are using a report definition to obtain initial values for a data page from a database, there might be a connectivity issue between the application and the database.

By default, after you create a Pega unit test case for a data page, the system generates the expected run-time assertion. The default value of the expected run time is the time that is taken by the rule to fetch results when the test was first run. The system compares that time against future run-time tests.

You can change the default value and configure expected run time assertions for all rule types.

Before you begin: Open the unit test case. For more information, see Opening a unit test case.

1. On the bottom of the **Definition** tab, click **Add expected result**.
2. From the **Assertion type** list, select **Expected run time**.
3. In the **Value** field, enter a value, in seconds, that specifies the amount of time within which the execution of the rule should be completed.
4. **Optional:** If you want the test case to fail when the rule is not run within the specified time, select the **Fail the test case when this validation fails** check box.
5. **Optional:** To add a comment, click the **Add comment** icon, enter a comment, and click **OK**.
6. Click **Save**.

- Defining expected test results with assertions

Configuring list assertions

You can create list assertions for page lists on a rule to determine if either the expected result is anywhere in the list of results returned by the rule. Even if the order of results changes, the test will continue to work.

For example, you can verify if a product is present in a product list in a data page, regardless of where the product appears in the list results. You can also verify if there is at least one employee with the name John in the results of the Employee list data page.

You can also configure assertions for page lists to apply assertions to all the results that are returned by a rule so that you do not have to manually create assertions for each result in the list.

For example, you can verify that a department name is Sales and that a department ID starts with SL for each department in the list of results in the Sales department data page. You can also verify if a discount of 10% is applied to each customer in the list of results of the VIP customers data page.

You can configure list assertions for page lists on a rule to apply assertions to all the results that are returned by the rule. Configure an ordered list assertion so that you do not have to manually create assertions for each result in the list.

Before you begin: Open the unit test case. For more information, see Opening a unit test case.

1. On the bottom of the **Definition** tab, click **Add expected result**.
2. From the **Assertion type** list, select **List**.
3. Add properties to the assertion.
    a. Click **Add properties**.
    b. If you are adding properties for flows, case types, decision trees, decision tables, or data pages:
        1. In the **of object** field, enter the path of the object with which the properties are compared during the assertion.
        2. Proceed to step d.
    c. If you are adding properties for data transforms or activities, complete the following tasks:
        1. From the **Thread** list in the **Actual results** section, select the thread that contains the page whose properties or pages you want to add.
        2. In the **Page** field, enter the page whose properties or pages you want to add.
        3. In the **of object** field, enter the path of the object with which the properties are compared during the assertion.
        4. Proceed to step d.
    d. Select the properties or pages that you want to add. You can search for a property or its value by entering text in the search bar and pressing **Enter**.

        If you select a page, all embedded pages and properties from the page are added. Added properties are displayed in the right pane.

        When you add multiple properties, the assertion passes if the expected output and results match for all properties.
4. **Optional:** In the **Filter** field, enter a property and value on which to filter results or open the Expression Builder by clicking the **Gear** icon to provide an expression that is used to filter results. The list assertion applies only to the page list entries that are specified for this filter value.
5. From the **Comparator** list, select the comparator that you want to use to compare the property with a specified value.

    Select the is in comparator to compare a text, integer, or decimal property to multiple values. The assertion passes if the property matches any of the values that you specify.
6. In the **Value** field, either enter a value with which to compare the property or open the Expression Builder by clicking the **Gear** icon to enter an expression that is used to provide the value.
    Note: The **Gear** icon is not displayed until after you have saved the rule form.
7. To add a comment, click the **Add comment** icon, enter a comment, and click **OK**.
8. Click **Done**.
9. Click **Save**.

Result:

When you run the test case, the system searches for the specified properties in the page list. One of the following occurs:

- If you selected **In ANY Instance**, the assertion passes if all the properties in the set match the expected values in the page list. If none of the properties match any of the values in the page list, the assertion does not pass.
- If you selected **In ALL instances**, the assertion passes if all the properties in the set match the expected values in every entry in the page list. If any of the properties do not match any entry in the page list, the assertion does not pass.

Configuring page assertions

Some rules, such as activities and data transforms, can create or remove pages from the system. You can create page assertions to determine whether or not a page exists after a unit test case runs. You can also assert if a property has an error and, if it does, what the message is so that you can validate that the message is correct.

You can configure page assertions for embedded pages, data pages, data pages with parameters, and embedded pages within data pages that either have or do not have parameter stop-level pages.

Before you begin: Open the unit test case. For more information, see Opening a unit test case.

1. On the bottom of the **Definition** tab, click **Add expected result**.
2. In the **Assertion type** list, select **Page**.
3. In the **Page** field, enter the name of the page.
4. In the **Comparator** list, select the comparator that you want to use to compare the property with a specified value:
    - To ensure that the page is created after the unit test runs, select **exists**. The assertion passes if the system does not find the page.
    - To ensure that the page is removed after the unit test runs, select **does not exist**. The assertion passes if the system does not find the page.
    - To ensure that the page has an error after the unit test runs, select **has errors**. The assertion passes if the system finds errors on the page.
    - To ensure that the page is free of errors after the unit test runs, select **has no errors**. The assertion passes if the system finds no errors on the page.
    - To ensure that the page has a specific error message after the unit test runs, select **has error with message** and then enter the message in the **Value** box or click the **Gear** icon to build an expression. The assertion passes if the page contains the complete error message.
    - To ensure that the page has a portion of an error message after the unit test runs, select **has error message that contains** and then enter the message in the **Value** box or click the **Gear** icon to build an expression. The assertion passes if the page contains the words or phrases in the error message.
5. **Optional:** To add another page to the assertion, click **Add pages**, and then perform steps 3 through 4.
6. **Optional:** To add a comment, click the **Add comment** icon, enter a comment, and then click **OK**.
7. Click **Save**.

For example: An activity runs every week to check the last login time of all operators and deletes any operator record (page) from the system if the last login was six months ago. When you test this activity, you can:

1. Set up the clipboard to load an operator page that has the last login time as six months ago.
2. Create a page assertion that ensures that the page no longer exists after the activity runs.

- Page assertions

  You can configure page assertions to determine if a page exists on the clipboard or if a page has errors.

- Defining expected test results with assertions

Page assertions

You can configure page assertions to determine if a page exists on the clipboard or if a page has errors.

You can configure page assertions on the following types of pages:

- Embedded pages
- Data pages
- Data pages with parameters
- Embedded pages within data pages that either have or do not have parameters
- Top-level pages

For example, an activity runs every week to check the last login time of all operators and deletes any operator record (page) from the system if the last login was six months ago. When you test this activity:

- Set up the clipboard to load an operator page that has the last login time as six months ago.
- Create a page assertion that ensures that the page no longer exists after the activity runs.

- Configuring page assertions
- Defining expected test results with assertions

Configuring property assertions

You can configure property assertions to validate that the actual values of properties returned by a rule are the expected values. You can also assert if a property has an error and, if it does, what the message is so that you can validate that the message is correct.

For example, you can create an assertion that verifies that a customer ID, which appears only once on a data page, is equal to 834234.

Before you begin: Open the unit test case. For more information, see Opening a unit test case.

1. On the bottom of the **Definition** tab, click **Add expected result**.
2. In the **Assertion type** list, select **Property**, and then click **Add properties**.
3. Select the properties to add by doing one of the following.
   - For data transforms, activities, flows, or case types, in the **Actual results** section, select the page containing the properties to add.
   - For other rules, select the property or page that you want to add.

   Result: Properties are displayed in the right pane. If you selected a page, then all embedded pages and properties from the page are added.
4. To add another property or page, click **Add row**, and then repeat step 3. **Result:**

   When you add multiple properties, the assertion passes if the expected output and results match for all properties.

5. In the **Comparator** list, select the comparator that you want to use to compare the property with a specified value. Do one of the following:

   - Select the `is in` comparator to compare a text, integer, or decimal property to multiple values. The assertion passes if the property matches any of the values that you specify.

   - Select the `is not in` comparator. The assertion passes if the property does not match any of the values that you specify.
   - Select the `has error with message` comparator to verify that the property has the exact message that you specify in the **Value** box.
   - Select the `has error message that contains` comparator to verify that the property has a portion of the message that you specify in the **Value** box.
6. In the **Value** field, enter a value with which to compare the property. Separate values for the comparators by using the pipe (|) character. For text properties, use double quotation marks at the beginning and end of the value, for example, `"23|15|88"` .

   For example, if you want the assertion to pass when Age property matches either the 5 or 7 values, configure the assertion as `.Age is in 5|7` .

7. **Optional:** To add a comment, click the **Add comment** icon, enter a comment, and then click **OK**.
8. Click **Save**.

- Defining expected test results with assertions

Configuring result count assertions

You can configure assertions to compare the number of items returned in a page list, page group, value list, or value group on the rule to the result that you expect to see on the clipboard.

For example, you can create an assertion that verifies that the number of returned results for the number of employees is greater than X number of employees, less than Y number of employees, or equal to Z number of employees.

Before you begin: Open the unit test case. For more information, see Opening a unit test case.

1. On the bottom of the **Definition** tab, click **Add expected result**.
2. For activities and data transforms, complete the following tasks:
   a. In the **Page** field, enter the page that contains the property for which you want to test the result count.
   b. In the **Page class** field, select the class to which the page belongs.
3. In the **of object** field, enter the path of the object with which the results are compared or counted against during the assertion.
   - For data pages, this value is usually .pxResults.
   - For data transforms and activities, you can use any page list on a page.
4. **Optional:** In the **Filter** field, enter a property and value on which to filter results or open the Expression Builder by clicking the **Gear** icon to provide an expression that is used to filter results. The list assertion applies only to the page list entries that are specified for this filter value.
5. Select the appropriate comparator from the **Comparator** list.
6. In the **Value** field, enter the value that you want to compare with the object.
7. **Optional:** To add a comment, click the **Add comment** icon, enter a comment, and click **OK**.
8. Click **Save**.

Result:

When you run the test case, the assertion fails if the expected value does not match the result count returned from the page list, page group, value list, or value group.

- Defining expected test results with assertions

Opening a unit test case

You can view a list of the unit test cases that have been created for your application and select the one that you want to open.

1. Open the test case. Complete one of the following actions:
    - In the navigation pane of Dev Studio, click **Configure Application Quality Automated Testing Unit Testing Test Cases** .
    - Click the **Test Cases** tab on the rule form.
2. In the **Test case name** column, click the test case that you want to open.

- Running a unit test case
- PegaUnit testing
- Managing unit tests and test suites

Running a unit test case

Run a unit test case to validate rule functionality.

1. Open the test case.
2. Complete one of the following actions:
    - To run multiple test cases, select the test cases that you want to run, and then click **Run selected**.
    - To run a disabled test case or a single test case, click the test case to open it, and then click **Actions Run** .

- PegaUnit testing
- Managing unit tests and test suites

Viewing test case results

After you run a unit test case, you can view the results of the test run.

The following information is displayed:

- When the test was last run and the user who ran it.
- The rule associated with the test.
- The parameters sent.
- Errors for failed tests.

- Unexpected results for failed tests. This information also includes the run time of the test and the expected run time of the test if the expected run time assertion fails.

1. Open the test case.
2. In the **Run history** column, click **View** for the test case that you want to view.
3. In the **Test Runs Log** dialog box, click the row for the instance of the test case that you want to view to open the test results in a new tab in Dev Studio.
   You can also view test case results in the Edit Test Case form after you immediately run the test, in the **Test Case** tab of the rule form or, for data pages, in the **Data Page testing** landing page.

- PegaUnit testing
- Managing unit tests and test suites

Exporting a list of test cases

You can export a list of all the unit test cases that are in your application or configured on a rule form.

1. Export a list of all the Pega unit test cases that are in your application or for a rule type.
   Complete one of the following actions:
    - To export a list of all the unit test cases that are in your application, in the header of Dev Studio, click **Configure Application Quality Automated Testing Unit Testing Test Cases** .
    - To export a list of Pega unit test cases that are configured on a rule form, click **Test Cases** in the rule form.
2. Click **Export to Excel**.

- PegaUnit testing
- Managing unit tests and test suites

Managing unit tests and test suites

On the Application: Unit testing landing page you can run unit test cases and test suites that are not marked as disabled, and view reports to check the quality of your application to identify rules that did not pass unit testing.

From the Application: Unit testing landing page, you can do the following tasks:

- View, open, and run unit test cases in your application, export all test cases in Excel format, and create test suites from selected test cases on the **Test Cases** tab and from existing test suites on the **Test Suites** tab. To access all test cases from your application stack, not just from the currently active application, select the **Include built-on**

applications option on the Application: Quality Settings landing page. **Note:** If a test case is marked as disabled, you cannot run it from the landing page or from a test suite. To run a disabled test case, open it in Dev Studio and click **Actions Run** .
- View a list of test cases with no associated rules in the **Test cases** tab by clicking **Tests without rules**.
- Create, view, open, and run Pega unit test suites in the **Test Suites** tab.
- View the test pass trend graph, the percentage of rules on which unit tests are configured, and see the run results of unit tests on the **Reports** tab.

Tip: To filter information by multiple criteria, click the **Advanced filter** icon.

To view and use the Automated Unit Testing (AUT) landing page, you must have the *AutomatedTesting* privilege enabled. Click **Switch to old version**. For more information, see the Pega Community article *Pega 7.2.2 and later behavior when switching between Pega unit testing and Automated Unit Testing features*.

AUT information is available on the Testing Applications landing page on Pega Community and in the Automated Unit Testing topics in the Pega 7.2.1 help.

- Changing application quality metrics settings
- PegaUnit testing
- Working with the deprecated AUT tool
- Running a unit test case
- Viewing unit test reports
- Viewing unit tests without rules

Grouping test cases into suites

You can group related unit test cases or test suites into a test suite so that you can run multiple test cases and suites in a specified order. For example, you can run related test cases in a regression test suite when changes are made to application functionality.

- Creating unit test cases for rules

  For most rules, you can create a reusable test case by converting a unit test to a test case, configuring case details, and then defining expected test results with assertions (test conditions). When the test case runs, the test results are compared to the expected results defined for the rule's assertions. If the test results do not meet the defined assertions, then the test fails.

- Opening a unit test suite

  You can view a list of the unit test suites that have been created for your application and select the one that you want to open.

- Running a unit test suite

  You can run a unit test suite to validate rule functionality by comparing the expected value to the output produced by running the rule. Test cases are run in the order in which they appear in the suite.

- Viewing unit test suite run results

  After you run a unit test suite, you can view the test run results. For example, you can view the expected and actual output for assertions that did not pass.

- Adding cases to a test suite

  You can add test cases to a unit test suite. When you run a test suite, the test cases are run in the order in which they appear in the suite.

- Viewing unit test suite run results

  After you run a unit test suite, you can view the test run results. For example, you can view the expected and actual output for assertions that did not pass.

- Opening a unit test suite
- Running a unit test case

Creating unit test suites

To create a unit test suite, add test cases and test suites to the suite and then modify the order in which you want them to run. You can also modify the context in which to save the scenario test suite, such as the development branch or the ruleset.

1. In the header of Dev Studio, click **Configure Application Quality Automated Testing Unit Testing Test Suites** .
2. Click **Create new suite**.
3. **Optional:** In **Description**, enter information that you want to include with the test suite. For example, enter information about when to run the test suite.

4. In the **Category** list, click the type of scenario test suite you are creating:
    - To informally test a feature, select **Ad-hoc**.
    - To verify critical application functionality, select **Smoke**.
    - To confirm that changes have not adversely affected other application functionality, select **Regression**.
5. **Optional:** Provide a value, in seconds, that specifies the length of time within which the run time of the suite should complete in the **Expected max runtime** field. If you want the test suite to fail when the expected run time has been exceeded, select the **Fail the test suite when runtime validation fails** check box.
6. Add unit tests cases or other test suites to the test suite:
    - To add test cases to the test suite, in the **Test cases** section, click **Add**, select the test cases to include in the suite, and then click **Add**.
    - To add test suites to the test suite, in the **Test suites** section, click **Add**, select the test suites to include in the suite, and then click **Add**.

    Note: To filter information by multiple criteria, click the **Advanced filter** icon.
7. **Optional:** To change the order in which the test cases or test suites run, drag them to a different position in the sequence.
8. Save the test suite:
    a. Click **Save** and then enter a **Label** that describes the purpose of the test suite.
    Note: Pega Platform automatically generates the **Identifier** based on the label you provide. The identifier identifies the scenario test suite in the system. To change the identifier, click **Edit**. The identifier must be unique to the system.
    b. **Optional:** In the **Context** section, change details about the environment in which the test suite will run. You can:
        - Change the development branch in which to save the scenario test suite.
        - Select a different application for which to run the scenario test suite.
        - Select a different ruleset in which to save the scenario test.
9. Click **Save**.
10. Complete any of the following actions:
    - Remove test cases or suites from the test suite by selecting them and clicking **Remove**.
    - Apply one or more data pages, data transforms, or activities to set up the clipboard before running a test suite in the **Setup** section of the **Setup & Cleanup** tab. You can also create objects, load work and data objects, and add user pages from the clipboard which will be available on the clipboard when running the test suite. For more information, see Setting up your test environment.
    - Apply additional data transforms or activities to clean up the clipboard in the **Cleanup** section of the **Setup & Cleanup** tab. You can also prevent the test data from being removed after the test suite runs. For more information, see Cleaning up your test environment.
    - Run a configured test suite by clicking **Actions Run** . **Note:** If you made changes to the suite, such as adding or removing test cases or test suites, save those changes before running the suite. Otherwise, the last saved version of the suite will run.
    - View more details about the latest result by clicking **View details** in the banner. Viewing details is possible after a test suite runs. For more information, see Viewing unit test suite run results.
    - To view historical information about previous test runs, such as test date, the run time, expected run time, and whether test passed or failed, click **View previous runs**.
11. Click **Save**. If you are saving the form for the first time, you can modify the Identifier. After you save the rule form, you cannot modify this field.

- Creating unit test cases for rules

Opening a unit test suite

You can view a list of the unit test suites that have been created for your application and select the one that you want to open.

1. In the header of Dev Studio, click **Configure Application Quality Automated Testing Unit Testing Test Suites** .
2. In the **Test suite name** column, click the test suite that you want to open.

- Running a unit test case
- Managing unit tests and test suites

Running a unit test suite

You can run a unit test suite to validate rule functionality by comparing the expected value to the output produced by running the rule. Test cases are run in the order in which they appear in the suite.

1. In the header of Dev Studio, click **Configure Application Quality Automated Testing Unit Testing Test Suites** .
2. Select the check box for each test suite that you want to run.
3. Click **Run selected**. The test cases run, and the **Result** column is updated with the result, which you can click to open test results.

    You can stop the test run by clicking **Stop test execution**.

    Note: The test suite continues to run even if you close or log out of the Pega Platform, close the Automated Testing landing page, or switch to another Dev Studio tab.

- Managing unit tests and test suites

Viewing unit test suite run results

After you run a unit test suite, you can view the test run results. For example, you can view the expected and actual output for assertions that did not pass.

1. In the header of Dev Studio, click **Configure Application Quality Automated Testing Unit Testing Test Suites** .
2. In the **Run history** column, click **View** for the test suite that you want to view.
   Note: To quickly view results of the most recent run, click the result in the **Result** column.
3. In the **Test suite runs log** dialog box, click the row for the instance of the test suite run that you want to view to open the results of that run in a new tab in Dev Studio.
   Note: You can also view test results after you run the test in the Edit Test Suite rule form.

- Running a unit test suite
- Managing unit tests and test suites

Adding cases to a test suite

You can add test cases to a unit test suite. When you run a test suite, the test cases are run in the order in which they appear in the suite.

1. **Optional:** Open the Pega unit test suite, if it is not already open.
2. Click **Add test cases**.
3. In the **Add test cases** dialog box, select the test cases that you want to add to the test suite.
   Note: You can click the **Advanced filter** icon to filter information by multiple criteria.
4. Click **Add**.
5. Save the rule form.

- Grouping test cases into suites

Viewing unit test suite run results

After you run a unit test suite, you can view the test run results. For example, you can view the expected and actual output for assertions that did not pass.

1. In the header of Dev Studio, click **Configure Application Quality Automated Testing Unit Testing Test Suites** .
2. In the **Run history** column, click **View** for the test suite that you want to view.
   Note: To quickly view results of the most recent run, click the result in the **Result** column.
3. In the **Test suite runs log** dialog box, click the row for the instance of the test suite run that you want to view to open the results of that run in a new tab in Dev Studio.
   Note: You can also view test results after you run the test in the Edit Test Suite rule form.

- Running a unit test suite
- Managing unit tests and test suites

Setting up and cleaning the context for a test case or test suite

You can set up the environment and conditions required for running a test case, determine how to clean up test data at the end of the test run, and set pages on which to automatically run rules.

You can set clipboard pages, apply data transforms, load data pages, execute activities, create and load objects. All the referenced data pages, and data objects and user pages that were created during a test run will be automatically removed at the end of each run. To further clean up the clipboard, add steps to apply additional data transforms and execute activities. You can set up or clean up the clipboard if you are running a test for which the output or execution depends on other data pages or information.

For example, your application contains a data page *D_AccountTransactionsList*. This data page is sourced by a report definition or activity that loads the transactions of the logged-in customer, based on the account type for which the customer views transactions.

The customer number and account type that the customer selects are dynamic properties that are stored on the work page of the case. The report definition or activity retrieves these properties as parameters from the work page and filters the results as it obtains the results for the data page.

When you create a test case for *D_AccountTransactionsList*, ensure that one of the following conditions is met:

- The parameter properties are on the work page of the clipboard before running the data page test.
- Your data page has an activity or report definition that refers to the properties of another data page that is on the clipboard to filter the results.

The system always runs data transforms, activities, and strategies on the *RunRecordPrimaryPage* page, regardless of which page you chose when you unit tested the rule in the **Run** dialog box. The system also runs flows and case types on

the *pyWorkPage*. To update the page with any information required to set up test data, click the **Setup** tab.

- Setting up your test environment

  Configure which actions you want to run and which objects and pages you want to view on the clipboard before, during, and after the test is run.

- Cleaning up your test environment

  After you run a unit test case or test suite, user and data pages used to set up the test environment and the parameter page are automatically removed by default. You can apply additional data transforms or activities to remove other pages or information on the clipboard before you run more test cases or suites.

Setting up your test environment

Configure which actions you want to run and which objects and pages you want to view on the clipboard before, during, and after the test is run.

To set up the environment and conditions that are required before running this test case, copy or create clipboard pages, apply data transforms, load data pages, execute activities, and create and load objects. Then, define the connections to data pages or third-party databases to simulate during the test. Finally, after running the test case, set up the environment and conditions that are required by applying data transforms, loading data pages, executing activities, and creating and loading objects.

Before you begin: Open the test case or test suite that you want to set up. For more information, see Opening a unit test case or Creating unit test suites.

1. Click the **Setup & Cleanup** tab.
2. **Optional:** To make specific conditions available during test execution, expand the **Before rule execution** section, and then configure the conditions:
   a. Copy or create clipboard pages.
      For more information, see Copy or create clipboard pages.
   b. Add additional clipboard data.
      For more information, see Add additional clipboard data.
3. **Optional:** To define simulation settings for the test, expand the **Simulation** section, and then configure the simulations.
   For more information, see Simulating data pages and third-party connections.
4. **Optional:** To make specific conditions available after test execution, expand the **After rule execution** section, and then add additional clipboard data.
   For more information, see Adding additional clipboard data.**Note:** You can set up actions after rule execution for test cases only.
5. To run the rule under on a page and avoid copying the entire page to *RunRecordPrimaryPage*, in the **Advanced** section, enter the page under which you want to run the rule.
6. Click **Save**.

- Copying and creating clipboard pages in setup

  When setting up your test environment, you can set to copy or create clipboard pages before the test runs.

- Adding additional clipboard data

  When setting up your test environment, you can add additional clipboard data before or after the test runs. You can apply data transforms, load data pages, execute activities, load objects, create data objects, and create work objects.

- Simulating data pages and third-party connections

  When setting up your test environment, you can simulate data pages and third-party connections. Such simulations let you run your tests without depending on the availability of third-party servers.

Copying and creating clipboard pages in setup

When setting up your test environment, you can set to copy or create clipboard pages before the test runs.

1. On the **Setup & Cleanup** tab for the test case or test suite for which you want to set up the context, expand the **Before rule execution** section, and then expand the **Setup data** section.
2. Click **Add data**.
3. In the **Description** box, enter a description of the clipboard page you want to copy or create.
4. **Optional:** Copy a clipboard page:
   a. In the **Type** section, select **Copy page**.
   b. To copy a clipboard page from a different thread, click the current thread name, and then click desired thread name.
   c. Select the check box next to the page that you want to be available in the clipboard during test execution, and then click **Next**.
   d. Edit the clipboard page and then click **OK**. You can rename the parent page, modify the values of existing

properties or add new properties and their values to the parent page and child pages.

5. **Optional:** Create a clipboard page:
    a. In the **Type** section, select **Create page**.
    b. In the **Page Name** field, enter a name of the page you want to create.
    c. In the **Class** field, enter or select the class of the page you want to create.
    d. Click **Next**.
    e. Edit the clipboard page and then click **OK**. You can rename the parent page, modify the values of existing properties or add new properties and their values to the parent page and child pages.
6. Save the test case or test suite.

Adding additional clipboard data

When setting up your test environment, you can add additional clipboard data before or after the test runs. You can apply data transforms, load data pages, execute activities, load objects, create data objects, and create work objects.

1. On the **Setup & Cleanup** tab for the test case or test suite for which you want to set up the context, choose whether to add the data before or after the rule runs.
    - To add the data before the test rule runs, select **Before rule execution** section, expand the **Additional clipboard data** subsection.
    - To add the data after the test rule runs, select **After rule execution**.
2. For each action you want to perform to the clipboard data, click **Add step** and then select the action.
    - To apply a data transform:, select **Apply data transform**, and then, in the **Name** field, enter or select the name of the data transform to apply.
    - To load a data page, select **Load data page**, and then, in the **Name** field, enter or select the name of the data to apply.
    - To execute an activity, select **Execute activity**, and then, in the **Name** field, enter or select the name of the activity to apply.
    - To load an object, select **Load object**, enter or select the class of the object in the **Of class** field, and then enter a name for the page in the **Load on page** field.
    - To create a data object, select **Create data object**, enter or select the class of the object in the **Of class** field, and then enter a name for the page in the **Load on page** field.
    - To create a work object, select **Create work object**, and then, in the **Of class** field, enter or select the class of the work object.
3. **Optional:** If parameters are configured on rules, then you can modify them by clicking the gear icon, and providing values in the **Configure parameters** dialog box, and then clicking **Submit**.
4. **Optional:** If keys are configured on loaded or created objects, then you can define their values by clicking the **With Keys** gear icon, and providing values in the **Configure parameters** dialog box.
5. Save the test case or test suite.

Simulating data pages and third-party connections

When setting up your test environment, you can simulate data pages and third-party connections. Such simulations let you run your tests without depending on the availability of third-party servers.

1. In Dev Studio, open a Pega unit test case.
2. Click the **Setup & Cleanup** tab.
3. In the **Setup** section, expand the **Simulation** section, and then click **Add rules**.
4. To include a rule that the test rule directly references, on the **Referenced rules** tab, select the rules to simulate, and then click **Add**.
5. To include any rule that the test rule does not directly reference, do the following for each rule:
    a. Click the **Other rules** tab and then click **Add**.
    b. In the **Rule type** list, click the type of rule that you want to simulate.
    c. In the **Class** box, enter the class of the rule that you want to simulate.
    d. In the **Rules** field, enter the rule that you want to simulate.
    e. Click the **Add** button.

    The selected rules display in the **Simulation** section on the **Setup & Cleanup** tab.
6. In the **Simulate with** list for each rule, click a simulation method:
    - Select **As defined in the rule** to use the default simulation defined in the rule.
    - Select **Select datatransform rule** to define your own data transform rule. You can reuse this rule in other test cases.
    - Select **Define data here** to manually provide test data specific to this particular test case. You can copy pages from the clipboard or create new pages and populate them with required test data.
    - Select **None** to disable the simulation.

- Understanding unit test cases

Cleaning up your test environment

After you run a unit test case or test suite, user and data pages used to set up the test environment and the parameter page are automatically removed by default. You can apply additional data transforms or activities to remove other pages or information on the clipboard before you run more test cases or suites.

Before you begin: Open the unit test case. For more information, see Opening a unit test case.

1. Click the **Setup & Cleanup** tab.
2. To keep the test data on the clipboard at the end of the test run, clear the **Cleanup the test data at the end of run** check box in the **Cleanup** section.
3. In the **Cleanup** section, click **Add step**.
4. Select **Apply data transform** or **Execute activity**, and then provide the appropriate data transform or activity in the next field.
5. If parameters are configured on the rule, you can configure them by clicking the **Parameters** link and providing values in the **Configure parameters** dialog box.
6. **Optional:** Provide additional information in the **Enter comments** field.
7. Click **Save**.

- Test environment cleanup

  After you run a unit test case or test suite, user and data pages used to set up the test environment and the parameter page are automatically removed by default.

- Creating unit test cases for flows and case types

Test environment cleanup

After you run a unit test case or test suite, user and data pages used to set up the test environment and the parameter page are automatically removed by default.

Note: You can override this behavior if you want the data from the current test to be available to the subsequent tests.

You can also apply additional data transforms or activities to remove other pages or information on the clipboard before you run more tests. Cleaning up the clipboard ensures that data pages or properties on the clipboard do not interfere with subsequent tests. For example, when you run a test case, you can use a data transform to set the values of the pyWorkPage data page with the AvailableDate, ProductID, and ProductName properties.

You can use a data transform to clear these properties from the pyWorkPage. Clearing these values ensures that, if setup data changes on subsequent test runs, the test uses the latest information. For example, if you change the value of the AvailableDate property to May 2018, you ensure that the data page uses that value, not the older (December 2018) information.

- Cleaning up your test environment
- Setting up and cleaning the context for a test case or test suite

Viewing unit test reports

View a graph with test pass rate trend data, a summary of Pega unit tests that were run, and an overview of Pega unit test compliance for currently included applications on the **Reports** tab on the Unit Testing landing page.

By default, a test case is considered as executed if it ran in the last 7 days. You can change the number of days for which a test can be considered executed on the Application: Quality Settings landing page. The overview also includes the percentage of rules on which Pega unit test cases are configured. View Pega unit test reports to check the quality of your application and identify rules that did not pass Pega unit testing.

1. In the header of Dev Studio, click **Configure Application Quality Automated Testing Unit Testing Reports** .
2. **Optional:** To filter information by more than one criterion, click the **Advanced filter** icon.
3. **Optional:** Generate and export a report for test coverage and test runs for a rule type.
   a. For the rule type for which you want to export a report, click a number in the column of the table for either pie chart.
   b. Click **Actions**.
   c. Click **Export to PDF** or **Export to Excel**.

- Changing application quality metrics settings
- Viewing application quality metrics
- Managing unit tests and test suites
- Viewing unit tests without rules

Viewing unit tests without rules

On the Application: Unit testing landing page you can display a list of unit tests that are not associated with any rule and export this list to an XLS or a PDF file. You should deactivate these unit tests because they will always fail.

1. In the header of Dev Studio, click **Configure Application Quality Automated Testing Unit Testing** .
2. Click **Tests without rules**.
3. **Optional:** Generate and export a report that contains a list of test cases that are not associated with any rules.
   - To export to PDF format, click **Actions Export to PDF**
   - To export to XLS format, click **Actions Export to Excel**

- Managing unit tests and test suites
- Viewing unit test reports

Understanding Pega Platform 7.2.2 and later behavior when switching between Pega unit testing and Automated Unit Testing features

Beginning with Pega 7.2.2, you can use Pega unit testing to create test cases to validate the quality of your application by comparing the expected test output with results that are returned by running rules. In previous versions, you used Automated Unit Testing (AUT) to create test cases.

If you have the *AutomatedTesting* privilege, you can use Automated Unit Testing (AUT) and switch between Pega unit testing and AUT, for example, if you want to view test cases that you created in AUT. The following list describes the application behavior when you use Pega unit testing and AUT:

- When you unit test activities that are supported by both Pega unit testing and AUT, the **Run Rule** dialog box displays updated options for creating unit tests for Pega unit testing. However, you cannot create unit test cases for AUT by using this dialog box.
- When you use Pega unit testing, you can create, run, and view the results of Pega unit testing on the **Test Cases** tab for the supported rule types.
- You can view, run, and view the results of Pega unit test cases by clicking **Dev Studio Automated Testing Test Cases** . You can also switch to the AUT landing page by clicking **Switch to old version**.
- When you switch to the AUT landing page, you can create, run, and view the results of unit test cases for AUT on the **Test Cases** tab for activities, data transforms, and data tables, which are supported by both Pega unit testing and AUT. You can create unit test cases only by clicking the **Record test case** button and using the older **Run Rule** dialog box.
- In the Automated Unit Testing landing page, you can restore the Automated Rule Testing landing page by clicking **Switch to new version**. When you click the **Test cases** tab in an activity, decision table, or decision tree, the tab displays options for creating Pega unit test cases.
- If you use the Automated Unit Testing landing page, and then log out of the system, Dev Studio displays the **Dev Studio Application Automated Unit Testing** > menu option instead of the **Dev Studio Application Automated Testing** > option. To return to the Automated Unit Testing landing page, click **Switch to new version** on the Automated Unit Testing landing page.

Working with the deprecated AUT tool

In older versions of Pega Platform, automated unit tests were created using the Automated Unit Testing (AUT) tool, which has since been replaced by PegaUnit testing. If you have automated unit tests that were created using AUT and they haven't been changed to PegaUnit test cases, then you can switch back to AUT to manage those tests.

Note: AUT has been deprecated and is not supported in the current version of Pega Platform. Switch to and use AUT only if you have existing automated unit tests created with AUT. See PegaUnit testing more information.

Note the following behavior:

- To use AUT, your operator ID must have the *AutomatedTesting* privilege through an access role.
- Switch from PegaUnit testing to AUT by clicking **Configure Automated Testing Test Cases** and clicking **Switch to old version** on the Automated Unit Testing landing page.
- Click the **Test cases** tab of the Automated Unit Testing landing page to display options for creating unit tests for activities, decision tables, and decision trees.
- If you are using the Automated Unit Testing landing page and then log out of the system, you can click **Configure Application Automated Unit Testing** , and then click **Switch to new version** to restore the Automated Testing landing page.

Viewing, playing back, and rerecording test cases

1. Click the **Automated Unit Tests** tab.
2. Select **Unit Test Cases** in the **Show** field.
    - To play back a test case, click its name in the **Name** column.
    - To rerecord a test case, right-click the test case name and click **Re-record**. **Note:** If the underlying test case rule belongs to a ruleset that uses the check-out feature, you must have the test case rule checked out to you before re-recording the test case.

Opening rules in test cases and unit test suites

1. Click the **Automated Unit Tests** tab.
2. Right-click a test case or suite and click **Open** to open its rule.

Withdrawing test cases and unit test suites

1. Click the **Automated Unit Tests** tab.
2. Right-click a test case or suite and click **Withdraw**.

Withdrawn test cases and suites are not displayed on the Automated Unit Tests tab.

Unit test suite run results

You can view the results of your recent unit test suite runs in either the **Dashboard** tab or **Reports** tab. The **Dashboard** tab displays the ten most recent runs. The **Reports** tab displays earlier results and, for a given unit test suite, shows results from the last fifty (50) runs of that unit test suite.

If you ran a unit test against a saved test case for a decision table, decision tree, activity, or Service SOAP rule and selected the **All Cases** option in the **Run Rule** form, those results also appear in the **Dashboard** tab.

For activity test cases, if the activity test case has an approval list, differences are reported only for pages and properties on the list. If the test case has an approval list and the only differences are for pages and properties not on the list, those differences are not reported. If differences are found for items on the approval list, you can remove the item from the approval list for that test case.

Creating and scheduling unit test suites

To create a unit test suite:

1. Click the Schedule tab.
2. Click **Create Suite**.
3. In the New Rule form, enter the requested information for creating a unit test suite.

To run a unit test suite or to schedule a run:

1. Click the **Schedule tab**.
2. Click the **Calendar** icon in the **Schedule** column for the unit test suite you want to run.
3. In the **Pattern** section in the **Schedule Unit Test Suite** window, specify how to run this unit test suite. When the run is complete, the system displays the results in the **Dashboard** tab. When you select **To run immediately**, the system runs the test suite in the foreground; for all other options, the system runs the test in the background.
4. For scheduled runs, you can specify additional options.
    1. Select to run the unit test suite by using a different operator ID. In the **Advanced Settings** section, enter the Operator ID in the **Override Default and Run Suite As** field. The system runs the unit test suite by using the rulesets and access rights associated with that operator. If the operator ID form has multiple access groups, the default access group is used.
    2. Send the completion email to multiple email addresses. Use the **Send Completion Email to** field to specify the email addresses.

       If you do not want any emails sent, clear the **Send Completion Email** field.
5. Click **OK**.

By default, the Pega-AutoTest agents run scheduled unit test suites run every five minutes. When the suite is finished, the agent activity sends an email with the results. By default, this email is sent to the operator who requested the unit test suite run and to any email addresses listed in the **Send Completion Email** array. If no email addresses appear in that field, no email message is sent.

- Create or Save as form in AUT test suite

  Unit Test Suites – Completing the Create or Save As form

- Creating unit test suites

  To create a unit test suite, add test cases and test suites to the suite and then modify the order in which you want them to run. You can also modify the context in which to save the scenario test suite, such as the development branch or the ruleset.

- Contents form in AUT test suite
- Managing unit tests and test suites

Creating test cases with AUT

You can automate testing of rules by creating test cases for automated unit testing. Automated unit testing validates application data by comparing expected output to the actual output that is returned by running rules.

To create test cases, you must have a rule set that can store test cases. For more information, see Creating a test ruleset to store test cases.

Note: AUT is deprecated. Use PegaUnit testing instead to create automated test rules.

Automated unit testing information is available on the Testing Applications landing page on Pega Community and in the automated unit testing topics in the help.

- Creating unit test suites

  To create a unit test suite, add test cases and test suites to the suite and then modify the order in which you want them to run. You can also modify the context in which to save the scenario test suite, such as the development branch or the ruleset.

AUT test cases

Create test cases for automated unit testing to validate application data by comparing expected output to the actual output that is returned by running rules.

You can create automated unit testing test cases for the following rule types:

- Activity
- Decision table
- Decision tree
- Flow
- Service SOAP

Note: AUT is deprecated. Use PegaUnit testing instead to create automated test rules.

Automated unit testing information is available on the Testing Applications landing page on Pega Community and in the automated unit testing topics in the help.

- Creating test cases with AUT

Creating unit test suites with AUT

Unit Test Suites identify a collection of Test Cases and their rulesets, and a user (Operator ID) whose credentials are used to run the Unit Test Suite. Unit Test Suites are used to automatically run groups of test cases together and make unit testing more efficient.

The **Unit Test Suite** rule form consists of the Contents tab.

Note: AUT is deprecated. Use PegaUnit testing instead to create automated test rules.

Automated unit testing information is available on the **Testing Applications** landing page on Pega Community and in the automated unit testing topics in the help.

You must have the *AutomatedTesting* privilege to work with unit test suite rules.

You can create a unit test suite that includes all the test cases for a specific rule type, or you can select individual rules and specify the sequence in which to run them.

To run a unit test suite, use the **Schedule** gadget on the Automated Unit Testing landing page. For more information, see Working with the deprecated AUT tool. From that gadget, you can choose to run the unit test suite immediately or schedule the run for a future time.

For unit test suites that are scheduled to run at future times, an agent activity in the *Pega-AutoTest* agents rule checks for unit test suite requests every five minutes and runs those that are due. When the agent activity finishes running a unit test suite, it sends an email message with the results. By default, this completion email message is sent to the person who scheduled the unit test suite run, and to any additional email addresses specified at the time the run is scheduled. If no email addresses are specified at the time the run was scheduled, no email message is sent.

Access

For more information, see Working with the deprecated AUT tool to work with the Unit Test Suites that are available to you. You can:

- Use the **Automated Unit Tests** gadget to see the Unit Test Suites and the test cases in each.
- Use the **Create Suite** button on the **Schedule** gadget to create unit test suites.
- Use the calendar button on the **Schedule** gadget to run unit test suites immediately and to schedule unit test suite runs.

Category

Unit Test Suites are instances of the *Rule-AutoTest-Suite* class. They belong to the SysAdmin category.

- Creating test cases with AUT

You can automate testing of rules by creating test cases for automated unit testing. Automated unit testing validates application data by comparing expected output to the actual output that is returned by running rules.

- Contents form in AUT test suite

  Use the Contents tab to define the unit test suite. Specify a user (Operator ID) that the Pega-AutoTest agents are to use by default when running the suite, and select the test cases to include.

Create or Save as form in AUT test suite

Unit Test Suites – Completing the Create or Save As form

To create a unit test suite rule, use the **Create Suite** button on the *Schedule* gadget of the Automated Unit Testing landing page. To open the Automated Unit Testing landing page, select Dev Studio *> Application > Automated Unit Testing*.

You must have the *AutomatedTesting* privilege to be able to create unit test suites. For information about how to enable this privilege, see About Automated Unit Testing.

A unit test suite rule has a single key part, the unit test suite name:

Field	Description
Name	Enter a short, descriptive name for the unit test suite.

Create a separate RuleSet to hold test cases and unit test suites, rather than using a RuleSet that will be moved to a production system. For more information, consult the articles in the *Testing Applications* category of Pega Community.

For general information about the Create and Save As forms, see:

- Creating a rule.
- Copying a rule or data instance .

> Rule resolution

As with most rules, when you search for a Unit Test Suite, the system shows you only those rules that belong to a RuleSet and version that you have access to.

Unit Test Suite rules cannot be qualified by circumstance or time.

Creating unit test suites with AUT

Contents form in AUT test suite

Use the **Contents** tab to define the unit test suite. Specify a user (Operator ID) that the *Pega-AutoTest* agents are to use by default when running the suite, and select the test cases to include.

The Operator ID specified here is the default one used to run the unit test suite. When defining the unit test suite's run schedule using the *Schedule* gadget of the Automated Unit Testing landing page, you have the option to specify a different Operator ID and override the one specified here.

You can specify Test Cases in both the **Rule Types To Include** section and the **Query Test Cases To Include** section of this form. If you specify Test Cases in both sections, when the unit test suite runs, those test cases defined in the **Rule Types To Include** section will run before the test cases in the **Query Test Cases To Include** section.

1. In the **RuleSets for Test Cases** field, select the RuleSet that holds the test cases you want to include in this test suite.
   If the test cases are in more than one RuleSet, click the **Add** icon to add rows to specify the additional RuleSets.
2. In the **User ID for Agent Processing** field, select the Operator ID for the *Pega-AutoTest* agents to use by default when they run this test suite.
   This ID must provide access to the RuleSet that this test suite belongs to, as well as access to the RuleSets listed in the RuleSets field.
3. **Optional:** To specify that the work items created during the test case execution are to be deleted afterwards, select the **Remove Test Work Objects?** check box.

   The fields in the **Application Test Cases To Include** section provide options to specify the test cases by application name and version.

4. In the **Application Name** field, select the name of the application that has the test cases you want to include in the unit test suite.
5. In the **Application Version** field, select the version of the application that has the test cases you want to include in the unit test suite.

   The fields in the **Rule Types To Include** section provide options to select the test cases by rule type. You can

specify that all the test cases for a particular rule type are included in this unit test suite, or you can constrain the list with a When condition rule.

6. In the **Rule Type** field, select those rule types for which you want to include their test cases in this unit test suite:
   - **Activities**
   - **Decision Tables**
   - **Decision Trees**
   - **Flows**
   - **Service SOAP service records**
7. In the **When Filter** field, do one of the following:
   - Leave blank to include all the test cases that were created for rules of the type specified in the Rule Type field.
   - Select the appropriate when condition rule to constrain the list.

     The test cases that meet the conditions in the when condition rule are included in the unit test suite

   The fields in the **Query Test Cases To Include** section provide options to select specific Test Cases to include in this unit test suite. List the test cases in the order in which you want them to be run.

8. In the **Test Case Name** field, enter a search string for the test case you want to find.
9. To list test cases that match the query string in the **Test Case Name** field, click **Query**.
   The list is not limited by RuleSet. If test cases exist that match the search string, the **List Test Case** window appears. Select the test cases you want to include and then click **OK**. The test cases are added to the list in this section of the form.
10. In the **Test Case Key** field, enter the three-part key of a Test Case rule.
    The key consists of the following parts:
    - Class Name
    - Instance Name (Ins Name)
    - Purpose

    When you use the **Query** button to find and add a test case, the system automatically fills in this field.
11. In the **Description** field, enter the short description of the Test Case.
    When you use the **Query** button to find and add a test case, the system automatically fills in this field.
12. In the **RuleSet** field, enter the RuleSet of the test case.
    When you use the **Query** button to find and add a test case, the system automatically fills in this field.
    Verify that this RuleSet is included in the RuleSets for Test Cases list at the top of this form. If the RuleSet for the test case is not in that list, add it now. Otherwise, the Test Case does not run when the unit test suite runs.

Running scenario tests against a user interface

Run scenario tests against a user interface to verify that the end-to-end scenarios are functioning correctly. The UI-based scenario testing tool allows you to focus on creating functional and useful tests, rather than writing complex code.

You can test either a specific case type or an entire portal by clicking **Scenario Testing** in the run-time toolbar to open the test recorder. When you use the test recorder and hover over a testable element, an orange highlight indicates that the element can be tested. Interactions are recorded in a visual series of steps and the execution of a test step can include a delay.

Provide data to your test cases with a predefined data page. This data page can provide unique values for each execution of the test case. You can populate the data page by using any source, including activities or data transforms.

Tests are saved in a test ruleset. After they are saved, tests are available on the scenario testing landing page. From the landing page you can run a test or view the results of a previous test run.

Testing applications in the DevOps pipeline

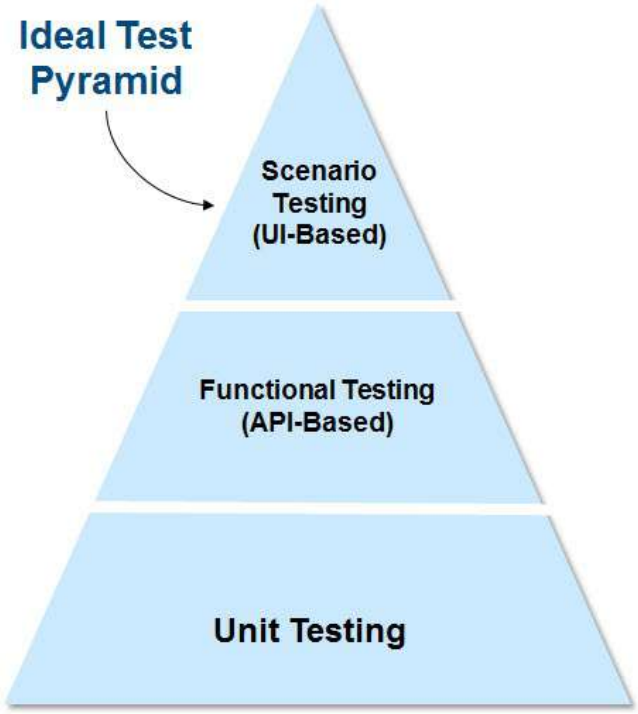
Having an effective automation test suite for your application in your continuous delivery DevOps pipeline ensures that the features and changes that you deliver to your customers are of high-quality and do not introduce regressions.

At a high level, the recommended test automation strategy for testing your Pega applications is as follows:

- Create your automation test suite based on industry best practices for test automation
- Build up your automation test suite by using Pega Platform capabilities and industry test solutions
- Run the right set of tests at different stages of your delivery pipeline
- Test early and test often

Industry best practices for test automation can be graphically shown as a test pyramid. Test types at the bottom of the pyramid are the least expensive to run, easiest to maintain, take the least amount of time to run, and should represent the greatest number of tests in the test suite. Test types at the top of the pyramid are the most expensive to run, hardest to maintain, take the most time to run, and should represent the least number of tests in the test suite. The higher up the pyramid you go, the higher the overall cost and the lower the benefits.

Ideal test pyramid



- Scenario testing on Pega platform
- Scenario testing as an add-on component

  As of Pega Platform 8.5 and newer, the scenario testing feature is being decoupled from the native Pega platform and is released as an independent component. See for release-specific enhancements and updates to the functionality.

Scenario testing on Pega platform

- Application: Scenario testing landing page

  The scenario testing landing page provides a graphical test creation tool that you can use to increase test coverage without writing complex code.

- Creating scenario tests

  Record a set of interactions for a case type or portal in scenario tests. You can run these tests to verify and improve the quality of your application.

- Opening a scenario test case

  You can view a list of the scenario test cases that have been created for your application and select the one that you want to open.

- Editing scenario tests

  Ensure that the test covers your current portal or case type scenario by updating an existing scenario test when the user interface or process flow changes. You can save time and effort by keeping existing tests functional instead of creating new ones.

- Grouping scenario tests into suites

  Group related scenario tests into test suites to run multiple scenario test cases in a specified order. You can then run the scenario test suites as part of purpose-specific tests, such as smoke tests, regression tests, or outcome-based tests. Additionally, you can disable or quarantine individual scenario tests for an application so that they are not executed when the test suite runs.

- Viewing scenario test results

  After you run a scenario test, you can view the test results. For example, you can view the expected and actual output for assertions that did not pass.

- Other features

Application: Scenario testing landing page

The scenario testing landing page provides a graphical test creation tool that you can use to increase test coverage

without writing complex code.

On the scenario testing landing page, you can view and run scenario test cases. By viewing reports, you can also identify case types and portals that did not pass scenario testing.

On the scenario testing landing page, you can perform the following tasks:

- View test execution and coverage information for case type tests and portal tests.
- Open a test case rule where you can add assertions to your test.
- View the results of the most recent test run.
- Select and run individual test cases, or group tests into test suites that you can use to run multiple tests in a specified order.
- Download a list of tests, their type, the name of a portal or case that is tested, and the time and the result of the last run.
- View the history of your test runs so that you can quickly analyze the history of all your test runs.
- View which scenario tests were automatically rerun if they failed.

- Creating scenario tests

Creating scenario tests

Record a set of interactions for a case type or portal in scenario tests. You can run these tests to verify and improve the quality of your application.

Before you begin: Create a test ruleset in which to store the scenario test. For more information, see Creating a test ruleset to store test cases.

1. Launch the portal in which you want to do the test.
2. In the **Scenario tests** pane, click **Create test case**, and then select the test type:
   - To record a test for a portal, select **Portal**.
   - To record a test for a case, select **Case type**, and then select the type of case for which you want to record the test.**Note:** When you select the case type, a new case of that type is created.
3. Record the steps for the test by clicking the user interface elements. **Result:** When you hover over a testable element, an orange highlight box appears. When you click an element, you record an implicit assertion and add the interaction to the list of test steps.
4. **Optional:** To add an explicit assertion to the test, do the following tasks:
   a. Hover over an element.
   b. Click the **Mark for assertion** icon on the orange highlight box.
   c. In the **Expected results** section, click **Add assertion**.
   d. Define the assertion by completing the **Name**, **Comparator**, and **Value** fields.
   e. Click **Save step**.
5. To delay the execution of a step, do the following tasks. You can delay the execution of a step to add latency to a web browser and actions on a web page to prevent tests from failing when there is complex processing or slow UI rendering.
   a. Click the **Mark for assertion** icon on the orange highlight box.
   b. In the **Wait** field, enter the number of milliseconds by which to delay the execution of the step.
   c. Click **Save step**.
   Note: To add delays to all steps in all scenario tests in your application, follow the procedure in Delaying scenario test execution.**Note:** Delaying the execution of a single step takes precedence over delaying all of the steps at the application level.
6. When you finish adding steps, in the **Test case** pane, click **Stop and save test case** .
7. On the **New test case** form, save the test:
   a. Enter a name and a description for the test.
   b. In the **Context** section, select a branch or ruleset in which you want to save the test.
   c. In the **Apply to** field, enter the name of a class that is relevant to the test.
   d. Click **Save**.
   Result: The test case appears on the Scenario testing landing page.

- Application: Scenario testing landing page

Opening a scenario test case

You can view a list of the scenario test cases that have been created for your application and select the one that you want to open.

1. In the navigation pane of Dev Studio, click **Configure Application Quality Automated Testing Scenario Testing Test Cases** .
2. In the **Test case name** column, click the test case that you want to open.

Editing scenario tests

Ensure that the test covers your current portal or case type scenario by updating an existing scenario test when the user

interface or process flow changes. You can save time and effort by keeping existing tests functional instead of creating new ones.

Before you begin: Create a scenario test case for a portal or case type. For more information, see Creating scenario tests

1. Launch the portal in which you want to do the test
2. Do one of the following to open the automation recorder:
   - In App Studio, on the lower-left side of the screen, click the **Test** icon.
   - In Dev Studio, on the lower-right side of the screen, toggle the run-time toolbar, and then click the **Toggle Automation Recorder** icon.
3. In the **Scenario tests** pane, click the name of the test that you want to edit, and then click **Edit**.
4. Update the test sequence by clicking the **More** icon next to a step, and then selecting an action:
   - To remove a step from the test case, click **Remove step**.
   - To add a step to the test case, click **Add steps**.

     The test runs from the start and stops at the selected step so that you can add steps to the specific part of the test sequence.

   - To record the test case again from a specific step, click **Re-record from here**.

     All steps after the selected step are removed. The test runs from the start and stops at the selected step so that you can add steps to the end of the test sequence.
5. **Optional:** To modify a step, click the **Edit** icon next to a step, modify the assertions and other properties of the step, and then click **Save step**.
6. Click **Save**.

What to do next: Your scenario test case now matches the current user interface and process flow. Run the test to check the quality of your current portal or case type scenario.

Grouping scenario tests into suites

Group related scenario tests into test suites to run multiple scenario test cases in a specified order. You can then run the scenario test suites as part of purpose-specific tests, such as smoke tests, regression tests, or outcome-based tests. Additionally, you can disable or quarantine individual scenario tests for an application so that they are not executed when the test suite runs.

- Creating unit test suites with AUT

  Unit Test Suites identify a collection of Test Cases and their rulesets, and a user (Operator ID) whose credentials are used to run the Unit Test Suite. Unit Test Suites are used to automatically run groups of test cases together and make unit testing more efficient.

- Running scenario test suites

  Run scenario test suites to check application functionality. You can check the run history, add or remove test cases from the suite, or reorder the test cases before running the suite.

- Viewing scenario test suite results

  After you run a scenario test suite, you can view the test results. For example, you can view the expected and actual output for assertions that did not pass.

- Creating scenario tests
- Application: Scenario testing landing page

Running scenario test suites

Run scenario test suites to check application functionality. You can check the run history, add or remove test cases from the suite, or reorder the test cases before running the suite.

1. In the header of Dev Studio, click **Configure Application Quality Automated Testing Scenario Testing Test Suites** .
2. **Optional:** View or modify the test cases included in a test suite.
   a. Click the name of the suite.
   b. View summary information about previous test results in the header.
      - To view more information about the latest test results, click **View details**.
      - To view information about earlier results, click **View previous runs**.
   c. Modify the test cases in the suite from the **Scenario test cases** section.
      - To remove test cases from the suite, click **Remove**, and then click **Save**.
      - To include additional test cases in the suite, click **Add**, select the test case, click **Add**, and then click **Save**.
   d. To change the order in which the test cases will run, drag a case to a different position in the sequence and then click **Save**.

    e. To prevent individual test cases from running as part of the suite, select the case, click **Disable**, click **Save**, and then close the test case.
    f. Close the test suite, return to the **Application: Scenario** testing page, and then click **Actions Refresh** .
3. **Optional:** To view details about previous test results, click **View** in the **Run history** column.
4. Select the check box for each test suite that you want to run and then click **Run selected**. The test suites run and the **Result** column is updated with the result, which you can click to open test results.

- Grouping scenario tests into suites
- Running scenario test suites
- Viewing scenario test suite results
- Creating scenario tests
- Application: Scenario testing landing page

Viewing scenario test suite results

After you run a scenario test suite, you can view the test results. For example, you can view the expected and actual output for assertions that did not pass.

1. In the header of Dev Studio, click **Configure Application Quality Automated Testing Scenario Testing Test Suites** .
2. To view results of the most recent run, click the result in the **Result** column. For information about why a test failed, click **Failed** in the **Result** column.
3. To view historical details about a specific test suite, in the **Run history** column, click **View**.

- Grouping scenario tests into suites
- Running scenario test suites
- Creating scenario tests
- Application: Scenario testing landing page

Viewing scenario test results

After you run a scenario test, you can view the test results. For example, you can view the expected and actual output for assertions that did not pass.

1. In the header of Dev Studio, click **Configure Application Quality Automated Testing Scenario Testing Test Cases** .
2. To view results of the most recent run, click the result in the **Result** column.
3. To view historical details about a specific test, in the **Run history** column, click **View**.

- Creating scenario tests
- Opening a scenario test case
- Application: Scenario testing landing page

Other features

- Delaying scenario test execution

  You can delay the execution of a test to add latency to a web browser and actions on a web page. This prevents tests from failing when a dynamic web page does not load all page elements at once, but the test finds the page elements that render immediately.

- Automatically rerunning failed scenario tests

  You can automatically rerun scenario test cases after they fail. Test cases can sometimes fail if there are temporary stability issues on your environment, or if there are first-time First Use Assembly (FUA) issues that could cause the application UI to be rendered slowly.

Delaying scenario test execution

You can delay the execution of a test to add latency to a web browser and actions on a web page. This prevents tests from failing when a dynamic web page does not load all page elements at once, but the test finds the page elements that render immediately.

Use this procedure to delay the execution of all the steps in all the scenario tests in an application. To delay the execution of a specific step, configure the time of delay on each step when you create or edit a scenario test.**Note:** Delaying the execution of a single step takes precedence over delaying all the steps at the application level.

1. In the header of Dev Studio, click **Configure Application Quality Settings** .
2. On the **Application: Quality Settings** page, in the **Scenario test case execution** section, select the **Configure delay for scenario test execution** check box.
3. In the **Milliseconds** field, enter the number of milliseconds by which to delay the execution of all the steps in all your scenario tests.

4. Click **Save**.

- Changing application quality metrics settings
- Creating scenario tests

Automatically rerunning failed scenario tests

You can automatically rerun scenario test cases after they fail. Test cases can sometimes fail if there are temporary stability issues on your environment, or if there are first-time First Use Assembly (FUA) issues that could cause the application UI to be rendered slowly.

To automatically rerun scenario tests that fail, do the following steps:

1. In the header of Dev Studio, click **Configure Application Quality Settings** .
2. On the **Application: Quality Settings** page, in the **Scenario test case execution** section, select the **Re-run tests automatically on failure** check box.
3. In the **Number of runs** list, select the maximum number of times to rerun failed scenario test cases.
4. Click **Save**.

Result: The Application: Scenario testing landing page indicates whether a scenario test has been rerun.

- Changing application quality metrics settings

Scenario testing as an add-on component

As of Pega Platform 8.5 and newer, the scenario testing feature is being decoupled from the native Pega platform and is released as an independent component. See Release notes for release-specific enhancements and updates to the functionality.

- Installing the add-on component

  If you are using scenario testing on-premises, you must first install it. You can download the add-on component from the Pega Marketplace.

- Release notes

  Release notes contain important information that you should know before you install a release of the scenario testing component. These notes describe the changes that are included in each component release. Before you install the component, familiarize yourself with the changes, new features, and resolved issues that are listed in respective release below.

Installing the add-on component

If you are using scenario testing on-premises, you must first install it. You can download the add-on component from the Pega Marketplace.

1. On each system, browse to the Pega Marketplace page, and then download the **PegaTestAutomationKit_1.5.zip** file for your version of the add-on component.
2. Use the Import wizard to import the **PegaTestAutomationKit_1.5.zip** into the appropriate systems. For more information about, see Importing rules and data by using the Import wizard.
3. After the import, add major version (1) of **PegaTestAutomationKit** as a built-on application in the target application app stack.

Release notes

Release notes contain important information that you should know before you install a release of the scenario testing component. These notes describe the changes that are included in each component release. Before you install the component, familiarize yourself with the changes, new features, and resolved issues that are listed in respective release below.

- Pega Test Automation Kit 1.3

  The Pega Test Automation Kit 1.3 release includes the following new features and resolved issues.

- Pega Test Automation Kit 1.2

  The Pega Test Automation Kit 1.2 release includes the following new features and resolved issues.

- Pega Test Automation Kit 1.1

  The Pega Test Automation Kit 1.1 release includes the following new features and resolved issues.

Pega Test Automation Kit 1.3

The Pega Test Automation Kit 1.3 release includes the following new features and resolved issues.

Enhancements

The following new features are available in this release:

- **Alert on new version of Pega Test Automation Kit.**
  - The scenario testing landing page now provides an announcement when a new Pega Test Automation Kit is released.
- **Attachment content control support in scenario testing.**
  - Scenario tests now support the attach content control during recording. This control allows users to select documents or images and attach them to their application by either browsing in a local directory for a file or dragging and dropping files on to the attachment window. For more information, see Attachment control support in scenario testing.
- **More operators for string data type in scenario testing.**
  - All string attributes (value, placeholder, label, tooltip, and button caption) now include new operators for comparing values in the validations.

Resolved issues

The following issues were resolved in this release:

- **Not enough information when scenario test execution fails from a pipeline.**
  - Pega Test Automation Kit now returns appropriate failure or exception messages if a scenario test does not execute successfully when invoked from a CI pipeline.
- Attachment control support in scenario testing

  Scenario tests now support the attach content control, pxAttachContent, during recording. This allows you to attach images or files directly into your case and record it for testing and playback.

Attachment control support in scenario testing

Scenario tests now support the attach content control, pxAttachContent, during recording. This allows you to attach images or files directly into your case and record it for testing and playback.

1. Launch App Studio or Dev Studio, based on where you want to perform the test.
2. Do one of the following to open the automation recorder:
   - In App Studio, on the lower-left side of the screen, click the **Test** icon.
   - In Dev Studio, on the lower-right side of the screen, toggle the run-time toolbar, and then click the **Toggle Automation Recorder** icon.
3. In the **Scenario tests** pane, click **Create test case**, and then select the test type:
   - To record a test for a portal, select **Portal**.
   - To record a test for a case, select **Case type**, and then select the type of case for which you want to record the test.**Note:** When you select the case type, a new case of that type is created.
4. Record the steps for the test by clicking the UI elements. **Result:** When you hover over a testable element, an orange highlight box appears. When you click an element, you record an implicit assertion and add the interaction to the list of test steps.
5. Click on **Upload file** to open the attachment dialogue page.
   a. Navigate to the file you want to upload and select it.
   b. Enter a name for the attachment in the **Enter name** field.
   c. Enter a description for the attachment in the **Enter description field.**
   d. Click **Publish**.
6. Continue recording as many steps as necessary. When you finish adding steps, in the **Test case** pane, click **Stop and save test case**. **Result:** The test case appears on the **Scenario testing landing page**.

Pega Test Automation Kit 1.2

The Pega Test Automation Kit 1.2 release includes the following new features and resolved issues.

Enhancements

The following new features are available in this release:

- Scenario tests now support the Anypicker and Multi-select controls in inspectable text fields during recording. Anypicker is used to enhance navigation in your scenario test by grouping drop-down list items into expandable categories, and Multi-select allows you to record more than one value for a single field on a form.
- A new Setup & Cleanup tab has been added, which you can use to create or fetch test data using one of the setup options available from the drop down menu. You can apply a data transform, execute an activity, or load an object that your scenario test case can create or fetch dynamically.

- A cleanup feature has been added to automatically remove all referenced data pages, data objects, and user pages at the end of each test run. See Setting up and cleaning the context for a scenario test for more information.

- Anypicker and Multi-select support in scenario testing

  Scenario tests now support multiple new controls in text fields to include both Anypicker and Multi-select. Anypicker is used to enhance navigation in your scenario test by grouping drop-down list items into expandable categories, and Multi-select allows you to record more than one value for a single field on a form.

- Setting up and cleaning the context for a scenario test

  Scenario tests can now apply dynamic data to text fields for real-time application of information in a UI recording.

Anypicker and Multi-select support in scenario testing

Scenario tests now support multiple new controls in text fields to include both Anypicker and Multi-select. Anypicker is used to enhance navigation in your scenario test by grouping drop-down list items into expandable categories, and Multi-select allows you to record more than one value for a single field on a form.

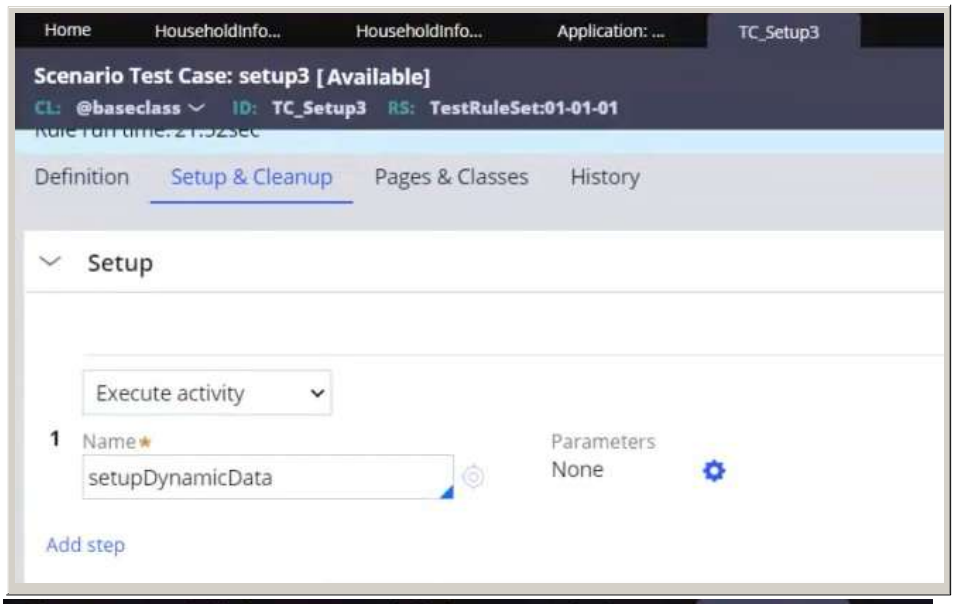
1. Launch App Studio or Dev Studio, based on where you want to perform the test.
2. Do one of the following to open the automation recorder:
   - In App Studio, on the lower-left side of the screen, click the **Test** icon.
   - In Dev Studio, on the lower-right side of the screen, toggle the run-time toolbar, and then click the **Toggle Automation Recorder** icon.
3. In the **Scenario tests** pane, click **Create test case**, and then select the test type:
   - To record a test for a portal, select **Portal**.
   - To record a test for a case, select **Case type**, and then select the type of case for which you want to record the test.**Note:** When you select the case type, a new case of that type is created.
4. Record the steps for the test by clicking the UI elements. **Result:** When you hover over a testable element, an orange highlight box appears. When you click an element, you record an implicit assertion and add the interaction to the list of test steps.
5. Click on an inspectable text field, denoted by an orange highlight.
   - If the pyTextArea is being used for Anypicker, use the dropdown menu to select the required input.
   - If the pyTextArea is being used for Multi-Select, add as many input values as necessary.
6. Continue recording as many steps as necessary. When you finish adding steps, in the **Test case** pane, click **Stop and save test case**. **Result:** The test case appears on the **Scenario testing landing page**.

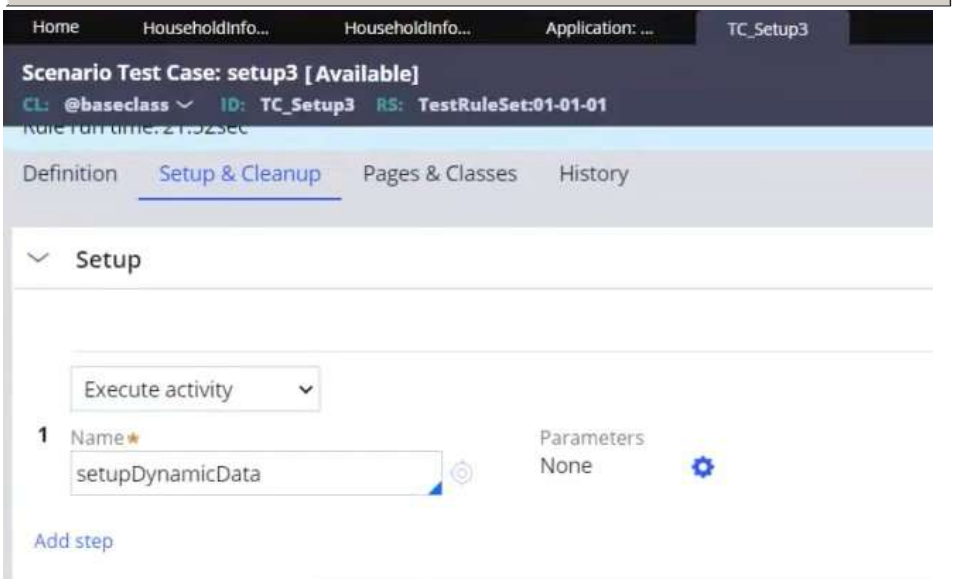
Setting up and cleaning the context for a scenario test

Scenario tests can now apply dynamic data to text fields for real-time application of information in a UI recording.

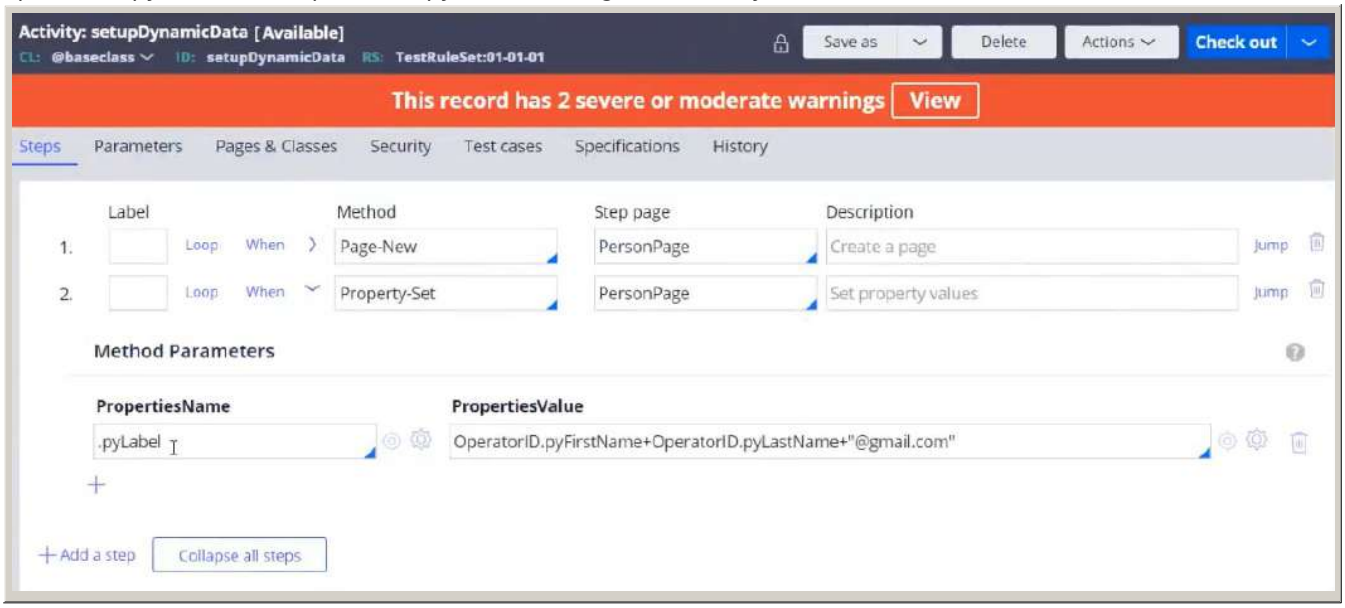
After an initial scenario test case has been recorded, any static text field can now be re-characterized as a dynamic field to apply referenced data using the editable scenario test case dashboard. A new **Setup & Cleanup** tab has been added to the scenario test case dashboard to set up dynamic data. To perform this action:

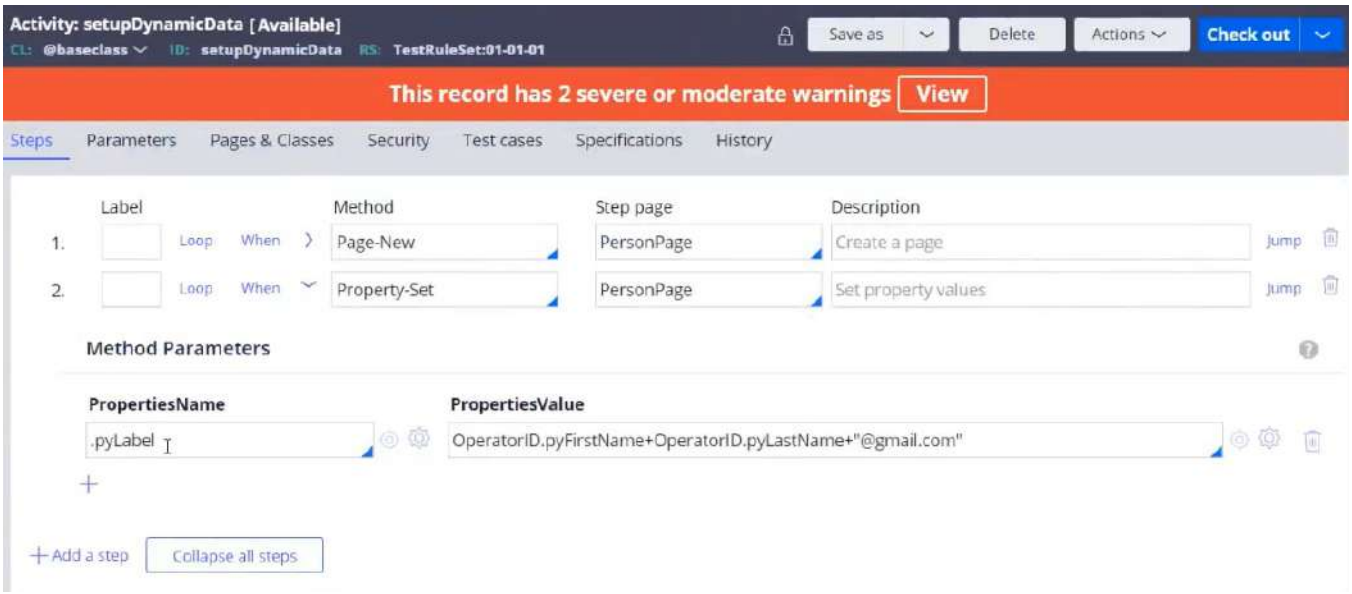
1. Record a scenario test using static data. See Creating scenario tests for more information.
2. In the navigation page of **Dev Studio** click **Configure Application Quality Automated Testing Scenario Testing Test Cases** .
3. Select the test case for which you want to edit.
4. Navigate to the **Setup & Cleanup** tab of the test case.
5. In the Setup section of this tab, you can create or fetch test data using one of the setup options available from the drop down menu. You can apply a data transform, execute an activity, or load an object. Scenario tests can fetch and use any of this setup information as dynamic input data.
6. Input a name for the activity.



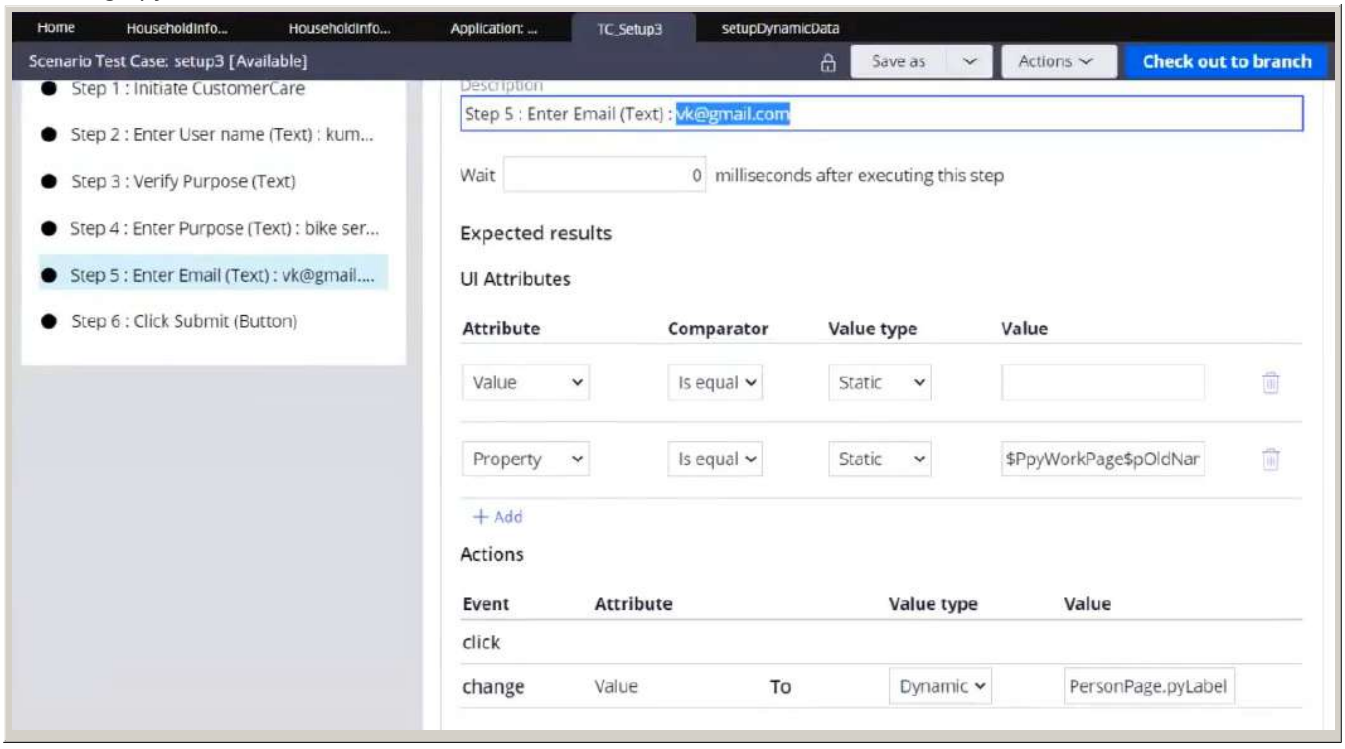


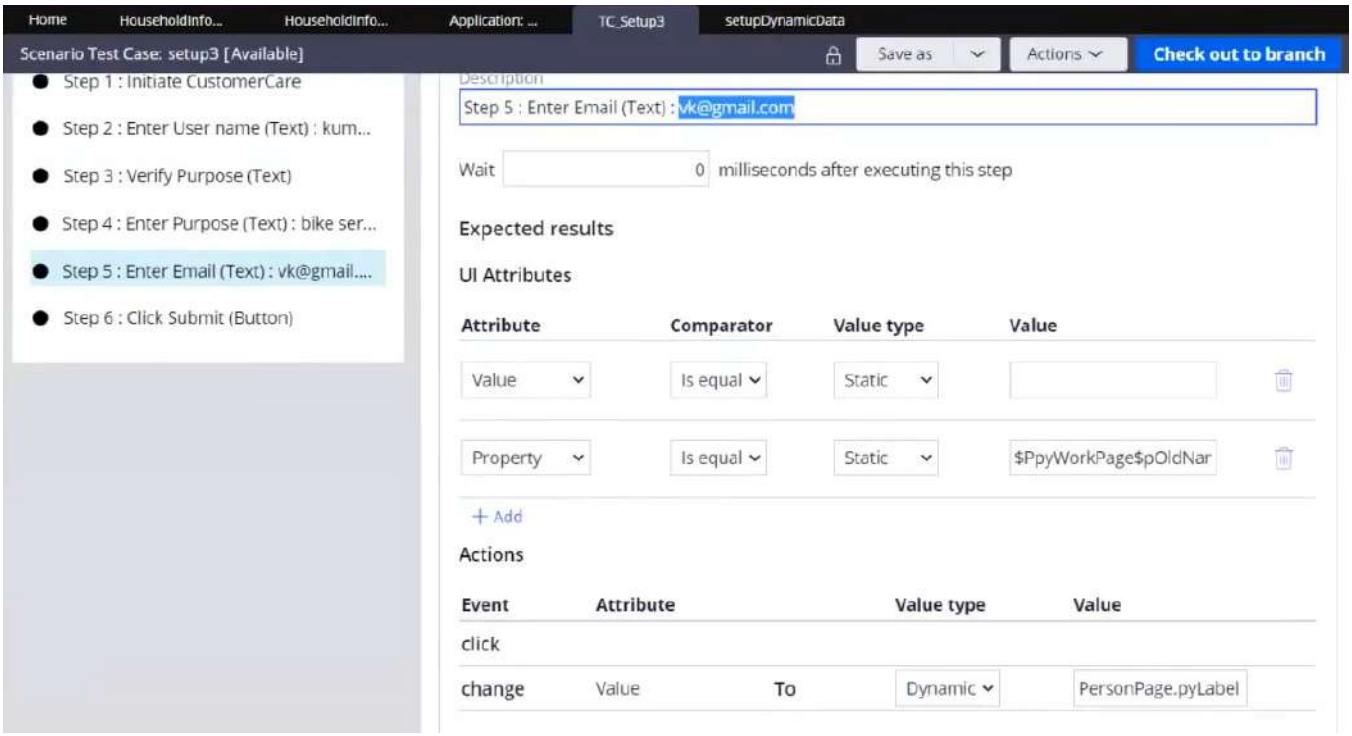
7. On the setupDynamicData page, create the dynamic value and associate it to a property for use in your scenario test case. For instance, on the **PersonPage** in the example below, the *.pyLabel* property consists of the *OperatorID.pyFirstName+OperatorID.pyLastName"@gmail.com"* dynamic data structure.





8. To utilize this dynamic field in your test case, navigate back to your scenario test case.
9. From the **Test case steps** section, select the step that contains the text field.
10. After selecting the test case step, navigate to the **Actions** field, and change the value type to **Dynamic**.
11. In the Value field, enter the page.property value created as part of your activity in Step 8. In this case, it is *PersonPage.pyLabel*.





12. Click **Save**.

Result: To automatically remove all referenced data pages, data objects, and user pages at the end of each test run, select the checkbox for **Clean up the test data at the end of run** .

- Setting up your test environment
- Cleaning up your test environment

Pega Test Automation Kit 1.1

The Pega Test Automation Kit 1.1 release includes the following new features and resolved issues.

Enhancements

The following new features are available in this release:

- You can now view and run scenario tests directly from Dev Studio on the rule form action menu, eliminating the need to launch the portal for every instance.
- The processes of adding and editing attachments are now recordable steps in the UI. See Attachment support in scenario testing for more information.
- The metrics tile of the Application Quality dashboard now includes scenario testing compliance metrics and execution rate, and provides a centralized location for all testing-specific application data. See Application Quality landing page for more information.
- All scenario testing functionality now supports the tracer tool for easier debugging.

Resolved issues

The following issues were resolved in this release:

- Unable to view failed scenario tests information directly from the portal.

  If a scenario test fails, you can now click on the **Failed** dialogue in the testing window to view and debug the test results. This dialogue provides the same test result summary information as in Dev Studio, without navigating out of the portal.

- Attachment support in scenario testing

  Enhance your end-to-end UI testing of run-time scenarios by simulating the user experience of interacting with attachments. When recording scenario tests, you can add and edit files attached to a test case and view the interaction during playback.

Attachment support in scenario testing

Enhance your end-to-end UI testing of run-time scenarios by simulating the user experience of interacting with

attachments. When recording scenario tests, you can add and edit files attached to a test case and view the interaction during playback.

To interact with attachments during a scenario test recording, perform the following steps:

Note: You can only add attachments in a scenario test if the attachment field is configured with the pzMultiFilePath control.

1. Launch App Studio or Dev Studio, based on where you want to perform the test.
2. Do one of the following to open the automation recorder:
   - In App Studio, on the lower-left side of the screen, click the **Test** icon.
   - In Dev Studio, on the lower-right side of the screen, toggle the run-time toolbar, and then click the **Toggle Automation Recorder** icon.
3. In the **Scenario tests** pane, click **Create test case**, and then select the test type:
   - To record a test for a portal, select **Portal**.
   - To record a test for a case, select **Case type**, and then select the type of case for which you want to record the test.**Note:** When you select the case type, a new case of that type is created.
4. Record the steps for the test by clicking the UI elements. **Result:** When you hover over a testable element, an orange highlight box appears. When you click an element, you record an implicit assertion and add the interaction to the list of test steps.
5. Click **Attachments** to attach a file to the work object. You can also attach files from **Recent attachments**, or anywhere that files can be attached from the UI.
   a. To attach a local file, select **File from device**.
   b. To attach a link, select **URL**.
6. Provide the required information and select **Submit** to attach the file to the work object. Attached files appear on the **Files & documents** pane on the right side.
7. Continue recording as many steps as necessary. When you finish adding steps, in the **Test case** pane, click **Stop and save test case**.
8. You can edit attached files by changing the **Attachment key** in the test case step. To edit the attachment:
   a. Navigate to the test case that you want to edit and select **Edit**. See Opening a scenario test case for more information.
   b. Identify the step that includes the attachment and select the pencil icon to edit that attachment.
   c. Edit the **Attachment key** to configure the test case attachment as required.
   d. Click **Save step**.
   e. Click **Save** to exit the editor and update the test case.

   Result: The test case appears on the **Scenario testing landing page**.

Analyzing application quality metrics

Quickly identify areas within your application that need improvement by viewing metrics related to your application's health on the Application Quality dashboard.

- Viewing application quality metrics

  Quickly identify areas within your application that need improvement by viewing metrics related to your application's health on the Application Quality dashboard.

- Changing application quality metrics settings

  The Application Quality settings provides configurable options related to quality metrics. You can change the default settings for metrics displayed to meet your business needs.

- Estimating test coverage

  View historical test coverage metrics and generate reports containing the number of executable rules and their test coverage. Use the data to analyze changes in test coverage, and to verify which rules require testing.

- Creating unit test suites with AUT

  Unit Test Suites identify a collection of Test Cases and their rulesets, and a user (Operator ID) whose credentials are used to run the Unit Test Suite. Unit Test Suites are used to automatically run groups of test cases together and make unit testing more efficient.

Viewing application quality metrics

Quickly identify areas within your application that need improvement by viewing metrics related to your application's health on the Application Quality dashboard.

For example, view your application's compliance score and see the number and severity of guardrail violations that were found in your application. You can then improve your application's compliance score and overall quality by investigating and resolving the violations.

To open the Application Quality dashboard, from the Dev Studio header, click **Configure Application Quality Dashboard** .

You can view the following metrics:

- **Rule, case, and application** – View the number of executable rules (functional rules that are supported by test coverage) and the number of case types in the selected applications. To view metrics for a different combination of applications, select a different list on the **Application: Quality Settings** page.
- **Guardrail compliance** – View the compliance score and the number of guardrail violations for the included applications, as well as a graph of changes to the compliance score over time. To see more details about the application's guardrail compliance, click **View details**.
- **Test coverage** – View the percentage and number of rules that are covered by tests, and the last generation date of the application-level coverage report for the selected applications, as well as a graph of changes to application-level coverage over time. To see test coverage reports or to generate a new coverage report, click **View details**. **Note:** If the *EnableBuiltOnAppSelectionForQuality* switch is turned on, then coverage sessions metrics are also displayed on the Application Quality Dashboard for the built-on applications selected in Application: Quality Settings.
- **Unit testing** – View the percentage and number of Pega unit test cases that passed for the selected applications, over the period selected on the **Application Quality Settings** landing page. The graph illustrates the changes to the test pass rate over time. To see reports about test compliance and test execution, click **View details**.
- **Case types** – View guardrail score, severe guardrail warnings, test coverage, unit test pass rate, and scenario test pass rate for each case type in the applications. To view additional details about a case type, click **View details**.
- **Data types** – View guardrail score, severe guardrail warnings, test coverage, and unit test pass rate for each data type in the applications. To view additional details about a data type, click **View details**.
- **Other rules** – View guardrail score, test coverage, test pass rate, the number of warnings, a list of rules with warnings, the number and list of uncovered rules, and the number and list of failed test cases for rules that are used in the selected applications but that are not a part of any case type.

- Application quality metrics

  The Application Quality dashboard displays metrics for guardrails, test coverage, and unit testing that you can use to assess the overall health of your application and identify areas that require improvement. You can change the default ranges for the color codes by modifying the corresponding when rules in the Data-Application-Quality class.

- Changing application quality metrics settings
- Application quality metrics
- Estimating test coverage

Application quality metrics

The Application Quality dashboard displays metrics for guardrails, test coverage, and unit testing that you can use to assess the overall health of your application and identify areas that require improvement. You can change the default ranges for the color codes by modifying the corresponding when rules in the *Data-Application-Quality* class.

The following table describes the relationship between colors, default ranges, and when rules. For each metric in the Red, Orange, and Green columns, the top row indicates the default range for each color and the bottom row indicates the corresponding when rule.

Metric		Red – stop development and fix issues	Orange – continue development and fix issues	Green – continue development
Guardrails	Weighted score	0–59	60–89	90–100
		pyIsWeightedScoreSevere	pyIsWeightedScoreModerate	pyIsWeightedScorePermissible
	Number of warnings	Not applicable	More than 0	0
			pyIsWarningsModerate	pyIsWarningsPermissible
	Number of severe warnings	More than 0	Not applicable	0
		pyAppContainSevereWarning		pyAppContainNoSevereWarning
Test coverage	Rules covered	0%–59%	60%–89%	90%–100%
		pyIsRuleCoverageScoreSevere	pyIsRuleCoverageScoreModerate	pyIsRuleCoverageScorePermissible
Unit testing	Test pass rate	0%–59%	60%–89%	90%–100%
		pyIsTestPassRateScoreSevere	pyIsTestPassRateScoreModerate	pyIsTestPassRateScorePermissible

Metric		Red – stop development and fix issues	Orange – continue development and fix issues	Green – continue development
Guardrails	Weighted score	0–59	60–89	90–100
		pyIsWeightedScoreSevere	pyIsWeightedScoreModerate	pyIsWeightedScorePermissible
	Number of warnings	Not applicable	More than 0	0
			pyIsWarningsModerate	pyIsWarningsPermissible
	Number of severe warnings	More than 0	Not applicable	0
		pyAppContainSevereWarning		pyAppContainNoSevereWarning
Test coverage	Rules covered	0%–59%	60%–89%	90%–100%
		pyIsRuleCoverageScoreSevere	pyIsRuleCoverageScoreModerate	pyIsRuleCoverageScorePermissible
Unit testing	Test pass rate	0%–59%	60%–89%	90%–100%
		pyIsTestPassRateScoreSevere	pyIsTestPassRateScoreModerate	pyIsTestPassRateScorePermissible

- Viewing application quality metrics

Changing application quality metrics settings

The Application Quality settings provides configurable options related to quality metrics. You can change the default settings for metrics displayed to meet your business needs.

Note: To change settings to enable the *EnableBuiltOnAppSelectionForQuality* toggle that allow you to select which built-on applications are included, your operator ID must have the *SysAdm4* privilege.

On the Application Quality settings landing page, you can modify the following settings for application quality:

- **Application(s) included** – If you want the test coverage report to include only rules from the current application, select **Current application only**. If you want the test coverage report to also include rules from built-on applications, select **Include built-on applications**. By default, only the current application is selected. If you enable the *EnableBuiltOnAppSelectionForQuality* toggle, you can select which built-on application will be included. **Note:** If a master user starts an application-level coverage session for an application, then that user's configuration of this setting is in effect for all users that execute test coverage for the duration of the session.
- **Ignore test rulesets when calculating Guardrail score** – When you enable this setting, the guardrail score is calculated without taking test rulesets into account. This is the default behavior. When you disable this setting, test rulesets are taken into account when calculating the guardrail score.
- **Quality trends** – Use this setting to change the date range of the trend graphs on the Application Quality, Application: Test coverage and Application: Unit testing landing pages. The default value is **Last 2 weeks**.
- **Test case execution** – Use this setting to change the number of days, from the time that they are executed that tests are treated as executed by the Application Quality dashboard and coverage reports. By default, a test executed later than the last seven days is excluded from the Application Quality dashboard and in reports.
- **Scenario test case execution** – Use this setting to add a delay (in milliseconds) to the execution of all the steps in all the scenario tests in an application. For more information, see Delaying scenario test execution.
- **Re-run tests automatically on failure** – Use this setting to automatically rerun scenario tests if they fail, up to a maximum of two times. For more information, see Automatically rerunning failed scenario tests.

- Viewing application quality metrics
- Estimating test coverage
- Application quality metrics

Estimating test coverage

View historical test coverage metrics and generate reports containing the number of executable rules and their test coverage. Use the data to analyze changes in test coverage, and to verify which rules require testing.

On the Test Coverage landing page, view a chart displaying test coverage metrics and generate specific user-level, application-level, and merged coverage reports. User-level reports contain the results of a single test coverage session that a user performs, while application-level reports contain results from multiple test coverage sessions that many users run. Merged reports contain results from multiple most recent application-level reports.

The following rule types are included in test coverage reports.

• Activity • Case type • Collection • Correspondence • Data page • Data transform • Decision data • Decision table • Decision tree	• Declare expression • Declare trigger • Flow • Flow action • Harness • HTML • HTML fragment • Map value • Navigation	• Paragraph • Report definition • Scorecard • Section • Strategy • Validate • When • XML Stream

Note: Ensure you have the appropriate access to generate reports by associating the *pzStartOrStopMasterAppRuleCoverage* privilege with your account. For more information, see Granting privileges to an access role.

- Generating a user-level test coverage report

  Generate a user-level test coverage report to identify which executable rules in your currently included applications are covered and not covered by tests. The results of this type of report are not visible on the Application Quality Dashboard.

- Generating an application-level test coverage report

  Generate an application-level coverage report that contains coverage results from multiple users. Use this report to identify which executable rules in your currently included applications are covered and not covered by tests. The results of this type of report are visible on the Application Quality Dashboard.

- Participating in an application-level test coverage session

  When an application-level coverage session is running, you can perform tests of the application to contribute to an application-level test coverage report that identifies the executable rules in your application that are covered and not covered by tests.

- Generating a merged coverage report

  Generate application-level coverage reports for every application in your system and in your application stack, and then merge the most recent reports to a single report, to gain a consolidated overview of test coverage for all your top-level or built-on applications.

Generating a user-level test coverage report

Generate a user-level test coverage report to identify which executable rules in your currently included applications are covered and not covered by tests. The results of this type of report are not visible on the Application Quality Dashboard.

1. In the header of Dev Studio, click **Configure Application Quality Test Coverage** .
2. Click **User level**.
   Note: If the Application level coverage is in progress message is displayed, you cannot start a user-level coverage session.
3. Click **Start new session**.
4. Enter the title of the coverage report, and then click **OK**.
5. To provide data for the report, run all of the tests that are available for your included applications, for example, Pega unit automated tests and manual tests.
6. Click **Stop coverage**, and then click **Yes**.
   Note: If you close the tab or log out without clicking **Stop**, the report is not generated.
7. Review the results of the coverage session. In the **Coverage history** section, click **Show Report**.
8. **Optional:** To see whether coverage reports were generated by other users, click **Refresh**.
9. **Optional:** To see a list of application-level coverage reports, click **Application level**.

- Participating in an application-level test coverage session

Generating an application-level test coverage report

Generate an application-level coverage report that contains coverage results from multiple users. Use this report to identify which executable rules in your currently included applications are covered and not covered by tests. The results of this type of report are visible on the Application Quality Dashboard.

1. In the header of Dev Studio, click **Configure Application Quality Test Coverage** .
2. Click **Application level**.
3. Click **Start new session**.
   Note: To start application-level coverage, your operator ID must have the *pzStartOrStopMasterAppRuleCoverage* privilege.
4. Enter the title of the coverage report, and then click **OK**.
5. **Optional:** To provide data for the report, run all of the tests that are available for your currently included applications, for example, Pega unit automated tests and manual tests.

6. Inform all relevant users that they can log in to the application and start running tests.
7. Wait until all users have completed their tests and have logged off.
   Note: If you stop an application coverage session before a user has logged off, the coverage data of this user is not included in the report.
8. Click **Stop coverage**, and then click **Yes**.
9. Review the results of coverage session. In the **Coverage history** section click **Show Report**.
10. **Optional:** To see whether coverage reports were generated by other users, click **Refresh**.
11. **Optional:** To see a list of user-level coverage reports, click **User level**.

Participating in an application-level test coverage session

When an application-level coverage session is running, you can perform tests of the application to contribute to an application-level test coverage report that identifies the executable rules in your application that are covered and not covered by tests.

Before you begin: Ensure that application-level coverage is in progress before you log in. If application coverage is started after you log in, you cannot contribute to it unless you log off and log in again. Only users with the *pzStartOrStopMasterAppRuleCoverage* privilege can initiate application-level coverage.

1. Check if application-level coverage is in progress.
   a. In the header of Dev Studio, click **Configure Application Quality Test Coverage** .
   b. Verify that you see the Application level coverage is in progress message.
      If you do not see the message, application-level coverage is not active, however, you can still start a user-level test coverage session.
2. To provide data for the report, execute all the tests that are available for the included applications, for example, Pega unit automated tests and manual tests.
   Note: During the coverage session your local configuration for included applications is overridden by the configuration of the user that started the application-level coverage session.
3. Click your profile icon and then click **Log off**.
   Note: If you do not log off before the rule coverage session is stopped, you will not contribute to the report. If you log off and then log in again while the coverage session is still active, your test coverage sessions are saved as a new session that will be included in the application coverage report.

- Generating an application-level test coverage report

Generating a merged coverage report

Generate application-level coverage reports for every application in your system and in your application stack, and then merge the most recent reports to a single report, to gain a consolidated overview of test coverage for all your top-level or built-on applications.

Insight from a merged application report helps you avoid creating duplicate tests for rules that are used across multiple applications. **Note:** Because a merged report is an instance of an application-level report, when a merged report is the most recent one for an application, it is included in the next merged report.
Before you begin: Ensure that your operator ID has the *pzStartOrStopMasterAppRuleCoverage* privilege. Generate at least one application-level coverage report for another application in your system or for a built-on application in your current application. For more information, see Generating an application-level test coverage report.

1. Switch to the main application that you want to use as the baseline for the merged report. See Switching between applications.
2. In the header of Dev Studio, click **Configure Application Quality Test Coverage** .
3. Click the **Application level** tab.
4. In the **Coverage history** section, click **Merge reports**.
5. Enter the title of the merged report, and then click **Next**.
6. In the list of the most recent reports, select the reports that you want to include in the merged report, and then click **Create**.
7. Close the **Merge confirmation** window. **Result:** Dev Studio automatically adds the MRG_ prefix to every merged report to differentiate them from standard application-level coverage reports and to facilitate finding them.
8. Open the merged report. In the **Coverage history** section find the merged report that you created and click **Show report**.
9. **Optional:** To open a report that is included in the merged report, in the **Merged reports** section, click the report name.

Viewing test coverage reports

View a report that contains the results of test coverage sessions to determine which rules in your application are not covered with tests. You can improve the quality of your application by creating tests for all uncovered rules that are indicated in the reports.

1. In the header of Dev Studio, click **Configure Application Quality Test Coverage** .
2. Choose the type of report that you want to view:
   - To view application-level test coverages, click the **Application level** tab.

    - To view user-level test coverages, click the **User level** tab.
3. In the **Coverage history** section, hover over the row with the relevant test coverage session, and then click **Show Report**.
4. **Optional:** Choose the data you want to include in the report:
    - To include only the rules that were updated after a specific date, in the **Rules updated after** field, click the calendar icon, select a date and time, and then click **Apply**.
    - To include all the rules that are covered with tests, click **Covered**.
    - To include all the rules that are not covered with tests, click **Uncovered**.
    - To filter the rules, in the column header that you want to filter, click the filter icon, enter the filter criteria, and then click **Apply**.
    - To open a single report that is included in a merged report, in the **Merged reports** section, click the report name.
    - To open a rule that is included in the report, click the rule name.

Setting up for test automation

Before you create Pega unit test cases and test suites, you must configure a test ruleset in which to store the tests.

- Creating a test ruleset to store test cases

  Before you can create unit test cases or scenario tests, you must configure a test ruleset in which to store the tests.

Creating a test ruleset to store test cases

Before you can create unit test cases or scenario tests, you must configure a test ruleset in which to store the tests.

1. In the Dev Studio header, open your application rule form by clicking your application and selecting **Definition**.
2. In the **Application rulesets** section, click **Add ruleset**.
3. Complete one of the following actions:
    - Enter a ruleset name and version of an existing ruleset.
    - Click the **Open** icon and create a new ruleset.

   Note: The test ruleset must always be the last ruleset in your application stack.
4. Open the test ruleset and click the **Category** tab.
5. Select the **Use this ruleset to store test cases** check box.
6. Save the test and application ruleset forms.

Result: When you save test cases for rules, they are saved in this ruleset.

- PegaUnit testing