



# Tokyo DevOps

Last updated: 07/05/2023

Some examples and graphics depicted herein are provided for illustration only.  
No real association or connection to ServiceNow products or services is intended  
or should be inferred.

ServiceNow, the ServiceNow logo, Now, and other ServiceNow marks  
are trademarks and/or registered trademarks of ServiceNow, Inc., in  
the United States and/or other countries. Other company and  
product names may be trademarks of the respective companies  
with which they are associated.

Please read the ServiceNow Website Terms of Use at  
[www.servicenow.com/terms-of-use.html](http://www.servicenow.com/terms-of-use.html)

Company Headquarters  
2225 Lawson Lane  
Santa Clara, CA 95054  
United States  
(408) 501-8550

# Table of Contents

DevOps.....	4
DevOps Change Velocity.....	6
Exploring DevOps Change Velocity.....	8
Getting started with DevOps Change Velocity.....	14
Integrating DevOps Change Velocity with third-party tools.....	18
Create an application in DevOps Change Velocity.....	173
Accelerating your DevOps change process.....	177
DevOps Insights reports and Pipeline UI.....	197
DevOps Change Velocity reference.....	229
DevOps Config.....	283
Exploring DevOps Config.....	285
Configuring DevOps Config.....	290
Integrating your Azure or Jenkins pipeline in DevOps Config.....	334
Using DevOps Config.....	361
Analytics and reporting using the DevOps Config Insights dashboard.....	419
DevOps Config reference.....	421
Index.....	a

# DevOps

---

Use the ServiceNow® DevOps application with your DevOps toolchain to provide data insights, accelerate change, development performance, validate configuration data, and increase the visibility of your DevOps environment using a single system.

## Achieve enterprise DevOps

Companies are turning to DevOps to speed up product delivery and innovation. DevOps is integral to delivering higher-quality software, improving customer responsiveness, and building competitive advantage.

ServiceNow DevOps aims at shorter development cycles, increased deployment frequency, and more dependable releases, in close alignment with business objectives.

## Accelerate change while minimizing risk

DevOps Change Velocity connects to the DevOps tools that your development teams are using – especially tools associated with the DevOps pipeline.

The DevOps Change Velocity application provides default integrations with CI/CD tools like Microsoft Azure DevOps, Jenkins, GitHub Actions, and GitLab along with tools that connect to the pipeline like Jira (for planning) or testing and security scanning tools. These integrations enable organizations to bridge the gap between development and operations allowing DevOps teams to roll out releases rapidly and efficiently.

DevOps Change Velocity enables you to automatically gather and connect the right information from your DevOps toolchain to maintain an end-to-end audit trail. For more information, see [DevOps Change Velocity](#)

## Gain insights across the software delivery value stream

The DevOps Insights dashboard provides visibility across the CI/CD pipeline, allowing Development Leadership to take data-driven decisions, balancing priorities of Development and Operations. Increase change throughput while minimizing risk and overhead using AI-based change policies. For more information, see [DevOps Insights](#)

## Manage, secure, and validate data and configuration changes

DevOps Config allows developers and operations teams to track configuration changes, identify and prevent potential configuration-related issues before new code is deployed. DevOps Config helps reduce the cost of configuration data management and increase the reliability and quality of enterprise applications along the continuous delivery pipeline. For more information, see [DevOps Config](#)

In addition to Now Platform® core features, DevOps integrates with the following ServiceNow applications:

- Now Intelligence [Performance Analytics](#)
- IT Service Management [Change Management](#)
- Strategic Portfolio Management (SPM) [Agile Development](#)
- Governance, Risk, and Compliance [Risk Management](#)

[View and download the full infocard](#) for a highlight of the DevOps features.

	<p>Accelerate deployments by automating change</p> <p>Automatically create change requests at any stage for deployments that require change control in your environment. Use change approval policies to automate change request approval to continue deployment through the execution pipeline automatically.</p>
	<p>Visualize results and change outcomes across a pipeline execution</p> <p>Use the DevOps Pipeline UI view to show pipeline stage progression and details for each app.</p>
	<p>Manage, secure, and validate configuration data</p> <p>The DevOps Config application stores configuration data keys and their values for infrastructure, environments, releases and applications, in a configurable data model. This data model results in a structured configuration data providing reusability, hierarchies with inheritance, identification of duplicates and alerts for conflicting settings.</p>
	<p>Evaluate and monitor DevOps process and results</p> <p>Use the data-driven DevOps Insights dashboard to view change results reports, pipeline value stream, and evaluate the overall DevOps process.</p>
	<p>Set up user-created integrations for additional planning, coding, and test tools</p> <p>Integrate planning, coding, and test tools not included in the integrations provided with the DevOps application.</p>

## Accelerate deployments by automating change

Change requests are automatically created for stages under change control. You can also enable automatic approval decisions of DevOps changes using the Flow Designer and Change Approval Policies such as DevOps Change Policy.

## Visualize results and change outcomes across a pipeline execution

Quickly view how everything is connected to see exactly what's happening with the pipeline and when. View the artifacts, commits, work items, test results, and other details for a change request in one place.

## Manage, secure, and validate configuration data

DevOps Config provides a single source of truth for all your configuration data used at any given moment for any given application, across multiple app versions and in any given environment. Define policies to pinpoint configuration data that contain unencrypted sensitive information. DevOps Config continuously monitors incoming data changes and apply all validation rules to prevent other tools from consuming any broken configuration data along the continuous delivery tool chain.

## Evaluate and monitor DevOps process and results

Use the DevOps Insights dashboards to help you plan and track your DevOps processes and evaluate the performance of your software delivery value chain.

## Set up user-created integrations for additional planning, coding, and test tools

The DevOps application includes tool definitions for integrating some common planning, coding, and test tools. You can also set up user-created integrations for additional tools in your DevOps environment.

### Learn

- [What is DevOps? ↗](#)
- [What is CI/CD? ↗](#)
- [What is CI/CD pipeline? ↗](#)
- [What is continuous integration? ↗](#)
- [What is infrastructure-as-a-service \(IaaS\)? ↗](#)
- [What is infrastructure-as-code \(IaC\)? ↗](#)
- [What is the software development life cycle \(SDLC\)? ↗](#)

### Get started

- Work with an implementation specialist to streamline your DevOps setup process. To learn more, visit the [Customer Success Center ↗](#).
- See the [DevOps overview](#) for information on how to request DevOps and begin setup.
- Take a DevOps course to learn about DevOps fundamentals and how to implement DevOps. To sign up, visit [ServiceNow training and certification ↗](#).

### Applications and features

- [DevOps Change Velocity](#)
- [DevOps Config](#)

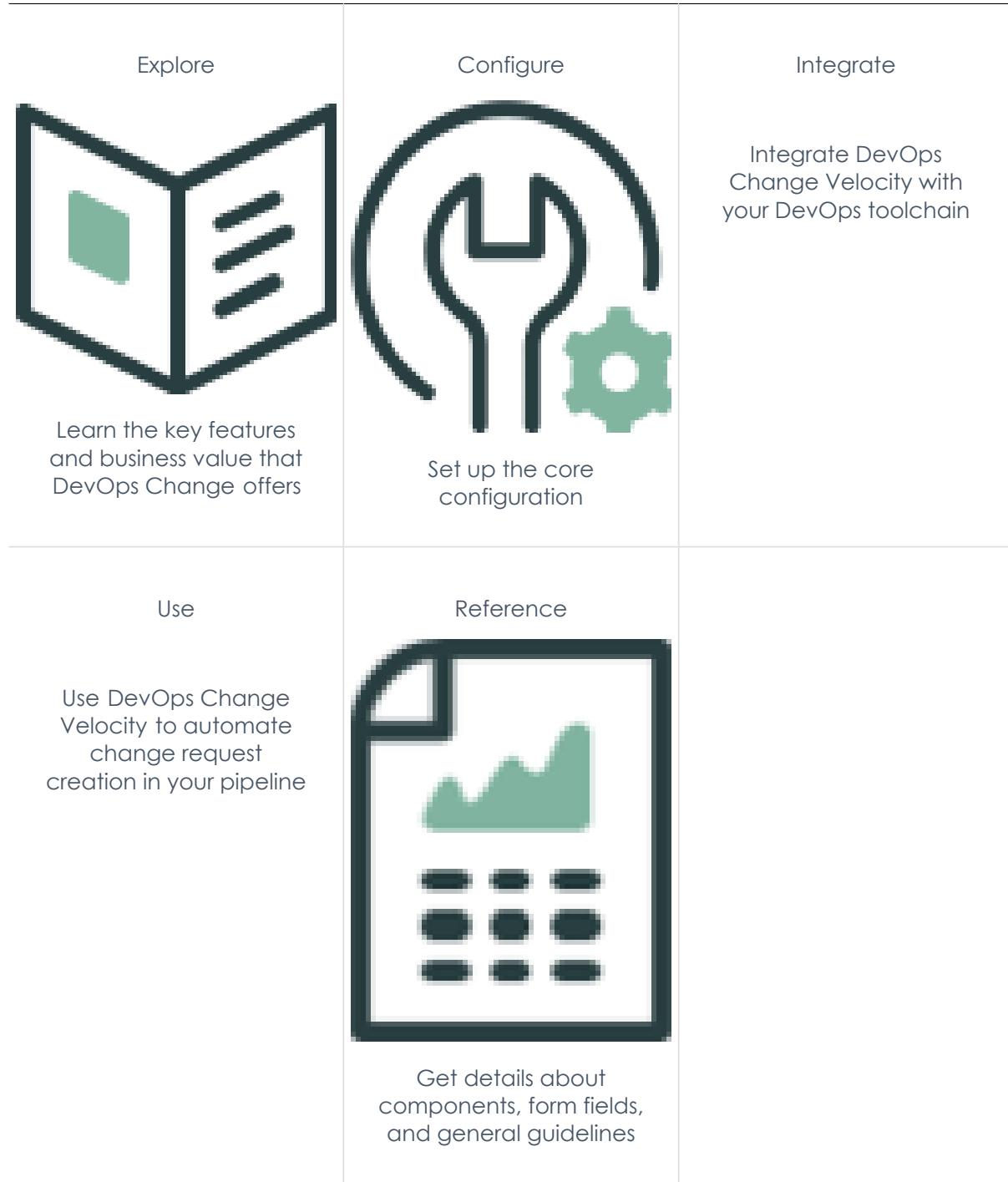
## DevOps Change Velocity

With ServiceNow® DevOps Change Velocity, you can connect data from your DevOps tool chain with ServiceNow platform to deliver changes faster without sacrificing compliance. You can automatically create change requests from the CI/CD pipelines and even automatically approve the change requests based on policies. This allows developers to continue to remain in their tools but still provide visibility, accelerate time required to approve change requests and provide an audit trail.

### DevOps Change Velocity overview

[ServiceNow DevOps overview](#)

## Get started



## Troubleshoot and get help

- Ask or answer questions in the Now Community [↗](#)
- Search the Known Error Portal for known error articles [↗](#)
- Contact Customer Service and Support [↗](#)

## Exploring DevOps Change Velocity

Learn how ServiceNow DevOps Change Velocity helps your Dev and Change Management teams to accelerate change and connect your DevOps tool chain with ServiceNow so that DevOps tool data can be used to automate the change process.

### Overview

DevOps Change Velocity collects data from your DevOps planning, coding, testing and orchestration toolchain, to provide visibility across the entire software delivery life cycle. This enables a single view for both the Development team and the Operations team across the end-to-end activities (plan, develop, build, test, deploy, and operate).

The DevOps Change Velocity application provides the following major benefits:

- **Change acceleration:** Provides the ability to automate change workflow and approval decisions in order to increase change velocity whilst ensuring governance and control.
- Automaticaly create and complete DevOps change requests and automate approval decisions based on input data (auto-approve, auto-reject or manually approve).
- **Change traceability:** Track the progress of your work items, commits, artifacts, builds, and releases using the change requests to lend additional transparency and provide a single source of truth to all the personas involved.
- **DevOps Insights:** Provides flow metrics, change acceleration metrics and other KPIs across the end-to-end software delivery value chain and connected tools.

To learn about the tools that the DevOps Change Velocity application supports, see [DevOps Change Velocity integrations](#).

To understand how you can incrementally progress to get value faster using DevOps Change Velocity, see [User journey for DevOps Change Velocity](#).

### Benefits of DevOps Change Velocity per persona

Persona	Challenges	DevOps Change Velocity benefits
Developer	<ul style="list-style-type: none"> <li>• Long feedback loop for change approval</li> <li>• Manual collection of data</li> <li>• Swivel chair from changes to work items</li> <li>• Lack of end-to-end visibility of entire DevOps lifecycle</li> <li>• Significant time spent on admin tasks</li> </ul>	<ul style="list-style-type: none"> <li>• Automated data collection process through integrations</li> <li>• No time lost on manually creating a change request or providing evidence manually</li> <li>• More time for actual development work</li> <li>• Developers can keep working in their own tools</li> </ul>

Persona	Challenges	DevOps Change Velocity benefits
Change manager	<ul style="list-style-type: none"> <li>• Difficulty in handling increasing number of changes</li> <li>• Not able to focus on only the changes that truly need human intervention</li> <li>• Manual work to respond to audit request</li> <li>• Significant time spent on gathering data</li> </ul>	<ul style="list-style-type: none"> <li>• Accelerate change process with automated flows</li> <li>• Visibility over pipeline, tests, and change in a single tool</li> <li>• Automated documentation of change content and controls</li> <li>• Automate appropriate change governance driven by data</li> <li>• One-click audit response</li> </ul>
Release manager	<ul style="list-style-type: none"> <li>• Lack of correlation between changes and release to measure progress</li> <li>• Unable to eliminate manual tasks from delivery processes</li> </ul>	<ul style="list-style-type: none"> <li>• Enhanced visibility into the release plan and progress</li> <li>• Automation of more tasks with integrations</li> <li>• Improved release reliability</li> </ul>
Application development leadership	<ul style="list-style-type: none"> <li>• Lack of visibility across teams and value streams</li> <li>• Lack of team performance comparison data</li> <li>• Inability to identify and resolve performance bottlenecks</li> </ul>	<ul style="list-style-type: none"> <li>• Visibility into the entire value stream</li> <li>• Normalized views of team performance</li> <li>• Identify bottlenecks and take appropriate action</li> </ul>
Operations and Support teams	<ul style="list-style-type: none"> <li>• Limited forecast and speed of remediation</li> <li>• Lack of visibility over change content and impact</li> <li>• Too many tools to correlate change, release, and incidents</li> </ul>	<ul style="list-style-type: none"> <li>• Single tool to manage changes and incidents</li> <li>• Easier correlation between change and incident</li> <li>• Reduced MTTR from improved visibility</li> </ul>
IT compliance officer	<ul style="list-style-type: none"> <li>• Limited visibility to ensure compliance</li> <li>• Lack of transparency of data and process to measure risks</li> </ul>	<ul style="list-style-type: none"> <li>• Ensure compliance with automated change policy</li> <li>• Improved control and visibility with automated gathering of data</li> </ul>

Explore more details such as the user adoption journey, supported integrations, and Insights in the following sections.

## User journey for DevOps Change Velocity

Review the adoption journey of the DevOps Change Velocity application to allow for a phased (crawl, walk, run) approach towards complete automation of change approvals.

While automating change request creation from the CI/CD Pipelines and automatically approving them will still be end goals with DevOps Change Velocity, we recommend approaching your implementation in this phased manner. Each phase below provides incremental value over the previous phase and allows you to get started with minimum changes to your existing processes:

1. Connect Tools and Applications: Get started by integrating your key DevOps tools and choosing the key objects from these tools to track. This can be done without needing to modify your pipelines or any of the development assets. And then create DevOps Apps for the teams you want to onboard first. Completing this phase is necessary to bring data from your tools over to the Now Platform.

For more information on integrating tools, see [Integrating DevOps Change Velocity with third-party tools](#).

2. Change traceability: Once you have established integration with your key DevOps tools, relevant data from these tools will start coming into ServiceNow. Then you can continue to use your existing process to create Changes but save time by having all relevant information added to the Change request with just a few clicks. This information includes stories, code commits, test results, quality scans, and others. For more information on modeling a change request flow, see [Using DevOps Model Change Request flow](#)

After these two phases are complete, you can progress toward the next two phases which help you achieve automation of your Change process, accelerate deployments, and increase velocity. We recommend that you get started with these two phases as they don't need any change to your existing processes or pipelines.

3. Change registration: This phase builds value further now by automating the change request creation from the CI/CD pipelines. With this phase you will need to modify the team's CI/CD pipeline to enable automatic creation of Change requests. This automation saves time of your developers as they don't have to manually fill the Change requests, thereby reducing the risk of human error. For more information, see [Configuring DevOps change request details within the pipeline](#).
4. Change automation: This is the final phase in value realization where data from the tools will come into ServiceNow in real time, Change requests will be automatically created and they will be automatically approved or rejected based on data driven policies. This phase requires creating policy guidelines and enabling automated change approvals decisions based on input data to reduce risk and increase rate of Change success. For more information, see [Accelerating your DevOps change process](#).

## DevOps Change Velocity integrations

Get an overview of how DevOps Change Velocity integrates with your external DevOps toolchain and the tools supported for this integration.

The DevOps Change Velocity integration with external tools is achieved by exposing REST endpoints to receive webhook notifications or direct REST calls from these tools in real-time. Additionally, Devops Change also allows importing of data from these tools using polling.

DevOps API allows integration with any coding, planning, or orchestration tools. For more information, see [DevOps API](#).

## Supported tools

The DevOps Change Velocity application provides connectors by default to connect the following commonly used DevOps tools to import data. If your tool is not included in this list, you can manually connect and configure an integration. See [User-created integrations in DevOps Change Velocity](#).

Tool type	Tools
Planning	<ul style="list-style-type: none"> <li>• Azure Boards (Azure DevOps latest cloud version as of Jan 2023)</li> <li>• Jira (ver 8.5.1)</li> <li>• ServiceNow Agile Development 2.0 (ver Sandiego, Tokyo, Utah)</li> <li>• Rally (latest cloud version as of Jan 2023)</li> </ul>
Coding	<ul style="list-style-type: none"> <li>• Azure Repos (Azure DevOps latest cloud version as of Jan 2023)</li> <li>• Bitbucket (ver 7.19.2)</li> <li>• GitHub (latest cloud version as of Jan 2023)</li> <li>• GitHub Enterprise (ver 3.5.7) <ul style="list-style-type: none"> <li>◦ Basic authentication</li> <li>◦ OAuth</li> </ul> </li> <li>• GitLab (ver 13.0.6 for on-premises or 2023 GitLab B.V for cloud)</li> </ul>
Orchestration	<ul style="list-style-type: none"> <li>• Azure Pipelines (Azure DevOps latest cloud version as of Jan 2023) <ul style="list-style-type: none"> <li>Jobs supported:</li> <li>◦ Agent job</li> <li>◦ Agentless (server) job</li> </ul> </li> <li>• Jenkins (ver 2.289.1) <ul style="list-style-type: none"> <li>Jobs supported:</li> <li>◦ Freestyle project</li> <li>◦ Folder (default is 3 levels)</li> <li>◦ Pipeline</li> <li>◦ Multibranch Pipeline</li> </ul> </li> <li>• GitLab (ver 13.0.6 for on-premises or 2023 GitLab B.V for cloud)</li> </ul>

Tool type	Tools
	<ul style="list-style-type: none"> <li>• GitHub (latest cloud version as of Jan 2023)</li> <li>• GitHub Enterprise (ver 3.5.7) <ul style="list-style-type: none"> <li>◦ Basic authentication</li> <li>◦ OAuth</li> </ul> </li> </ul>
Repository artifacts	<ul style="list-style-type: none"> <li>• JFrog (ver 7 for on premises or latest cloud version as of Jan 2023)</li> <li>• Azure Artifacts (Azure DevOps latest cloud version as of Jan 2023)</li> </ul>
Testing	<p>If any test is run as part of the pipeline executions of the following supported Orchestration pipelines, the information is shown in the test summary</p> <ul style="list-style-type: none"> <li>• GitHub (latest cloud version as of Jan 2023)</li> <li>• GitHub Enterprise (ver 3.5.7) <ul style="list-style-type: none"> <li>◦ Basic authentication</li> <li>◦ OAuth</li> </ul> </li> <li>• GitLab (ver 13.0.6 for on-premises or 2023 GitLab B.V for cloud)</li> <li>• Azure DevOps (latest cloud version as of Jan 2023)</li> <li>• Jenkins (ver 2.289.1)</li> </ul>
Software Quality	<p>SonarQube (ver 8.9.6 or latest cloud version as of Jan 2023) scans supported on</p> <ul style="list-style-type: none"> <li>• Azure DevOps pipelines (latest cloud version as of Jan 2023)</li> <li>• Jenkins (ver 2.289.1) pipelines</li> </ul>
Feature Flag	Split (latest cloud version as of Jan 2023)

## Third-party extensions

Use the ServiceNow extensions to model your pipeline in DevOps and configure branch analysis for tools like SonarQube.

Additional extensions may be required for applications such as Jenkins or Azure DevOps. These extensions are used when ServiceNow DevOps cannot integrate using only the native REST API and push notifications.

Jenkins plugin for ServiceNow DevOps

A Jenkins plugin is provided to enable change acceleration so your orchestration tool can communicate and control certain aspects of pipeline executions from within ServiceNow DevOps.

Visit the Ancillary Software section on the [ServiceNow Store](#) website to download the Jenkins plugin for ServiceNow DevOps.

#### ServiceNow DevOps for Azure DevOps

Use the **ServiceNow DevOps** extension on [Visual Studio Marketplace](#) if you plan to integrate your Azure DevOps pipeline with ServiceNow DevOps.

For more information, see [Use the ServiceNow DevOps extension for Azure DevOps](#).

#### ServiceNow DevOps custom actions for GitHub Actions

Use the **ServiceNow DevOps** custom actions on [GitHub marketplace](#) if you plan to integrate your GitHub workflows with ServiceNow DevOps.

For more information, see [Additional information - GitHub Actions](#).

To start integrating DevOps Change Velocity with your toolchain, see [Integrating DevOps Change Velocity with third-party tools](#).

#### Related topics

[Getting started with DevOps Change Velocity](#)

## DevOps dashboards

Use the DevOps Insights dashboard to evaluate the results of your overall DevOps process. Use the DevOps Pipeline UI view to visually analyze your pipeline executions.

Once the DevOps toolchain integrations and change policies are configured, all data is connected to provide insights about the process and velocity from the team level to the enterprise level.

Using the DevOps Insights reports, monitor pipeline modeling and change acceleration benefits in your environment.

- DevOps administrators can check the current health of the system.
- DevOps app owners can view the development and quality metrics by app or team.
- Executives and Change managers can track the velocity of changes.

## DevOps Insights dashboard

The DevOps Insights dashboard provides a flexible graphical view of operational and business reports.

DevOps managers, Change managers, and executives can use this dashboard to determine the overall efficiency# and performance of the development teams and processes.

- Monitor activities, KPIs, and metrics using a centralized dashboard.
- Track performance over time using ServiceNow Performance Analytics.
- View break down of the numbers by application, repository, service, business app, work item type and product over time.

For more information, see [DevOps Insights Standard dashboard](#).

## DevOps System Health dashboard

The System Health dashboard lets the DevOps administrator view the overall health of integrations, connectivity status, as well as view trends of inbound event processing data.

For more information, see [DevOps System Health dashboard](#).

## Getting started with DevOps Change Velocity

Learn about the setup process for DevOps Change Velocity that is required before configuring integrations with your toolchain.

After installing the DevOps Change Velocity application, use the DevOps Change workspace to perform the tasks listed below. The default Playbook provides a task-oriented view and guides you through the various steps thus automating most of the manual steps while setting up your applications, roles, and discovering and configuring your pipelines. For more information about the Playbooks, see [Playbook Experiences](#). If you have not enabled the Next Experience UI, you can still perform the following tasks using the Classic UI. For more information about migrating to the Next Experience UI and configuring workspaces, see [Next Experience UI](#)

### 1. Install DevOps Change Velocity.

#### 2. Assign DevOps roles. See [Assign a role to a user](#).

Roles that are installed with DevOps Change Velocity are listed in [Components installed with DevOps Change Velocity](#).

#### 3. Set up system accounts in DevOps Change Velocity.

#### 4. Integrate your third-party (plan, code, orchestration, test, security, and artifact) tools for end-to-end change traceability and automation. We recommend integrating with at least one plan, one code and one orchestration tool to get core value. See [Integrate your third-party tools to enable change outcomes](#).

#### 5. Create your DevOps application to track and manage its lifecycle in ServiceNow DevOps Change Velocity. An application roughly correlates to a team. See .

#### 6. Accelerate your DevOps change process using DevOps Change Velocity. See [Accelerating your DevOps change process](#).

## Install DevOps Change Velocity

Install the DevOps Change Velocity application from ServiceNow Store if you have the admin role.

### Before you begin

Ensure that the application and all of its associated store applications have valid ServiceNow entitlements. For more information, see [Get entitlement for a ServiceNow product or application](#).

Role required: admin

### About this task

To install the DevOps application on a sub-prod instance, navigate to the DevOps Change Velocity app on the ServiceNow Store and click **Request Install** to send your request.

## Procedure

1. Navigate to **System Applications > All Available Applications > All**.
2. Click the **Not Installed** tab to view a list of applications available for installation.
3. Find the DevOps Change Velocity application using the filter criteria and search bar.

You can search for the application by its name or ID (sn\_devops\_chgvcty). If you cannot find an application, you may have to request it from ServiceNow store.

Visit the [ServiceNow Store](#) website to view all the available apps and for information about submitting requests to the store. For cumulative release note information for all released apps, see the [ServiceNow Store version history release notes](#).

4. Click **Install**.
5. In the Application installation dialog box, review the application dependencies.

If your application requires other applications, install them first if they are not already installed.

Installing your application also automatically installs the dependent applications or plugins if they are not installed already.
6. If you want to install demo data, select **Load demo data**.

Demo data includes sample records that describe application features for common use cases. Load demo data when you first install the application on a development or test instance.

**Note:** If you don't load the demo data for a store application during installation, you can not load it later.
7. Click **Install**.

Based on your connection, the installation procedure can take almost an hour or more to complete.

## Result

Installing DevOps Change Velocity also installs the following:

- Dependent applications:
  - DevOps Data Model
  - DevOps Integrations
  - DevOps Insights
- User roles, tables, scheduled jobs, and other components.

See [Components installed with DevOps Change Velocity](#).

## What to do next

- Assign roles to users based on their responsibilities. See [Assign a role to a user](#).

Roles that are installed with DevOps Change Velocity are listed in [Components installed with DevOps Change Velocity](#).

- Set the `com.snc.change_management.change_model.type_compatibility` property to **True**. This setting allows Change Requests to be created with both the type-based workflow and Change models.

For more information, see [Change Models properties](#)

## Set up system accounts in DevOps Change Velocity

Set up the integration user and connection and credentials accounts to enable connecting to and integrating with your third-party tools.

### Before you begin

Role required: admin

### About this task

This is a one-time setup. You can set the integration user password and create connection and credential record either using Workspace or Classic UI.

- Set up integration user: The DevOps integration user is the ServiceNow system user that your third-party orchestration tools use to send data to DevOps Change Velocity.

This is required to configure Azure DevOps (create webhooks automatically) and Jenkins (enable DevOps configuration - **Manage Jenkins > Configure System > ServiceNow DevOps Configuration**). The DevOps admin or tool owner requires this password to

configure so that real-time data is sent to DevOps Change Velocity. You must use this credential in your pipeline when using any ServiceNow DevOps extensions (in ADO) or custom actions (in GitHub Actions) or methods (in Jenkins). For example, if you have a change custom action in the GitHub Actions workflow, you'll need to use the integration user credential in the custom action to authenticate the change REST API to persist the data coming from the tool in DevOps Change Velocity. In Azure DevOps, the service connection will be created automatically with this credential.

The default integration user credential will be applicable for all the tools. However, if you want to set up separate integration user credential per tool connection you can also do the same but ensure that the user account has the `sn_devops.integration` role assigned.

- Configure the CreateDevOpsTool connection and credential alias: The user configuring connection and credential records must have the `connection_admin` role. You can use the DevOps System user that is available by default. You can also select any other user who already has the `connection_admin` role. If the user doesn't have the role, it will be automatically granted while configuring. DevOps CreateDevOpsTool Connection & Credential alias is required for connection to external DevOps tools. While connecting to your tools, DevOps uses the alias to create secure connection and credential records on the Now Platform where the tool access credentials are stored and accessed.

Ensure that the application scope is set to **DevOps Data Model** to configure the alias.

### Procedure

- Set the integration user password in one of the following ways.

Option	Steps
Using workspace	<ol style="list-style-type: none"> <li>Navigate to <b>Workspaces &gt; DevOps Change Workspace</b>.</li> <li>From the Accounts and users widget on the Home page, select <b>Set up system ac-</b></li> </ol>

Option	Steps
Using Classic UI	<p><b>c.</b> In the DevOps integration user account section, select <b>Set new password</b>.</p> <p><b>d.</b> By default, the DevOps integration user is selected. You can also select another user from the list.</p> <p><b>i Note:</b> The <b>Username</b> field will only list the users that have the sn_devops.integration role.</p> <p><b>e.</b> Enter a password for the user and select <b>Set password</b>.</p> <p>Set a password for the DevOps integration user by navigating to <b>All &gt; User Administration &gt; Users</b>.</p> <p>For more information, see <a href="#">Set password for</a></p> 

## 2. Set up the connection and credential record in one of the following ways.

Option	Steps
Using Workspace	<p><b>a.</b> Navigate to <b>Workspaces &gt; DevOps Change Workspace</b>.</p> <p><b>b.</b> From the Accounts and users widget on the Home page, select <b>Set up system accounts</b>. You will be redirected to <b>Administration &gt; Set up system accounts</b>.</p> <p><b>c.</b> In the Configure CreateDevOpsTool Connection &amp; Credential alias section, select <b>Configure alias</b>.</p> <p><b>d.</b> In the <b>Username</b> field, select a user. By default, the DevOps System user is selected.</p> <p>You can choose another user. The user that you select is automatically assigned the connection_admin role.</p> <p><b>e.</b> In the <b>Password</b> field, enter the password for this user.</p> <p>You can use an existing password or create a new one for this user account.</p> <p><b>f.</b> Select <b>Configure alias</b>.</p>

Option	Steps
Using Classic UI	<ol style="list-style-type: none"> <li>a. Navigate to <b>All &gt; Connection &amp; Credential Aliases</b>.</li> <li>b. Select the <b>CreateDevOpsTool</b> alias.</li> <li>c. From the Connections related list, select <b>New</b>.</li> <li>d. Enter a name of your choice.</li> <li>e. Select an existing credential or create one using the lookup list.</li> <li>f. Enter the connection URL in the format <b>https://&lt;instance name&gt;.service-now.com/</b></li> <li>g. Select <b>Submit</b>.</li> </ol>

### What to do next

Start [Integrating DevOps Change Velocity with third-party tools](#).

## Integrating DevOps Change Velocity with third-party tools

Integrate your third-party (plan, code, orchestrate, test, security, and artifact) tools for end to end change traceability and automation.

You must onboard a minimum of one planning, one coding, and one orchestration tool to enable change outcomes.

Use the workspace to onboard a tool for a guided experience.

**Note:** You must enable [Next Experience](#)  to use the DevOps Playbook.

Alternatively, you can use the Catalog or Classic experience.

### Jira integration with DevOps Change Velocity

Connect your Jira instance to discover plans and enable real-time notifications for projects, stories, features, and epics.

### Overview

When you connect and configure a Jira instance, all existing work items are imported through Plans and a webhook is established to ensure updates are tracked real-time. You can then associate these planning objects to application for end-to-end traceability.

### Get started

Use one of the following options to onboard Jira. For a guided experience, use the workspace to onboard a tool. Alternatively, you can use the Service Catalog or Classic experience.

#### Onboard Jira to DevOps Change Velocity — Workspace

Connect and configure your Jira instance to discover available plans and register webhooks to track projects, stories, features, and epics.

## Before you begin

Complete the tasks in [Getting started with DevOps Change Velocity](#).

**Note:** From 18th January 2023, Jira has extended the length of API tokens for Atlassian accounts. You must increase the maximum password value to more than 255 in the discovery\_credentials table to accommodate the extended character length. For more information, see the [KB1269878](#) kb article and [Atlassian documentation](#).

Role required: sn\_devops.admin or sn\_devops.tool\_owner

## Procedure

- Navigate to **Workspaces > DevOps Change Workspace** and use one of the following options to open the Playbook to onboard Jira.

Option	Steps
Homepage	<ul style="list-style-type: none"> <li>a. Select the <b>Connect tool</b> widget</li> <li>b. On the# Connect to a tool# modal, select Jira from the <b>Plan</b> category.</li> </ul>
Applications module	<ul style="list-style-type: none"> <li>a. Select <b>Applications ()</b>.</li> <li>b. Select an existing application, or create one. To create an application, see <a href="#">Create an application in DevOps Change Velocity</a></li> <li>c. From the# Recommended actions# pane, select the#<b>Connect a tool</b> card.</li> <li>d. On the# Connect to a tool# modal, select Jira from the <b>Plan</b> category.</li> </ul>
Tools module	<ul style="list-style-type: none"> <li>a. Select <b>Tools ()</b>.</li> <li>b. From the Tools list, select <b>Planning tools</b>.</li> <li>c. Select <b>Connect Planning tool</b>.</li> <li>d. On the# Connect to a tool# modal, select <b>Jira</b>.</li> </ul>

- In the **Tool name** field, enter a name for the tool.

This name is used to identify your Jira instance from the list of tools connected to DevOps Change Velocity.

- Select **Next**.

The DevOps playbook page opens to help you complete the onboarding tasks.

- Complete the connection and configuration of your Jira tool using the playbook.

- Enter your Jira instance URL.
- Enter the username and password or access token to access this instance.

The Jira user that you use here must have the Jira Administrators permissions.

- c. Optional: If your Jira instance is attached to a MID Server, select the **MID Server** option and enter its details.

(Optional) For more information about MID server, see [MID Server selection](#)

- d. Select **Connect**.

Permission checks are run on the credentials that you entered. Permissions required and permissions that are available are displayed in the pop-up modal. You can choose to continue with the tool connection even if you don't have all the required permissions.

- e. Select **Continue** or **Connect Anyway**.

If the connection is successful, the tool is created in the ServiceNow instance and connected to your Jira instance.

- f. Specify the access for the tool.

- i. If you want to control access to the tool, add the groups that must be given access to the tool in the **Maintained by** field.

The tasks these users in the groups can perform depends on the role assigned to them.

- DevOps Tool Owner role: Can view and edit the tool.
- DevOps App Owner role: Can view the tool and can associate, discover, import historical data, and modify pipeline steps (if applicable) of the tool's objects (such as plans, repositories, and pipelines).
- DevOps Administrator role: Can edit all tools.
- Other DevOps roles: Can view the tool.

**i Note:** If you don't select a group and skip this step, all users with the DevOps Tool Owner role will be able to edit the tool.

- ii. If you choose to control access to the tool, the **All App Owners can view and associate tool objects to applications** option becomes available for selection.

This option enables all users having the DevOps App Owner role to access the tool. If selected, they'll be able to view, associate, discover, import historical data, and modify pipeline steps (if applicable) of the tool's objects.

- iii. Select **Assign**.

- g. To send data in real time between DevOps and your Jira instance, select **Configure**.

This action does the following:

- Registers a webhook between your Jira and ServiceNow instances, and enables real-time data transfer between the two.
- Discovers all available projects from your Jira instance.

All the discovered Jira projects are listed as Plans.

If you don't choose to configure now, you can enable nightly polling later to fetch data for any tracked plans by setting the *Enable Polling* system property to **Yes**.

- h. Select the plans that you want to track the updates for and select **Next**.

After the tool onboarding is complete, the work items only for these selected plans are automatically imported.

- i. From the **Summary** page, select **View tool record** to review the details of the connected instance and the plans discovered from it.

## Result

You've successfully onboarded your Jira tool to DevOps Change Velocity.

## What to do next

From the tool record page, you can:

- See the details and status of the tool.
- **Discover** plans and **Configure** webhooks.
- Assign groups to control access to the tool using the **Maintained by** field.
- Check credential permissions and update credentials for the tool. For more details, see [Check permissions and update credentials for tools — Workspace](#).
- From the **Plans** tab, select a plan to view its details such as imported work items, features, and others.
- To import historical data from the plans, associate the plans with an application, and import the data. For more information, see [Associate tool objects to applications - Workspace](#).

## Onboard Jira to DevOps Change Velocity — Service Catalog

Create, connect, discover, and configure your Jira instance using the ServiceNow Service Catalog.

### Before you begin

Role required: sn\_devops.admin or sn\_devops.tool\_owner

### Procedure

1. Navigate to **All > Service Catalog > Catalog Definitions > Maintain Items** and search for DevOps.

**Note:** You can also access the service catalog from the Employee Center or Service portal.

2. From the DevOps catalog items, select and activate **DevOps App Onboarding** and **DevOps Tool Onboarding**.

**3.** After activating, select **DevOps Tool Onboarding** and select **Try it**.

**4.** In the DevOps Tool Onboarding form, enter the tool details:

Field	Description
Tool name	Name for your Jira integration.
Tool integration	Select Jira.
Tool URL	URL of your Jira instance.
Tool username	Username for your Jira account.
Tool password/ Access token	Password or Atlassian API token.
Do you wish to configure webhook for the tool?	Select this option if you want to configure webhooks for all the projects discovered in your Jira instance.
Use MidServer	Optional. Select#MID Server#for an on-premises tool that is attached to a#MID Server. Application is automatically set to #DevOps and capability is set to REST.

**5.** Select **Order Now**.

A request is created. When the request is approved, the tool is created, connected, configured, and discovered.

**6.** From the DevOps catalog items, select **DevOps App Onboarding** and select **Try it**.

**7.** In the DevOps App Onboarding form, enter the details:

Are you creating a new app or adding to an existing one?	Select from the options whether to create a new app or use an existing app.
App	Enter the name for the app that you're creating or using.
Onboarding pipelines	Leave empty.
Onboarding repositories	Leave empty.
Onboarding plans	Enter the connected Jira tool name.
Plans	Select the plans for which you want to import historical data.
Import from and Import to	Select the dates for which you want to import the data. By default, the last 30 days are selected. You can choose to import data for a maximum of 90 days.

**8.** Select **Order Now**.

A request is created. When the request is approved, the plan objects are associated to the app record, which enables real-time tracking. Historical data is also imported for the selected plans. The **Track** field is automatically enabled for imported plans.

## Onboard Jira to DevOps Change Velocity — Classic

Connect your Jira instance to discover, configure, and import projects and work items such as stories, epics, and features.

### Before you begin

Complete the tasks in [Getting started with DevOps Change Velocity](#).

Role required: sn\_devops.admin or sn\_devops.tool\_owner

### Procedure

**1.** Connect to a Jira instance.

**a.** Navigate to **DevOps > Tools > Create new**.

**b.** In the **Tool name** field, enter a name and fill in tool details.

Field	Description
Tool Integration	Jira
Tool URL	URL of your Jira instance.
Credential type	<p>Basic Auth</p> <p>Enter the username and password for the provided Jira URL.</p> <p>The Jira user that you use here must have the Jira Administrators permissions.</p>

**c.** Optional: If your Jira instance is attached to a MID Server, select the **MID Server** option and enter its details.

(Optional) For more information about MID server, see [MID Server selection](#) ↗

**d. Select Submit.**

The tool is automatically connected using a connection alias and HTTP tool connection (Basic Auth credential).

On successful tool creation, you're taken to the tool record page.

**2.** If you want to control access to the tool, add the groups that must be given access to the tool in the **Maintained by** field on the **Access** tab.

The tasks the users in the groups can perform depends on the roles assigned to them.

- DevOps Tool Owner role: Can view and edit the tool.
- DevOps App Owner role: Can view the tool and can associate, discover, import historical data, and modify pipeline steps (if applicable) of the tool's objects (such as plans, repositories, and pipelines).
- DevOps Administrator role: Can edit all tools.
- Other DevOps roles: Can view the tool.

**Note:** Only groups containing users with DevOps roles are available for selection in the **Maintained by** field.

The **All App Owners can view and associate tool objects to applications** option becomes available for selection if you choose to restrict access to the tool. This option enables all users having the DevOps App Owner role to access the tool. If selected, they'll be able to view, associate, discover, import historical data, and modify pipeline steps (if applicable) of the tool's objects.

**3.** Select **Discover**# to discover all existing projects from Jira.

**4.** Select **Configure** to configure webhooks.

This action does the following:

- Registers a webhook between your Jira and ServiceNow instances, and enables real-time data transfer between the two.
- Discovers all available projects from your Jira instance.

All the discovered Jira projects are listed as Plans.

If you don't choose to configure now, you can enable nightly polling later to fetch data for any tracked plans by setting the *Enable Polling* system property to **Yes**.

**Note:** If you don't have admin privileges for Jira to allow automatic configuration of the webhook URL, request your Jira admin to configure it for you. This action would require creating and configuring the webhook URL manually in your tool instance. After the webhook is configured in the tool, select **Enter Manual Configuration Mode** to connect to Jira manually, then exit the manual configuration mode.

**5.** Select the **#Import#** related link to import historical data from Jira projects.

Imported work items are added to the corresponding related lists.

## Rally integration with DevOps Change Velocity

Connect the Broadcom Rally planning tool with DevOps Change Velocity to discover plans and configure real-time notifications to enable change traceability and automation.

### Overview

With this integration, you can configure and expose data like projects, user stories, and defects from Rally. You can then associate these planning objects to application for end-to-end traceability.

### Install Rally plugin

You must install the Rally application from [ServiceNow Store](#) .

## **Note:**

- The default object types configured for Rally are "ObjectTypes":["HierarchicalRequirement","Defect","Feature"].
- The file script include **Rally state mapping helper** is editable. You can override this file with your specific customization as required.
- Scheduledstate in Rally is used when mapping object state to internal ServiceNow work item state.
- For tool mappings, see [Tool mappings](#).

## Get started

Use one of the following options to onboard Rally. For a guided experience, use the workspace to onboard a tool. Alternatively, you can use the Service Catalog or Classic experience.

### Onboard Rally to DevOps Change Velocity — Workspace

Connect your Rally instance using the playbook in DevOps Change workspace to discover plans.

#### Before you begin

Role required: sn\_devops.admin or sn\_devops.tool\_owner

#### Procedure

1. Navigate to **Workspaces > DevOps Change Workspace** and use one of the following options to open the Playbook to onboard Rally.

Option	Steps
Homepage	<ol style="list-style-type: none"> <li>Select the <b>Connect tools</b> widget.</li> <li>On the# Connect to a tool# modal, select Rally from the appropriate category (Plan, Code, Orchestration, Artifact, or Software quality).</li> </ol>
Applications module	<ol style="list-style-type: none"> <li>Select <b>Applications ()</b>.</li> <li>Select an existing application, or create one. To create an application, see <a href="#">Create an application in DevOps Change Velocity</a>.</li> <li>From the# Recommended actions# pane, select the#<b>Connect a tool</b> card.</li> <li>On the# Connect to a tool# modal, select Rally from the appropriate category (Plan, Code, Orchestration, Artifact, or Software quality).</li> </ol>
Tools module	<ol style="list-style-type: none"> <li>Select <b>Tools ()</b>.</li> <li>From the Tools list, select the appropriate category (Plan, Code, Orchestration, Artifact, or Software quality).</li> </ol>

Option	Steps
	<p>c. Select <b>Connect tool</b>.</p> <p>d. On the# Connect to a tool# modal, select <b>Rally</b>.</p>

2. Enter a tool name to identify your tool and click **Next**.
3. On the Enter Rally instance details playbook activity, enter the following details:
  - a. In the **URL of the Rally project** field, enter your Rally project URL.
  - b. In the **Password or access token** field, enter the API token for your Rally instance.
  - c. (Optional) If your Rally instance is attached to a MID Server, select the **Use MID Server** option and enter its details.  
A MID server is required if your tool instance is hosted on-prem. For more information about MID server, see [MID Server selection](#).
4. Click **Connect**.  
On successful connection, the tool is created in ServiceNow and connected to your Rally instance. Plans are discovered and you can select the plans that you want to track.
5. Specify the access for the tool.
  - a. If you want to control access to the tool, add the groups that must be given access to the tool in the **Maintained by** field.  
The tasks these users in the groups can perform depends on the role assigned to them.
    - DevOps Tool Owner role: Can view and edit the tool.
    - DevOps App Owner role: Can view the tool and can associate, discover, import historical data, and modify pipeline steps (if applicable) of the tool's objects (such as plans, repositories, and pipelines).
    - DevOps Administrator role: Can edit all tools.
    - Other DevOps roles: Can view the tool.

**Note:** If you don't select a group and skip this step, all users with the DevOps Tool Owner role will be able to edit the tool.
  - b. If you choose to control access to the tool, the **All App Owners can view and associate tool objects to applications** option becomes available for selection.  
This option enables all users having the DevOps App Owner role to access the tool. If selected, they'll be able to view, associate, discover, import historical data, and modify pipeline steps (if applicable) of the tool's objects.
  - c. Select **Assign**.
6. On the Select plans to track playbook activity, select the plans that you want to track and select **Next**.

**Note:** You can configure webhooks from the tool details page in the workspace or from the Tool record page.

7. From the **Summary** tab, select **View tool record** to review the details of the connected instance and the plans discovered from it.
  
8. From the **Summary** tab, select **View tool record** to review the details of the connected tool.

### Result

You've successfully onboarded your Rally tool to DevOps Change Velocity.

### What to do next

From the tool record page, you can:

- See the details and status of the tool.
- Discover plans.
- Configure webhooks for real time data exchange:
  1. Select **Configure**.
  2. Select the projects for which you want to configure webhooks and click **Continue**.
  3. Enter your DevOps integration user credentials and click **Send**.
- Assign groups to control access to the tool using the **Maintained by** field.
- Check credential permissions and update credentials for the tool. For more details, see [Check permissions and update credentials for tools — Workspace](#).
- From the **Plans** tab, select a plan to view its details such as imported work items, features, and others.
- To import historical data from the plans, associate the plans with an application, and import the data. For more information, see [Associate tool objects to applications - Workspace](#).

## Onboard Rally to DevOps Change Velocity — Service Catalog

Create, connect, discover, and configure your Rally instance using the ServiceNow Service Catalog.

### Before you begin

Role required: sn\_devops.admin or sn\_devops.tool\_owner

### Procedure

1. Navigate to **All > Service Catalog > Catalog Definitions > Maintain Items** and search for DevOps.
  
- Note:** You can also access the service catalog from Employee Center or Service portal.
  
2. From the DevOps catalog items, select and activate **DevOps App Onboarding** and **DevOps Tool Onboarding**.

3. After activating, select **DevOps Tool Onboarding** and select **Try it**.

4. In the DevOps Tool Onboarding form, enter the tool details:

Field	Description
Tool name	Name for your Rally integration.
Tool integration	Select Rally.
Tool URL	URL of your Rally instance to integrate.
Tool password/ Access token	Enter the API key for Rally.
Use MidServer	Optional. Select#MID Server# for an on-premises tool that is attached to a#MID Server. Application is automatically set to # DevOps# and capability is set to REST.

5. Select **Order Now**.

A request is created. When the request is approved, the tool is created, connected, discovered, and configured.

6. From the DevOps catalog items, select **DevOps App Onboarding** and select **Try it**.

7. In the DevOps App Onboarding form, enter the details:

Are you creating a new app or adding to an existing one?	Select from the options whether to create a new app or use an existing app.
App	Enter the name for the app that you're creating or using.
Onboarding pipelines	Leave empty.
Onboarding repositories	Leave empty.
Onboarding plans	Enter the connected Rally tool name.
Plans	Select the plans for which you want to import historical data.
Import from and Import to	Select the dates for which you want to import the data. By default, the last 30 days are selected. You can choose to import data for a maximum of 90 days.

8. Select **Order Now**.

A request is created. When the request is approved, the plan objects are associated to the app record, which enables real-time tracking. Historical data is also imported for the selected plans. The **Track** field is automatically enabled for imported plans.

### Onboard Rally to DevOps Change Velocity — Classic

With this integration, you can configure and expose data like projects, user stories, and defects from the Rally. You can then associate these planning objects to application for end-to-end traceability.

#### Before you begin

Role required: sn\_devops.admin or sn\_devops.tool\_owner

## Procedure

1. Navigate to **All > DevOps > Tools > Create New**.
2. In the Create DevOps tool form, enter the tool details:

Tool name	Name for the tool you're integrating.
Tool integration	Tool to integrate. Select Rally.
Tool URL	URL of your Rally instance to integrate.
Tool password / Access token	Enter the API key for Rally.
Use MID Server	MID Server is optional. Select MID Server for an on-premises tool that is attached to a MID Server. The Application value is automatically set to DevOps and the Capability value is set to REST. For more information, see <a href="#">MID Server selection</a> .

3. Click **Submit** to connect to your Rally instance.

On successful tool creation, you're taken to the tool record page.

4. If you want to control access to the tool, add the groups that must be given access to the tool in the **Maintained by** field on the **Access** tab.

The tasks the users in the groups can perform depends on the roles assigned to them.

- DevOps Tool Owner role: Can view and edit the tool.
- DevOps App Owner role: Can view the tool and can associate, discover, import historical data, and modify pipeline steps (if applicable) of the tool's objects (such as plans, repositories, and pipelines).
- DevOps Administrator role: Can edit all tools.
- Other DevOps roles: Can view the tool.

**i Note:** Only groups containing users with DevOps roles are available for selection in the **Maintained by** field.

The **All App Owners can view and associate tool objects to applications** option becomes available for selection if you choose to restrict access to the tool. This option enables all users having the DevOps App Owner role to access the tool. If selected, they'll be able to view, associate, discover, import historical data, and modify pipeline steps (if applicable) of the tool's objects.

5. Click **Discover** to import existing projects associated with Rally.  
Plan records are added to the **Plans** related links.
6. To configure webhooks for Rally projects:
  - a. Click **Configure**.
  - b. Select the projects for which you want to configure webhooks and click **Continue**.
  - c. Enter your DevOps integration user credentials.
7. Optional: To import historical data, select a plan and click **Import**.

Select the dates to import historical data from the Rally project. You can see the imported data under **Work Items**. Select a work item to see details including commits made.

**Note:** Scheduledstate in Rally is used when mapping object state to internal ServiceNow work item state.

Formatted id is used for the work item NativeID, for associating work items and commits.

## Agile Development 2.0 integration with DevOps Change Velocity

Integrate ServiceNow Agile Development 2.0 planning tool to track your products and the stories, epics, and release versions associated to it.

### Get started

Use one of the following options to onboard Agile Development 2.0. For a guided experience, use the workspace to onboard a tool. Alternatively, you can use the Service Catalog or Classic experience.

#### Onboard Agile Development 2.0 to DevOps Change Velocity — Workspace

Configure integration with Agile Development 2.0 to enable tracking of stories and epics.

#### Before you begin

Activate the Agile Development 2.0 plugin. For more information, see [Activate Agile Development 2.0](#).

Role required: sn\_devops.admin or sn\_devops.tool\_owner

#### Procedure

1. Navigate to **Workspaces > DevOps Change Workspace** and use one of the following options to open the Playbook to onboard Agile Development 2.0.

Option	Steps
Homepage	<ol style="list-style-type: none"> <li>Select the <b>Connect tool</b> widget</li> <li>On the# Connect to a tool# modal, select Agile Development 2.0 from the <b>Plan</b> category.</li> </ol>
Applications module	<ol style="list-style-type: none"> <li>Select <b>Applications ()</b>.</li> <li>Select an existing application, or create one. To create an application, see <a href="#">Create an application in DevOps Change Velocity</a>.</li> <li>From the# Recommended actions# pane, select the#<b>Connect a tool</b> card.</li> <li>On the# Connect to a tool# modal, select Agile Development 2.0 from the <b>Plan</b> category.</li> </ol>
Tools module	<ol style="list-style-type: none"> <li>Select <b>Tools ()</b>.</li> <li>From the Tools list, select <b>Planning tools</b>.</li> </ol>

Option	Steps
	<p>c. Select <b>Connect Planning tool</b>.</p> <p>d. On the# Connect to a tool# modal, select <b>Agile Development 2.0</b>.</p>

**2.** In the **Tool name** field, enter a name for the tool.

**3.** Select **Next**.

The DevOps playbook opens to help you complete the onboarding tasks.

**4.** Select **Connect**.

The tool is connected and discovered plans are listed.

**5.** Specify the access for the tool.

- a. If you want to control access to the tool, add the groups that must be given access to the tool in the **Maintained by** field.

The tasks these users in the groups can perform depends on the role assigned to them.

- DevOps Tool Owner role: Can view and edit the tool.
- DevOps App Owner role: Can view the tool and can associate, discover, import historical data, and modify pipeline steps (if applicable) of the tool's objects (such as plans, repositories, and pipelines).
- DevOps Administrator role: Can edit all tools.
- Other DevOps roles: Can view the tool.

**i Note:** If you don't select a group and skip this step, all users with the DevOps Tool Owner role will be able to edit the tool.

- b. If you choose to control access to the tool, the **All App Owners can view and associate tool objects to applications** option becomes available for selection.

This option enables all users having the DevOps App Owner role to access the tool. If selected, they'll be able to view, associate, discover, import historical data, and modify pipeline steps (if applicable) of the tool's objects.

**c. Select Assign.**

**6.** Select the plans that you want to track and click **Next**.

If you don't choose to select plans now, you can enable nightly polling later to fetch the data for any tracked plans by setting the *Enable Polling* system property to **Yes**.

- From the **Summary** page, select **View tool record** to review the details of the connected instance and the plans discovered from it.

## Result

You've successfully onboarded your Agile Development 2.0 tool to DevOps Change Velocity.

### Onboard Agile Development 2.0 to DevOps Change Velocity — Classic

Connect to Agile Development 2.0 and discover plans from it.

#### Before you begin

Role required: sn\_devops.admin or sn\_devops.tool\_owner

#### Procedure

- Create a tool record in DevOps to automatically connect to Agile Development 2.0 and get the webhook URL.

- Navigate to **DevOps > Tools > Create new**.

- On the form, fill in the fields.

Tool name	Name of your choice to identify the tool.
Tool Integration	Select <b>Agile Development 2.0</b>

- Click **Submit**.

On successful tool creation, you're taken to the tool record page.

- If you want to control access to the tool, add the groups that must be given access to the tool in the **Maintained by** field on the **Access** tab.

The tasks the users in the groups can perform depends on the roles assigned to them.

- DevOps Tool Owner role: Can view and edit the tool.
- DevOps App Owner role: Can view the tool and can associate, discover, import historical data, and modify pipeline steps (if applicable) of the tool's objects (such as plans, repositories, and pipelines).
- DevOps Administrator role: Can edit all tools.
- Other DevOps roles: Can view the tool.

**Note:** Only groups containing users with DevOps roles are available for selection in the **Maintained by** field.

The **All App Owners can view and associate tool objects to applications** option becomes available for selection if you choose to restrict access to the tool. This option enables all users having the DevOps App Owner role to access the tool. If selected, they'll be able to view, associate, discover, import historical data, and modify pipeline steps (if applicable) of the tool's objects.

- Click **Discover** to discover Products from Agile Development 2.0.

Discovered records are added to the Plans related list.

4. Import data for the discovered plans by clicking the **Import** related link from a plan's record.

Epics, stories, and release versions associated to this plan are listed in the respective related lists.

- Work items related list shows stories and defects.

**Native State** and **Native Type** fields of the work item contain the original state and type values from the source tool.

- Plan Versions related list shows releases.
- Features related list shows epics.

## Azure DevOps integration with DevOps Change Velocity

Integrate Azure DevOps and discover boards, repositories, and pipelines to enable change automation and traceability.

### Overview

The following Azure DevOps tools are supported:

- Azure Boards (planning)
- Azure Repositories (coding)
- Azure Pipelines (orchestration)
  - Build (CI) pipelines — agent and agentless (server) jobs
  - Release (CD) pipelines

Capture tags from Azure DevOps coding tool commits.

Jenkins also supports testing capabilities with JUnit. [Test tool integration](#) lets you view test results in DevOps Change Velocity for Azure DevOps unit, functional, and performance tests.

### Azure DevOps extension

You can use the **ServiceNow DevOps** extension for Azure DevOps on [Visual Studio Marketplace](#) to integrate your Azure pipeline with DevOps Change Velocity.

The **ServiceNow DevOps** extension includes:

- ServiceNow DevOps service connection
- ServiceNow DevOps Release Gate
- Azure build (CI) pipeline agent and server job custom tasks
- ServiceNow DevOps Build Sonar Registration (for Build pipelines)
- ServiceNow DevOps Release Sonar Registration task (for Release pipelines)

### Bulk commits in Azure DevOps

Large commits are supported with Azure DevOps.

To support large commits, perform these actions:

- Install the ServiceNow IntegrationHub Action Template - Data Stream (com.glide.hub.action\_type.datastream) plugin.
- For optimal performance, disable flow logging by setting the Flow Designer `com.snc.process_flow.reporting.level` property to **Off**.
- For MID Server settings, view the [MID Server support for Data Stream actions](#) section.

The Azure DevOps data stream can currently process up to 8000-9000 commits per code push. The number of run commits listed for a task execution are limited to 200.

## Get started

Use one of the following options to onboard Azure DevOps. For a guided experience, use the workspace to onboard a tool. Alternatively, you can use the Service Catalog or Classic experience.

### Onboard Azure DevOps to DevOps Change Velocity — Workspace

Connect and configure your Azure DevOps instance using DevOps Change workspace to collect data for planning, coding, orchestration, artifact, and software quality functions.

#### Before you begin

Role required: sn\_devops.admin or sn\_devops.tool\_owner

#### About this task

You can connect to Azure DevOps at your organization level or at individual project level. If you're connecting at the project level, for each project at your organization, you must repeat the connection process. After connecting to an instance of the tool, you can configure additional settings that enable DevOps to import pipelines, task execution records, and step execution records.

**Note:** DevOps Change Velocity uses the term instance to refer to a particular occurrence of a tool. Azure DevOps uses the term project instead.

#### Procedure

1. Navigate to **Workspaces > DevOps Change Workspace** and use one of the following options to open the Playbook to onboard Azure DevOps.

Option	Steps
Homepage	<ol style="list-style-type: none"> <li>Select the <b>Connect tools</b> widget.</li> <li>On the# Connect to a tool# modal, select Azure DevOps from the appropriate category (Plan, Code, Orchestration, Artifact, or Software quality).</li> </ol>
Applications module	<ol style="list-style-type: none"> <li>Select <b>Applications ()</b>.</li> <li>Select an existing application, or create one. To create an application, see <a href="#">Create an application in DevOps Change Velocity</a>.</li> <li>From the# Recommended actions# pane, select the#<b>Connect a tool</b> card.</li> </ol>

Option	Steps						
Tools module	<p>d. On the# Connect to a tool# modal, select Azure DevOps from the appropriate category (Plan, Code, Orchestration, Artifact, or Software quality).</p> <p>a. Select <b>Tools ()</b>.</p> <p>b. From the Tools list, select the appropriate category (Plan, Code, Orchestration, Artifact, or Software quality).</p> <p>c. Select <b>Connect tool</b>.</p> <p>d. On the# Connect to a tool# modal, select <b>Azure DevOps</b>.</p>						
	<p>2. Select whether you're connecting to your organization or to a project in the organization.</p> <table border="1" data-bbox="219 723 1387 1758"> <thead> <tr> <th data-bbox="219 723 806 776">Connecting to</th><th data-bbox="806 723 1387 776">Steps</th></tr> </thead> <tbody> <tr> <td data-bbox="219 776 806 1303">Organization</td><td data-bbox="806 776 1387 1303"> <p>Connect directly at your Azure DevOps organization level. All the projects within the organization are discovered, and you can choose to configure multiple projects within the organization. You can manage the tool at the organization level.</p> <p>a. Select <b>Connect an organization</b> from the drop-down list.</p> <p>b. Enter the Azure DevOps URL for the organization.</p> <p>c. In the <b>Tool name</b> field, enter a name for the tool.</p> </td></tr> <tr> <td data-bbox="219 1303 806 1758">Project</td><td data-bbox="806 1303 1387 1758"> <p>Connect directly at the project level. If you have multiple projects within the organization, you must connect each of them separately.</p> <p>a. Select <b>Connect a project</b> from the drop-down list.</p> <p>b. Enter the Azure DevOps URL for the project.</p> <p>c. In the <b>Tool name</b> field, enter a name for the tool.</p> </td></tr> </tbody> </table>	Connecting to	Steps	Organization	<p>Connect directly at your Azure DevOps organization level. All the projects within the organization are discovered, and you can choose to configure multiple projects within the organization. You can manage the tool at the organization level.</p> <p>a. Select <b>Connect an organization</b> from the drop-down list.</p> <p>b. Enter the Azure DevOps URL for the organization.</p> <p>c. In the <b>Tool name</b> field, enter a name for the tool.</p>	Project	<p>Connect directly at the project level. If you have multiple projects within the organization, you must connect each of them separately.</p> <p>a. Select <b>Connect a project</b> from the drop-down list.</p> <p>b. Enter the Azure DevOps URL for the project.</p> <p>c. In the <b>Tool name</b> field, enter a name for the tool.</p>
Connecting to	Steps						
Organization	<p>Connect directly at your Azure DevOps organization level. All the projects within the organization are discovered, and you can choose to configure multiple projects within the organization. You can manage the tool at the organization level.</p> <p>a. Select <b>Connect an organization</b> from the drop-down list.</p> <p>b. Enter the Azure DevOps URL for the organization.</p> <p>c. In the <b>Tool name</b> field, enter a name for the tool.</p>						
Project	<p>Connect directly at the project level. If you have multiple projects within the organization, you must connect each of them separately.</p> <p>a. Select <b>Connect a project</b> from the drop-down list.</p> <p>b. Enter the Azure DevOps URL for the project.</p> <p>c. In the <b>Tool name</b> field, enter a name for the tool.</p>						

### 3. Select **Next**.

The DevOps playbook opens to help you complete the onboarding tasks.

### 4. Enter the instance details for your tool.

- a. Enter the password or access token to access this instance.  
For information on creating a PAT, see [Personal access token \(PAT\)](#).
- b. Optional: If your tool instance is attached to a MID Server, select the **MID Server** option and enter its details.  
(Optional) For more information about MID server, see [MID Server selection](#).

**c. Select Connect.**

Permission checks are run on the credentials that you entered. Permissions required and permissions that are available are displayed in the pop-up modal. If the permission check failed, log in to the management portal for the tool, set appropriate permissions, and then try again to connect to the tool.

You can choose to continue with the tool connection even if you don't have all the required permissions.

**i Note:**

- When onboarding a Project, the **Project Administrators** privilege requires the owner of the PAT to be a member of the project's **Project Administrators** group.
- When onboarding an Organization, the **Project Administrators** privilege requires the owner of the PAT to be a member of the organization's **Project Collection Administrators** group.

**d. Select Continue or Connect Anyway.**

**5. Specify the access for the tool.**

- a. If you want to control access to the tool, add the groups that must be given access to the tool in the **Maintained by** field.

The tasks these users in the groups can perform depends on the role assigned to them.

- DevOps Tool Owner role: Can view and edit the tool.
- DevOps App Owner role: Can view the tool and can associate, discover, import historical data, and modify pipeline steps (if applicable) of the tool's objects (such as plans, repositories, and pipelines).
- DevOps Administrator role: Can edit all tools.
- Other DevOps roles: Can view the tool.

**i Note:** If you don't select a group and skip this step, all users with the DevOps Tool Owner role will be able to edit the tool.

- b. If you choose to control access to the tool, the **All App Owners can view and associate tool objects to applications** option becomes available for selection.

This option enables all users having the DevOps App Owner role to access the tool. If selected, they'll be able to view, associate, discover, import historical data, and modify pipeline steps (if applicable) of the tool's objects.

**c. Select Assign.**

## 6. Configure the Azure DevOps instance to send data to DevOps Change Velocity.

Choose to send data by either nightly polling or configuring webhooks to send real-time data.

- Webhooks: Enable real-time notifications for your pipeline executions. Real-time notifications are ideal to maintain the most up-to-date information particularly for automating change requests.

To use webhooks, select **Configure**.

- Nightly polling: If you don't choose to configure now, you can enable nightly polling later to fetch data for any tracked plans by setting the *Enable Polling* property to **Yes**.

Connecting to	Steps
<b>Organization</b>	<p><b>a.</b> Enter the DevOps integration user name and password.</p> <p>For information about creating the DevOps integration user and password, see <a href="#">Set up system accounts in DevOps Change Velocity</a>.</p> <p><b>b.</b> Select the projects for which you want to configure webhooks.</p> <p><b>c.</b> Select <b>Configure</b>.</p> <p>Webhook configuration and discovery happen in the background, and you're taken to the Summary page.</p>
<b>Project</b>	<p><b>a.</b> Enter the DevOps integration user name and password.</p> <p>For information about creating the DevOps integration user and password, see <a href="#">Set up system accounts in DevOps Change Velocity</a>.</p> <p><b>b.</b> Select <b>Configure Azure DevOps</b>.</p> <p>Plans, repositories, and pipelines are discovered.</p> <p><b>c.</b> Select the plans that you want to track the updates for and select <b>Next</b>.</p> <p><b>d.</b> Select the repositories that you want to track the updates for and select <b>Next</b>.</p>

Connecting to	Steps
	<p><b>e.</b> Select the pipelines that you want to track the updates for and select <b>Next</b>.</p> <p>If you're connecting this tool from an application, then the selected pipelines are also associated to the application and tracked. You can associate pipelines to an application at any time.</p> <p><b>f.</b> Assign services to pipeline.</p> <p>Even though this step is optional, completing it as part of this task enables the DevOps Insights dashboards to show more meaningful data immediately.</p> <p>For each pipeline, specify:</p> <ul style="list-style-type: none"> <li>▪ Step type, such as Prod Deploy or Build and Test. At a minimum, specify <b>Prod deploy</b> for steps that represent the production deployment. Prod deploy steps enable DevOps Change Velocity to identify successful pipeline executions as production deployments.</li> <li>▪ Service, such as the CMDB application service that the pipeline step always maps to. The application service maps approximately to the environment. Service information enables DevOps Change Velocity to identify and report on operational metrics such as incidents, outages, and so on. If you use the same pipeline step to deploy to different environments, you can leave this field empty.</li> </ul> <p>You're taken to the Summary page.</p>

- From the Summary page, select **View tool record** to review the details of the connected tool.

## Result

You've successfully onboarded your Azure DevOps tool to DevOps Change Velocity.

## What to do next

From the **Projects** tab on the tool record page, select a project to navigate to the project record page. From here, you can discover project objects, and configure webhooks for the project.

- Select **Discover** to discover the project objects, including existing plans (boards), repositories, and pipelines.
- If you created the tool directly at the project level, then selecting **Discover projects** from the **Projects** tab of the tool record page will discover all the projects in your organization as well.
- Select **Configure** and enter the integration user credentials to configure webhooks for the project.
- If you're on the tool records page, selecting **Configure projects** and entering the integration user credentials gives the list of unconfigured projects in your organization. Select the projects that you want webhooks configured for and select **Configure**.
- To import historical data to the project objects like plans, repositories, or pipelines, associate the objects with an application, and import the data. For more information, see [Associate tool objects to applications - Workspace](#).

## Onboard Azure DevOps to DevOps Change Velocity — Service Catalog

Create, connect, discover, and configure your Azure DevOps instance using the ServiceNow Service Catalog.

### Before you begin

Role required: sn\_devops.admin or sn\_devops.tool\_owner

### Procedure

1. Navigate to **All > Service Catalog > Catalog Definitions > Maintain Items** and search for DevOps.
- Note:** You can also access the service catalog from Employee Center or Service portal.
2. From the DevOps catalog items, select and activate **DevOps App Onboarding** and **DevOps Tool Onboarding**.
3. After activating, select **DevOps Tool Onboarding** and select **Try it**.
4. In the DevOps Tool Onboarding form, enter the tool details:

Field	Description
Tool name	Name for your Azure DevOps integration.
Tool integration	Select Azure DevOps.
Connect to an organization or project	Select from the list. <ul style="list-style-type: none"> <li>◦ <b>Connect an organization:</b> Connect directly at your Azure DevOps organization level. All the projects within the organization will be discovered, and you can choose to configure multiple projects within the organization.</li> <li>◦ <b>Connect a project:</b> Connect directly at the project level.</li> </ul>
Tool URL	Azure DevOps organization URL (for example, <a href="https://dev.azure.com/&lt;your">https://dev.azure.com/&lt;your</a>

Field	Description
	organization> or the Azure DevOps project URL (for example, https://dev.azure.com/<your organization>/<your project>).
Tool username	Azure DevOps instance username.
Tool password/ Access token	<p>Personal access token (PAT) <a href="#">↗</a> for your Azure DevOps instance.</p> <p><b>i Note:</b> Only a personal access token is supported.</p> <p>When you generate a Personal access token (PAT) for Azure DevOps, you must select the scopes to authorize if you aren't granting complete access. See <a href="#">Azure DevOps PAT scopes for DevOps</a>.</p>
Do you wish to configure webhook for this tool?	<p>Option to enable configuring webhooks automatically for Azure DevOps. Select to enable.</p> <p><b>i Note:</b> This option isn't available if you're connecting at the organization level. You can configure webhooks from the tool records page.</p>
Integration username	<p>This field is available only when the option to configure webhook is selected.</p> <p>Enter the username for the DevOps Integration User account.</p>
Integration user password	<p>This field is available only when the option to configure webhook is selected.</p> <p>Enter the password for the DevOps Integration User account.</p>
Use MID Server	<p>Optional. Select #MID Server for an on-premises tool that is attached to a #MID Server. Application is automatically set to #DevOps and capability is set to REST.</p>

## 5. Select Order Now.

A request is created. When the request is approved:

- If connecting to an organization, the tool is created.
- If connecting to a project, the tool is created, connected, and project objects like plans, repositories, and pipelines are discovered.

When connecting to a project, for the discovered objects, you can import historical data from the tool and also associate an application with it.

**6.** From the DevOps catalog items, select **DevOps App Onboarding**.

**7.** Select **Try It**.

**8.** In the DevOps App Onboarding form, enter the details:

Are you creating a new app or adding to an existing one?	Select from the options whether to create a new app or use an existing app.
App	Enter the name for the app that you're creating or using.
Onboarding pipelines	Enter the connected Azure DevOps tool name.
Pipelines	Select the pipelines for which you want to import historical data.
Artifact repositories	Select the artifacts for which you want to import historical data.
Onboarding repositories	Enter the connected Azure DevOps tool name.
Import from and Import to	Select the dates for which you want to import the pipeline and artifact data. By default, the last 30 days are selected. You can choose to import data for a maximum of 90 days.
Repositories	Select the repositories for which you want to import historical data.
Import from and Import to	Select the dates for which you want to import the Repositories data. By default, the last 30 days are selected. You can choose to import data for a maximum of 90 days.
Onboarding plans	Enter the connected Azure DevOps tool name.
Plans	Select the plans for which you want to import historical data.
Import from and Import to	Select the dates for which you want to import the Plans data. By default, the last 30 days are selected. You can choose to import data for a maximum of 90 days.

**9.** Select **Order Now**.

A request is created. When the request is approved, the plans, repositories, artifacts, and pipeline objects are associated to the app record and webhooks are configured for real-time tracking. Historical data is imported for the selected items. The **Track** field is automatically enabled for imported plans, repositories, and pipelines. For repositories the **Track file changes** is also automatically enabled.

### Onboard Azure DevOps to DevOps Change Velocity — Classic

Create an Azure DevOps tool record in DevOps Change Velocity to connect, discover, and import Azure DevOps tool data.

**Before you begin**

Role required: sn\_devops.admin or sn\_devops.tool\_owner

**About this task**

Actions:

- **Connect** to Azure DevOps organization or project.
- **Discover** plans, repositories, orchestration tasks, and pipelines.
- **Configure** webhooks in Azure DevOps.
- **Import** work item, plan version, and feature records, branch and commit records, and task execution and step execution records.

To customize Azure Boards import of work item states or types, use the `DevOpsAzureDevOpsWorkItemHelper` script include.

**Procedure**

1. Navigate to **DevOps > Tools > Create New**.
2. On the Create DevOps tool form, fill in the fields.

Field	Description
Tool name	Name of your choice to identify this tool.
Tool Integration	Select <b>Azure DevOps</b> .
Connect to an organization or project	Select from the list. <ul style="list-style-type: none"> <li>◦ <b>Connect an organization:</b> Connect directly at your Azure DevOps organization level. All the projects within the organization can be discovered, and you can choose to configure multiple projects within the organization.</li> <li>◦ <b>Connect a project:</b> Connect directly at the project level.</li> </ul>
Tool URL	Azure DevOps organization URL (for example, <code>https://dev.azure.com/&lt;your organization&gt;</code> or the Azure DevOps project URL (for example, <code>https://dev.azure.com/&lt;your organization&gt;/&lt;your project&gt;</code> , depending on your earlier selection).
Tool Username	Azure DevOps instance username.
Tool password/Access Token	<a href="#">Personal access token (PAT)</a> for your Azure DevOps instance.

Field	Description
	<p><b>i Note:</b> Only a personal access token is supported.</p> <p>When you generate a Personal access token (PAT) for Azure DevOps, you must select the scopes to authorize if you aren't granting complete access. See <a href="#">Azure DevOps PAT scopes for DevOps</a>.</p>

3. Optional: If your Azure DevOps instance is attached to a MID Server, select the **Use MID Server** option and enter its details.

(Optional) For more information about MID server, see [MID Server selection](#)

4. Select **Submit**.

The tool is automatically connected using a connection alias and HTTP tool connection (Basic Auth credential) and the tool record details are shown in a form.

**i Note:** If you do not have global admin privileges for your tool (to allow automatic configuration of the webhook URL), you might need to have the tool admin user configure it for you (cut and paste the webhook URL into the tool configuration). Once the webhook is configured in the tool, **Enter Manual Configuration Mode** to connect to the tool manually, then exit.

On successful tool creation, you're taken to the tool record page.

5. Optional: If you want to control access to the tool, add the groups that must be given access to the tool in the **Maintained by** field on the **Access** tab.

The tasks the users in the groups can perform depends on the roles assigned to them.

- DevOps Tool Owner role: Can view and edit the tool.
- DevOps App Owner role: Can view the tool and can associate, discover, import historical data, and modify pipeline steps (if applicable) of the tool's objects (such as plans, repositories, and pipelines).
- DevOps Administrator role: Can edit all tools.
- Other DevOps roles: Can view the tool.

**i Note:** Only groups containing users with DevOps roles are available for selection in the **Maintained by** field.

The **All App Owners can view and associate tool objects to applications** option becomes available for selection if you choose to restrict access to the tool. This option enables all users having the DevOps App Owner role to access the tool. If selected, they'll be able to view, associate, discover, import historical data, and modify pipeline steps (if applicable) of the tool's objects.

6. Discover tool objects, including existing application plans (projects associated with the tool), repositories, orchestration tasks, and pipelines.

- Connected to organization: Select **Discover projects** to discover the projects from the organization. After discovery, the projects from your organization are available under the **Projects** related list.
- Connected to a project: The project is discovered during connection creation and is available under the **Projects** related list. Selecting **Discover projects** from the **Projects** tab of the tool record page discovers all the projects in your organization.

Select a project from the **Projects** related list to navigate to the project record page. Select **Discover** to discover the project objects, including existing plans (boards), repositories, and pipelines.

## 7. Configure webhook URLs in Azure DevOps.

- You can configure directly from the tool records page, or from the project record page for each product.
  - To configure from the tool records page, select **Configure projects**. Select the projects that you want to configure and select **Continue..**
  - To configure from the project records page, first select the project from the tool records page and select **Configure**.
- Enter the DevOps integration user and password and select **Send**.

On send, webhooks and service connections are automatically created in Azure DevOps. This sends the notifications from Azure DevOps tools to DevOps Change Velocity using the DevOps Integration user. For information on setting up Integration user account, see [Set up system accounts in DevOps Change Velocity](#).

**Note:** For plans, the **Track** field is automatically set to **True**.

## 8. For the discovered plans, you can import historical data from the tool and also associate an application with it.

- Open a plan record from the Plans related list.
- Select the **Import** related link.  
Records are created for the plan in the Work Items, Plan Versions, and Features related lists.
  - Work items related list shows tasks, bugs, and stories.

**Native State** and **Native Type** fields of the work item contain the original state and type values from the source tool.

- Plan Versions related list shows releases.
- Features related list shows epics and features.

**Note:** Historical import of Azure DevOps work items isn't supported for Agile Boards CMMI process.

- In the Apps related list, select **Edit...** to select an App to associate with the plan (project), or select **New** to create one.  
A plan must have an associated App record to complete the planning tool setup.

9. For the discovered repositories, import historical data for the tool, and associate the repository with an app.
  - a. Open the repository record from the Repositories related list.
  - b. Select **Import**.  
Imported branch records and commit records from the repository are added to the corresponding related lists.
  - c. In the **App** field, select the lookup list and select an App record to associate with the repository, or click **New** to create one.  
Imported historical data records are added to the corresponding related lists.

### What to do next

For Azure pipelines, [Model an Azure pipeline in DevOps](#) to complete the configuration in DevOps Change Velocity.

- Map each pipeline to a specific app.
- Create pipeline steps and map each step to an Azure pipeline job.
- Configure change control.

### Model an Azure pipeline in DevOps

Model an Azure pipeline by mapping the pipeline to an app, and mapping DevOps Change Velocity pipeline steps to Azure pipeline jobs.

#### Before you begin

Role required: sn\_devops.admin

#### Procedure

1. Map your pipeline to an app in DevOps.
  - a. Navigate to **DevOps > Apps & Pipelines > Apps** and open the application record to associate with the pipeline.
  - b. In the Pipelines related list, click **Edit...** to select a pipeline to associate with the app, or click **New** to create the pipeline.

**i Note:** While associating a pipeline with an app, the pipeline steps are also fetched during import.

For a new pipeline, fill in the **Orchestration pipeline** field using the project name and pipeline name as specified in Azure DevOps Pipelines in path format.

For example, My Project/My Classic Build Pipeline.

**i Note:** The project name must be specified with the pipeline because there could be multiple pipelines with the same name in different projects.

- c. Click **Submit**.
2. Open the pipeline record again and select the **Track** check box so events from the pipeline are received.

The **Track** check box must be selected to integrate the pipeline with DevOps Change Velocity.

3. Create DevOps steps automatically or manually to map to each Azure pipeline job so an orchestration task can be created.

**i Note:** Manual creation of steps is not required when you [Use the ServiceNow DevOps extension for Azure DevOps](#).

- Automatically create and map pipeline steps in DevOps by running your Azure pipeline when you [Use the ServiceNow DevOps extension for Azure DevOps](#).

Pipeline steps are automatically created, mapped, and associated when DevOps receives step notifications from your Azure pipeline during the run.

- Manually create and map each pipeline step to an Azure pipeline job.

In the Steps related list, click **New** to create a DevOps step for each Azure pipeline job (**Orchestration stage** field).

**i Note:** The **Orchestration stage** field value of each step is case-sensitive and must match the original name of the corresponding Azure pipeline job.

Name	Name of the pipeline step.
Pipeline	Pipeline in which the step is configured.
Type	<p>Pipeline step type.</p> <ul style="list-style-type: none"> <li>▪ Build and Test</li> <li>▪ Test</li> <li>▪ Deploy</li> <li>▪ Deploy and Test</li> <li>▪ Manual</li> <li>▪ Prod Deploy</li> </ul>
Order	<p>Order in which the steps are run.</p> <p><b>i Note:</b> The step order determines the order of the cards in the <a href="#">Pipeline UI</a>. The order of the cards in the Pipeline UI is by task execution.</p>
Orchestration stage	<p>Azure pipeline job name (case-sensitive).</p> <p><b>i Note:</b> For step association with Azure pipeline jobs, the <b>Orchestration stage</b> field must be configured.</p>

Business service	Configuration service that applies to the step.
------------------	---

Once orchestration tasks are created, [associate](#) each orchestration task in the Orchestration Tasks related list with a DevOps pipeline step.

#### 4. Enable change control automatically or manually.

- If you are using the ServiceNow DevOps extension for Azure DevOps, run your Azure DevOps pipeline to automatically enable change control.
- If you are configuring change control manually, select the **Change control** check box in a step to enable [change acceleration](#) and the corresponding configuration fields. For Azure release (CD) pipelines, enable change control in the first step of the required stage only. Change control for Azure release (CD) pipelines is supported only in pre-deployment gates.

**i Note:** ServiceNow Change Management  must be installed for change acceleration.

Change receipt	Select to enable change receipt for the step so the pipeline doesn't pause when a change request is created.  All pipeline data is included in the change, but approval is not required for the pipeline to proceed.
Change approval group	Approval group for the change request.  The change approval group becomes the <b>Assignment group</b> in the DevOps change request.  <b>i Note:</b> Ensure that the selected group has members and a group manager so the approver field is not empty.
Change type	Change request type to create. <ul style="list-style-type: none"> <li>▪ Normal (default)</li> <li>▪ Standard</li> <li>▪ Emergency</li> </ul>
Template	List of templates to use to auto populate fields for Normal or Emergency change requests.  Select a template or create a new one.
Standard change template	List of standard change templates to use for Standard change requests.

**Note:** This field is shown only when **Change type** is Standard.

**Note:** This field is required for Standard change type.

Change controlled branches

(Multibranch only) Comma-separated list of branches under change control. Wildcards are supported.

You can set up change control in your Azure DevOps YAML or Classic Azure pipeline using the [Azure Invoke REST API](#) or the [ServiceNow DevOps extension for Azure DevOps](#).

## What to do next

Configure the Azure pipeline for DevOps

### Associate Azure pipeline steps in DevOps

For manually created DevOps steps, associate each orchestration task in the Orchestration Tasks related list with a DevOps pipeline step to track the activity of each stage in your Azure pipeline.

### Before you begin

Role required: sn\_devops.admin

### Procedure

1. Navigate to **All > DevOps > Tools > Orchestration Tools**, and open the Azure DevOps tool record.
2. In the Orchestration Tasks related list, enter the corresponding DevOps pipeline step in the **Step** field of each orchestration task.

**Note:** The **Track** field is set to **True** by default when you discover orchestration tasks and pipelines. Tracking is required to receive job notifications from Azure DevOps.

Azure pipeline job run notifications are sent to the DevOps application. Each task execution notification corresponds to an orchestration task and, because orchestration tasks are mapped to a certain step in your DevOps pipeline, you can track the activity in each stage of your pipeline.

### Example:

[DevOps associate steps](#)

### Configuring the Azure pipeline for DevOps

Change control, artifacts, and packages can be configured in the Azure pipeline for integration with DevOps.

You can configure change control in Azure pipelines two ways.

- Use the [Azure Invoke REST API](#).
- Use the [ServiceNow DevOps extension for Azure DevOps](#).

The **ServiceNow DevOps** extension for Azure DevOps includes:

- ServiceNow DevOps service connection
- ServiceNow DevOps Release Gate
- Azure build (CI) pipeline agent and server job custom tasks

### Configuring change control using the Azure Invoke REST API

You can use the Azure Invoke REST API in your YAML or Classic Azure pipeline to configure change control for DevOps.

For Azure Invoke REST API details, visit the [Microsoft documentation site](#) and search for [Invoke HTTP REST API task- Azure Pipelines](#).

#### **Important:**

If you have duplicate or reused job names in your pipeline execution steps, ensure that the stageName attribute contains azureStageName/jobName in its value, i.e. `stageName = azureStageName/jobName`. The artifact registration tasks send both stage and job names to associate the artifact version to the correct task execution.

### Generic service connection

Using the Azure Invoke REST API requires the creation of a generic service connection in Azure DevOps.

### YAML Azure pipeline

In Azure DevOps, a server task must be created with the service connection as the change control endpoint.

#### Invoke REST API payload requirements

Azure pipeline type	Values
Build	<ul style="list-style-type: none"> <li>• buildNumber</li> <li>• isMultiBranch</li> <li>• branchName</li> </ul>
Release	<ul style="list-style-type: none"> <li>• releaseNumber</li> <li>• projectName</li> </ul>

**Note:** For release pipelines, set the **Pre-deployment conditions > Advanced > Completion event** field to Callback.

Build pipeline:

```
- task: InvokeRESTAPI@1
  inputs:
    connectionType: 'connectedServiceName'
    serviceConnection: 'change1'
    method: 'POST'
```

```

body: |
{
  "buildNumber": "$(build.buildId)",
  "isMultiBranch": "true",
  "branchName": "$(build.sourceBranchName)"
}
waitForCompletion: 'true'

```

Release pipeline:

```

- task: InvokeRESTAPI@1
  inputs:
    connectionType: 'connectedServiceName'
    serviceConnection: 'change1'
    method: 'POST'
    body: |
    {
      "releaseNumber": "$(Release.ReleaseId)",
      " projectName": "$(System.TeamProject)"
    }
  waitForCompletion: 'true'

```

## Classic Azure pipeline

For a Classic Azure pipeline, an Invoke REST API server task must be added.

### Classic Azure build pipeline example

### Classic Azure release pipeline example

## Use the ServiceNow DevOps extension for Azure DevOps

Install and configure DevOps extension for Azure DevOps to send build and release notifications from your Azure pipeline to ServiceNow DevOps application.

### Before you begin

Role required: sn\_devops.admin

### About this task

You can use **ServiceNow DevOps** extension on [Visual Studio Marketplace](#) to integrate your Azure pipeline with the ServiceNow DevOps application.

- **ServiceNow DevOps** service connection

Required to connect the Azure pipeline to ServiceNow

- **ServiceNow DevOps Release Gate**

Required to enable change control in Azure release (CD) pipelines (in pre-deployment conditions only)

Azure build (CI) pipeline custom tasks:

- Agentless (server) job
  - **ServiceNow DevOps Server Change Acceleration** custom task  
Required for agentless (server) jobs to automatically create a change request in ServiceNow Change Management as part of the Azure pipeline.
  - i Note:** The **ServiceNow DevOps Server Change Acceleration** task does not require **ServiceNow DevOps Server Job Notification** tasks.
  - **ServiceNow DevOps Server Package Registration** custom task  
Required for agentless (server) jobs to register a package in the ServiceNow instance
  - **ServiceNow DevOps Server Artifact Registration** custom task  
Required for agentless (server) jobs to register an artifact in the ServiceNow instance
- Agent job
  - **ServiceNow DevOps Agent Package Registration** custom task  
Required for agent jobs to register a package in the ServiceNow instance
  - **ServiceNow DevOps Agent Artifact Registration** custom task  
Required for agent jobs to register an artifact in the ServiceNow instance
- Software Quality scans (SonarQube/SonarCloud)
  - **ServiceNow DevOps Build Sonar Registration** custom task (for Build pipelines)
  - **ServiceNow DevOps Release Sonar Registration** custom task (for Release pipelines)

## Procedure

1. Go to [Visual Studio Marketplace](#), search for the **ServiceNow DevOps** extension, and click **Get it free**.
2. In Azure DevOps Pipelines, navigate to the service connections section in your project settings and create a **New service connection** using the **ServiceNow DevOps** service connection.

ServiceNow URL

`https://<your-instance>.service-now.com/`

Tool ID

The sys\_id of the orchestration tool.

You can copy this value using the `Copy sys_id` command on the Orchestration Tool form. If you are using the workspace, select **More form options () > Copy sys\_id** on the ADO tool connection form. If you are using Service Catalog or Classic, select

**Additional actions () > Copy sys\_id** on the ADO tool connection form.

Username	devops.integration.user
Password	Password for DevOps Integration User.
Service connection name	DevOps Connection
Grant access permission to all pipelines	Select check box.

### Example:

#### Azure pipeline - ServiceNow DevOps service connection

### Set up an Azure build (CI) pipeline in DevOps

Use the ServiceNow DevOps extension for Azure DevOps to configure change control and artifacts and packages in your Azure build (CI) pipeline.

### Before you begin

Role required: sn\_devops.admin

### Procedure

- In Azure DevOps Pipelines, **Add (+)** the **ServiceNow DevOps Server Change Acceleration** custom task to the Tasks section of your Azure pipeline agentless (server) job to configure change acceleration.

For more information regarding change acceleration, see [change acceleration](#).

Display name	ServiceNow Change Acceleration
ServiceNow endpoint	My Connection
Upstream job executed	Indicates the previous job in line.  For example, the job before Server might be Test.
Change request details	Set <a href="#">closure code and change request fields</a> from within the pipeline.  Click the information icon to view sample input.

- In Azure DevOps Pipelines, **Add (+)** the ServiceNow DevOps server or agent artifact and package registration custom tasks to the Tasks section of your Azure pipeline server or agent job to configure artifacts and packages.

See [DevOps change acceleration for releases](#) for more information regarding artifacts.

### Example:

#### Azure pipeline: ServiceNow DevOps Change Acceleration custom task

## Change request details example

### Set up an Azure release (CD) pipeline in DevOps

Use the ServiceNow DevOps extension for Azure DevOps to configure change control, and artifacts and packages in your Azure release (CD) pipeline.

#### Before you begin

Role required: sn\_devops.admin

#### About this task

- Note:** Change control in Azure release (CD) pipelines is supported in pre-deployment gates only. Pre-deployment gate change requests are mapped to the step execution of the first job in that stage.

Phases in an Azure release pipeline are mapped to a step. The step name for a multi-config or multi-agent job must include the phase name only, and not the actual job name derived at pipeline execution run time.

Task executions for skipped jobs are marked as failed.

#### Procedure

1. In Azure DevOps Pipelines, navigate to your release pipeline and open the Pre-deployment conditions window.
2. Enable the Gates setting and click **+Add**.
3. Click the **ServiceNow DevOps Release Gate** and select the ServiceNow endpoint.
4. Exit the release gate configuration, and expand the Evaluation options section to configure the timing fields.

Time between re-evaluation of gates	<p>Re-evaluation interval in minutes, hours, or days.</p> <p>If the change request is canceled or rejected, the Azure pipeline release gate keeps re-evaluating the change request status at the configured interval until timeout.</p> <p>No user intervention is required for further attempts of re-evaluation after the change request has already been approved, rejected, or canceled.</p>
Timeout after which gates fail	<p>Timeout value in minutes, hours, or days.</p> <p>If the pre-deployment conditions fail, the subsequent jobs of the stage are marked as failed, and the start and end time defaults to the current system time.</p>

5. Optional: Configure artifacts in your Azure release (CD) pipeline.

To set up artifacts using the build pipeline as the source, these names must match.

- Repository name, and build pipeline name (for example, DeployableRepo).
- Name property of the build pipeline artifact, and source alias property of the release pipeline artifact (for example, BuildDrop).

In addition to the build pipeline, you can select artifacts from any other eight sources. To track commit & work item details, follow these rules.

- When the source is the build pipeline, the semantic version property of the artifacts should be in the format MAJOR.MINOR.PATCH (for example 5.1.3).
- When the source is not the build pipeline, define a semantic version by implementing the **DevOpsArtifactSemanticVersionAPI** extension interface.

See [DevOps change acceleration for releases](#) for more information regarding artifacts.

### Example:

#### ServiceNow DevOps extension for Azure DevOps - Release Gate

##### Azure release pipeline pre-deployment gate configuration

##### Artifact setup - build pipeline source

##### Artifact setup - release pipeline

#### Azure DevOps PAT scopes for DevOps

Scope access levels are required when using a personal access token (PAT) to access Azure DevOps during setup.

Scope access level settings are based on the capability you have configured. Set the corresponding access level for seamless functionality. For information on creating a PAT, see [Personal access token \(PAT\)](#).

**Important:** With the access level permissions specified in the following table in Azure DevOps, and ServiceNow DevOps extension, you can connect to Azure DevOps from ServiceNow. Your Azure DevOps admin does not have to manually configure webhooks and service connections in Azure DevOps.

**Important:**

- When onboarding a Project, the **Project Administrators** privilege requires the owner of the PAT to be a member of the project's **Project Administrators** group.
- When onboarding an Organization, the **Project Administrators** privilege requires the owner of the PAT to be a member of the organization's **Project Collection Administrators** group.

## Scope access level settings per capability and their impact

Capability	Scope	Access level	Impact
Boards	Work item	Read	Required to discover the boards and receive the work items either through import/polling or real time with a configured webhook.
Repos	Code	Read	Required to discover repositories and receive branches, commits, and tags either through import/polling or real time with a configured webhook.
Build pipelines	Build	Read & Execute	<ul style="list-style-type: none"> <li>• Read: Required to discover the build pipelines and receive pipeline execution details like stages, artifacts, test results, code security results, etc., either through import/polling or real time with a configured webhook.</li> <li>• Execute: Required to pause/resume the pipelines based on the change control step.</li> </ul>
Release pipelines and gates	Release	Read, write and execute	<ul style="list-style-type: none"> <li>• Read: Required to discover the release pipelines and receive pipeline execution details like stages, artifacts, test results, code security results, etc, either through import/polling or real time with a configured webhook.</li> <li>• Write and Execute: Required to pause/resume the pipelines based on change control step.</li> </ul>
Test build and release pipelines	Test management	Read	Required to receive test results for pipeline execution.
Service Connections	Service connection	Read, query, and manage	Required to create Service connection automatically which is used to configure ServiceNow tasks like change acceleration, artifact and package registration, etc.
Packaging	Packaging	Read	Required to discover the artifact repositories and receive the feeds and packages either through import/polling or real-time with a configured webhook.

## Limitation of Azure DevOps

If you create an Azure tool with custom defined access level, and you reconfigure such a tool because of change in your Integration user credentials, then the existing service hooks for release created and release deployment are not updated. Instead, two new service hooks are created with new configuration details. To avoid the duplication of these service hooks, you must create the tool with full access level.

### Configure SonarQube scans on Azure DevOps pipelines

Configure SonarQube or SonarCloud scans on MS Azure DevOps pipelines. Check Azure DevOps pipeline executions for SonarQube scans on every stage of the pipeline's execution

and fetch lists and details of scans from any stage to DevOps Change Velocity. Drill down on the Scan Details based on categories.

## Before you begin

Ensure that you meet the following prerequisites before you configure SonarQube scans on your Azure DevOps pipeline:

- You are using a compatible SonarQube version. See [DevOps Change Velocity integrations](#) for supported tool versions.
- Connect, configure the Azure DevOps tool and discover existing repositories, orchestration tasks, and pipelines.
- Create a SonarQube tool record. For more information, see [Create a SonarQube tool record](#).
- Install the SonarQube extension from the Visual Studio Marketplace and configure branch analysis to use the Azure Devops tasks in your build definitions to analyze your projects. For more information, see [SonarQube documentation](#).
- Install the following custom extension tasks on your Azure DevOps instance.
  - ServiceNow extension to integrate Azure Pipelines with DevOps Change Velocity. For more information, see [Use the ServiceNow DevOps extension for Azure DevOps](#)
  - ServiceNow DevOps Build Sonar Registration (for Build pipelines)
  - ServiceNow DevOps Release Sonar Registration task (for Release pipelines)

Role required:

- admin or sn\_devops.admin in DevOps Change Velocity
- admin in Azure DevOps
- admin role in SonarQube with access to all projects that the SonarQube scans are configured on.

## About this task

By default, Azure DevOps provides you with the following tasks to run a SonarQube scan on build and release pipelines:

- Prepare analysis on SonarCloud
- Run Code analysis
- Publish Quality Gate Result

Add and configure custom ServiceNow extension tasks in order to them default tasks, to fetch the scan details to ServiceNow DevOps from Azure DevOps build and release pipelines. The scan details that are fetched are:

- pipelineName
- buildNumber
- stageName
- branchName
- sonarProjectKey
- sonarInstanceUrl

## Procedure

1. Set up and configure the Azure DevOps and SonarQube Integration to fetch scan analysis from Build pipelines.
  - a. In the Azure DevOps console, navigate to **Organization > SonarIntegrations > Pipelines > Jobs**.
  - b. Click the **Add Tasks** icon (), and search for the ServiceNow extensions in the **Add tasks** search bar.
  - c. Add the following tasks.
    - ServiceNow DevOps Build Sonar Registration task.
    - ServiceNow DevOps Release Sonar Registration task.
  - d. On the form, fill in the fields.

### ServiceNow DevOps Build/ Release Sonar Registration

Field	Description
Display name	Auto-populates on entering key and URL.
ServiceNow endpoint	The ServiceNow instance endpoint that is automatically created during Tool configuration. Use the same service connection for this task.
Sonar project Key	Enter the same project key value which you used while configuring the <b>Prepare analysis on SonarCloud</b> task.
Sonar Instance Url	Enter the same URL which you used to connect to the Sonar tool during tool creation.
Job Name	<p>The name of the Azure DevOps pipeline job.</p> <p>This field appears only for the <b>ServiceNow DevOps Release Sonar Registration</b> task.</p>

**i Important:** In the list of tasks for the pipeline job, ensure that the **ServiceNow DevOps Build Sonar Registration** or **ServiceNow DevOps Release Sonar Registration** task is added after the **Run Code Analysis** task.

You have configured the build pipelines to send sonar scan results to your DevOps application. Scan results are mapped to the Software Quality capability are processed by an associated base system sub flow (*FetchSonarScanId*), once an event is created and processed in the Inbound Events table.

2. Associate Azure pipelines steps in DevOps.
3. Run the pipeline.

## Result

Based on the scan results on various stages of pipeline's execution, the results are shown in the corresponding step of the pipeline in ServiceNow DevOps. Inbound events are created for notifications and subflows are triggered based on the notification type and capability.

## What to do next

- View scan details as part of Task Executions. View details of all the Sonar scans that are part of the task execution mapped to a build or release pipeline execution step.

1. Navigate to **DevOps > Orchestrate > Task Execution** click a relevant Task Execution record.

2. Click the Software Quality Summary related list.

3. Click a relevant Scan ID record.

The Software Quality Scan Summary and Scan Details are displayed.

- View scan details as part of Change Request. View all the scans that were part of this build/release pipeline in the **Software Quality Results > Software Quality Summary** related list.

1. Navigate to **DevOps > Orchestrate > Pipeline Change Requests**

2. Click the Software Quality Summary related list.

3. Click a relevant Scan ID record.

The Software Quality Scan Summary and Scan Details are displayed.

## Restarting failed build or release pipeline jobs and stages

Rerun or redeploy Azure DevOps build, release changes, or pipelines that are failed or canceled in that stage or pipeline. The reattempts display on the DevOps pipeline UI as continuous runs instead of creating new executions.

## Rerun Azure DevOps pipelines or stages

You can rerun a failed or canceled build or release pipelines or change jobs in Azure DevOps. The reruns are processed as part of the same pipeline execution as the first run in ServiceNow DevOps. You can rerun entire pipelines or specific failed or canceled jobs and stages. You can now choose to reuse a change request instead of creating a new change request each time you restart a stage or a pipeline.

An *attemptNumber* parameter is added to the payload which helps us track reruns. Associated test summary, software quality scan results, commits, work items corresponding to every rerun attempt is also updated in ServiceNow DevOps.

If you are using the [Azure Invoke REST API](#) you must add the attempt number parameter to your payload body in the specified syntax format for build and release pipelines. If you do not specify the attempt number parameter, the default attempt number is set to 1.

Example attempt number parameter in build pipeline payload:

```
"attemptNumber": "${system.jobAttempt}"
```

Example attempt number parameter in the release pipeline payload:

```
"attemptNumber": "${Release.AttemptNumber}"
```

**1 Note:** Do not use the existing started and completed notifications for the stage jobs. If your jobs consider the started and completed notifications the rerun functionality does not work.

## Reusing Change Requests

If a change enabled job is rerun, and a change request exists for the previous run/attempt, you can choose to reuse the previous change request or create a new change request, using the base system 'DevOps Change Request Reusability Decision Subflow'. The default implementation of this subflow, allows you to reuse a change request from the previous attempt if the change request is in implement, or post-implement states. If the Change request is in any other state, by default, a new change request is created when you rerun the job. Per existing behavior, all associated details such as test summaries, and scans, are newly generated while commits and work items are retained unchanged for new change requests.

For example, when a pipeline fails at a specific stage after the change request is approved, and you rerun that stage. The change request is reused, the associated test summary and software quality scans, and the commits and work items associated to the artifact are associated with the same change request which you approved.

To apply a custom logic for reusability, you can copy the existing subflow, make the changes, publish it, and update the new subflow name under **DevOps Properties > DevOps Change Request Reusability Decision Subflow**.

In the regular base system flow when a change is created, '[DevOps Model Change Request flow](#)' is used to update the '#State# field of step execution record after a decision is taken on the change request. However, when you reuse a change, the first trigger condition of a change request being created is not met. A base system subflow 'DevOps Change Request Reusability Model Subflow' is triggered instead, whenever a change request is reused when a job is a rerun. The default implementation of this subflow is similar to the DevOps Model Change Request flow. You can create a custom subflow and update the subflow name at **DevOps Properties > DevOps Change Request Reusability Model Subflow**.

## Pipeline UI changes

ServiceNow DevOps synchronizes all the changes that are caused when you restart or rerun a stage or a job, and displays them in the DevOps pipeline UI.

- Click a card to view the latest attempt of that stage.
- Click the **View all attempts** link to see all the step executions and related information associated to the step or stage that is run more than once.
- The View change link displays the change request associated with the latest attempt.

In previous releases, failed jobs were either ignored or a new pipeline execution job was created for reruns and processed accordingly. For more information, see [DevOps Pipeline UI](#).

## Execution sequence and waiting logic for rerun jobs

Processing sequence and waiting logic for rerun jobs are different when you reuse or create a change request as part of a rerun job.

## Existing Considerations

- A change request should not exist in a stage which contains parallel jobs.
- If more than one stage is running in parallel, change request should not be the first job in both the stages.

**Note:** Parallel stages in release pipelines are processed and displayed in the pipeline UI as they occur in the Azure DevOps pipeline. Parallel stages in build pipelines are still processed in parallel but appear in a serial order in the pipeline UI.

## Upgrade Considerations

There is no change in the functionality or execution when you run the first pipeline attempt. All stages are processed sequentially and associated tests, software quality scans and change requests are executed and created as modeled.

### **Note:**

- Run a new pipeline after you upgrade if you have rerun stages and pipelines before upgrading. Rerun attempts and failed events prior to the upgrade are ignored by ServiceNow DevOps for reattempts.
- If you have only run the pipeline once before upgrade, you can rerun the stage or the pipeline. The rerun functionality applies as designed and is saved in ServiceNow DevOps.

## Execution Sequence and processing logic

- If the same artifact version registration call is received in reattempt, the registration call is ignored.
- Package registration calls with same package name are not ignored. A new package associated to artifact versions and pipeline execution is created during reattempt. The artifacts that are associated to the latest package, will be shown on the change request.

From the Azure DevOps GUI, if you rerun a stage in a build pipeline the subsequent stages reruns are also triggered in its specified sequence. If you reattempt processing a pipeline before all the stages of the previous attempt are not complete. The subsequent attempt waits until all the events in the previous attempt are processed.

For release pipelines, the stages are run in the specified sequence only during the first run. For subsequent rerun attempts manually run each stage. In release pipelines even if stages are running in parallel in Azure DevOps, from the second attempt onwards the events are processed in the specified sequence.

- When a new change request is created for a reattempt stage job, and the stage that you are reattempting includes a test and a software quality scan only the latest Test Summary and Software Quality scan results display on the change request related list.
- When a change request is reused for a rerun stage job, the Test Summary and Software Quality scan results for each attempt displays in the change request related list.

## Parallel stages in Azure DevOps release pipelines

Parallel stages in a release pipeline are now processed concurrently and displayed in the DevOps pipeline UI in real time. Base system pre-deployment conditions and release gates enable you to create change requests that include details from parallel stages.

## Base system parallel stage support for Azure DevOps

Organizations use parallel stages to automate and speed up release processes for tasks that can be performed in parallel. For example, a release pipeline has integrated multiple test and software quality tools and has jobs configured to run in parallel. Not running each job sequentially significantly speeds up the release pipeline execution.

ServiceNow DevOps supports processing parallel stages in release pipelines and displays the stages in a parallel view in the DevOps pipeline UI. Effectively, the DevOps pipeline UI replicates the Azure DevOps GUI in real time.

You can also see the details of the processed stages in the pipeline UI.

**Important:** Support for parallel stages is restricted to Release pipelines. Build pipelines continue to appear in a sequential or serial manner in the DevOps pipeline UI, even if parallel stages are configured for build pipelines in Azure DevOps.

## ServiceNow® DevOps release gate in pre-deployment conditions to create Change requests

A base-system ServiceNow DevOps Release Gate is added along with pre-deployment conditions. Enable base system deployment gates that are configured to call the Now Platform instance, to create a change request before the deploy to production stage. Change requests are now created after all previous (upstream) stages complete processing. The change request captures relevant details from all upstream stages and displays them in the following corresponding related lists.

- Commits
- Work Items
- Test Summaries
- Software Quality Summary
- Artifact Versions

After the pipeline execution completes processing the parallel stages preceding the production deployment stage, a change request is automatically created and mapped to the deploy to production stage in the Pipeline Executions view. The production stage completes processing once the change request is approved.

From the **Pipeline Execution** view of the relevant pipeline, click the **Pipeline UI** related link to view the real-time state of the pipeline as it appears in Azure DevOps. The associated artifact details which are sourced from the build pipeline, Test Results, Software Quality Summary Results display on the pipeline UI.

## Upgrade Considerations

Ensure that you review the following considerations before upgrading.

**Important:** A change request should not exist in a stage which contains parallel jobs.

- The **Upstream execution** column in the Task Executions table is not displayed for fresh installations. Any customizations that you have made using the **Upstream execution** column prior to the upgrade are unaffected.
- If stages are running in parallel, a change request should not be the first job in any stage.
- After upgrading, new release pipeline executions process parallel stages concurrently and display parallel stages and associated details in the pipeline UI. Azure DevOps release pipelines that are already executed and stored in ServiceNow DevOps prior to the upgrade remain unaffected and continue to display parallel stages (that are already executed and persisted) in ServiceNow DevOps serially.
- If the pre-deployment ServiceNow DevOps release gate is enabled in more than one start stage in a release pipeline with multiple start stages, it might result in multiple pipeline executions.

**Note:** A package is created for each start stage but any one package is associated per pipeline execution.

## DevOps work item import for Azure Boards

Azure Boards work items are mapped to default ServiceNow DevOps states and types during import. You can use the DevOpsAzureDevOpsWorkItemHelper script include to customize the mappings.

## Default Azure Boards work item mapping

**Native State** and **Native Type** fields of the work item contain the original state and type values from the source tool.

### Work item type mapping

ServiceNow DevOps	Azure Boards Basic	Azure Boards Agile	Azure Boards Scrum
Task	Task	Task	Task
		Test case	Impediment
			Test case
Bug	Issue	Bug	Bug
		Issue	
Story	--	User story	Product backlog item
Epic	Epic	Epic	Epic
Feature	--	Feature	Feature

**Note:** Historical import of Azure DevOps work items is not supported for Agile Boards CMMI process.

### Work item state mapping

ServiceNow DevOps	Azure Boards Basic	Azure Boards Agile	Azure Boards Scrum
Planned	To Do	New	New
			Open
			Approved
			Committed
			To do
WIP	Doing	Active	In Progress
		Design	Design
Complete	Done	Ready	Done
		Closed	Ready

## Work item state mapping (continued)

ServiceNow DevOps	Azure Boards Basic	Azure Boards Agile	Azure Boards Scrum
			Closed
Deleted	Deleted	Completed Deleted	Removed

**Note:** When an imported Azure Boards work item type or state is not recognized, the value is set to **Other**.

## Customize Azure Boards state and type mappings

Access the DevOpsAzureDevOpsWorkItemHelper script include in the **System Definition > Script Includes** module.

This script example adds new states and types for custom processes MyScrum and CustomBasic. CustomBasic inherits the state and type defined for Basic process.

```
var DevOpsAzureDevOpsWorkItemHelper = Class.create();

DevOpsAzureDevOpsWorkItemHelper.prototype =
Object.extendsObject(DevOpsAzureDevOpsWorkItemHelperSNC, {

setDefaultProcess: function (projectProcess) {

    DevOpsAzureDevOpsWorkItemHelperSNC.prototype.setDefaultProcess.call(this, projectProcess);

    //set custom states and types
    var newStates, newWITypes;
    if (projectProcess == 'NPScrum'){
        // no parent process set. So type and states avaibale will be limited
        to newStates
        // and newWITypes
        newStates = {
            'Delayed': 'planned',
            'Approved': 'wip'
        };
        newWITypes= {
            'Request': 'story',
            'Incident': 'task'
        };
    } else if (projectProcess == 'CustomBasic'){
        //set parent process to Basic to inherit basic states and types
        this.setParentProcess('Basic');
        newStates = {
            'Auto-Approved': 'wip'
        };
        newWITypes= {
            'UserStory': 'story'
        };
    }
}}
```

```

    }

    this.setStates(newStates);
    this.setWorkItemTypes(newWITypes);
} ,

type: 'DevOpsAzureDevOpsWorkItemHelper'
);

```

## Track specific pipelines in Azure DevOps

Enable and configure specific pipelines in Azure DevOps that you want to track. Select the pipelines you want to monitor and receive job notifications from Azure DevOps.

### Before you begin

- Connect DevOps to Azure DevOps tools.
- Ensure that you have modeled, configured, and associated Azure pipelines to DevOps.
- Review how to configure form layouts. For more information, see [Configuring the form layout](#)

Role required: admin, personalize\_form

### About this task

By default, the **Track Specific Pipeline** field is set to **False** when you discover Azure DevOps orchestration tasks and pipelines. The **Track** flag for all Azure DevOps pipelines is also disabled. So, by default, this leads to all pipelines sending job notifications to DevOps and increased processing times, which could cause the pipeline you really want to track to be delayed. So, enable the **Track Specific Pipeline** check box to process only notifications for the pipeline events that you specify to track, and ignore all other pipeline notifications.

### Procedure

1. Navigate to **All > DevOps > Tools > Orchestration Tools**, and open the Azure DevOps tool record.
2. Click the **Additional Actions** icon (), and select **Configure > Form Layout**.
3. Move the **Track Specific Pipeline** field to the selected list.
4. Click **Save**.  
You are redirected back to the DevOps Tool > Azure DevOps form, and the **Track Specific Pipeline** check box appears.
5. Select the **Track Specific Pipeline** check box.
6. Navigate to the Pipelines related list.
7. Click the **Update Personalized List** icon ()�.
8. In the Personalize List Columns form, move the **Track** field to the Selected list.  
Track Specific pipeline
9. Click **OK**.  
You are redirected back to the Pipelines related list, where the **Track** field now appears.
10. Navigate to the pipeline step which you want to track, double-click the **Track** field.
11. Change the **Track** field's value to **true**, and click **Save**.

## Result

Only the pipelines that have **Track** field set to **true** in the Pipelines related list are now being tracked and sent to the Inbound Events table [sn\_devops\_inbound\_event\_list.do]. All pipelines that have the **Track** field set to **false** are ignored.

### **i Note:**

- If the **Track Specific Pipeline** check box is selected only those pipelines that have the **Track** field value set to **true** are tracked and sent to the Inbound Events table. If **Track** field is set to **false** for all pipelines none of the pipelines are tracked.
- If the **Track Specific Pipeline** check box is unselected, all events of the pipeline are tracked and sent to the Inbound Events table.

## Managing Azure DevOps Artifacts

ServiceNow DevOps makes it easy to manage artifacts published through Microsoft Azure build pipelines, both build and Azure artifacts. You can import and track these artifacts for a seamless package management.

## Build Artifacts

Build and pipeline artifacts published using the Publish Build Artifacts (PublishBuildArtifacts) and Publish Pipeline Artifacts (PublishPipelineArtifacts) tasks respectively on the Azure DevOps are automatically created in your ServiceNow DevOps instance.

You can also import historical data for build artifacts using App Onboarding in Service Catalog.

Build artifacts created using the Publish Build Artifacts task will be part of the artifact repository whose name will be defaulted to the build pipeline name.

**i Note:** If two artifacts with the same name, belonging to the same repository, and having different versions are used as release triggers for a release pipeline, then the related lists (like work items, commits, test summaries, etc) will include DevOps data related to both the artifact versions when a change exists in the release pipeline. For example, if artifact-1.0 and artifact-2.0 are release triggers to manage rollback scenarios in deployment, then change (if part of release pipeline) will include related lists data for both the versions.

You can also continue to leverage the existing Extension-based artifact registration and import. For more information, see [Use the ServiceNow DevOps extension for Azure DevOps](#) and [DevOps change acceleration for releases](#).

**i Important:** Using Extension-based artifact registration is not required if the PublishBuildArtifacts and PublishPipelineArtifacts tasks mentioned above are used.

## Azure DevOps Artifacts

You can import and track the artifacts to your ServiceNow DevOps instance. You can import Azure DevOps artifacts that are published to the **Azure DevOps Artifacts** using the Universal Package task only.

**i Note:** Only Universal packages are supported currently for importing and tracking.

(DevOps 1.35 and later versions) The **Feed connection alias** field is available on the Azure DevOps Tools form. This field is auto-populated with an alias for the feed connection. For more information about connecting a DevOps tool, see [Create an artifact tool record in DevOps](#).

After upgrade, to select and add the artifact repositories using the App Onboarding, you must click **Discover** to discover existing artifact repositories. The records are added to the Artifact Repositories related list.

For tracking the artifact repositories, change the value of the **Track** column to True (default value, False).

When an Azure Artifact is published via the Universal Package Task, the artifact information is associated with a build pipeline within 24 hours using the *ADO Artifacts Daily*scheduled job.

### **i Note:**

- As Azure DevOps Artifact creation is not dependent on Build Pipeline completion. If Azure DevOps Artifact is used as Trigger for a Release Pipeline, it would be triggered even if Build pipeline is pending completion due to a change request.
- The scheduled job is executed every 24 hours by default and can also be executed on-demand. However, as the scheduled job is performance intensive, it should not be executed frequently.

Following points must be considered while importing Azure DevOps artifacts:

1. Artifacts cannot be published from the Release Pipelines due to Azure DevOps limitations.
2. Artifact Repositories are tracked as follows:
  - Historical Import of Azure DevOps artifacts is accomplished using the App Onboarding.
  - For real-time tracking of the artifacts, the value for the **Track** column must be changed to True (default value, False). Otherwise, the Feed or Artifact Repository won't be tracked.
3. The application supports the project-specific feeds but not the Organization specific feeds. Azure DevOps artifacts when published through Azure DevOps pipeline are only supported.
4. Azure DevOps artifacts not linked to any build or release pipeline (orphan artifacts) are not supported.
5. Azure DevOps Artifacts are stored in the following format:
  - Azure DevOps artifact linked to only Build Pipeline (or) Build and Release Pipeline:  
 <artifact-name>-<1.build-number.0> (Build Number of Build Pipeline)
  - Azure DevOps artifact linked only to Release Pipeline:  
 <artifact-name>-<1.build-number.0> (Build Number of Release Pipeline)

## Azure DevOps Artifact mapping with ServiceNow DevOps

The following table explains the mapping of Azure DevOps artifacts with ServiceNow DevOps fields:

Azure DevOps <b>Artifact</b>	ServiceNow DevOps
Feeds	Artifact Repositories
Packages	Artifacts
Provenance/build number	Versions

## Limitations

- Azure DevOps Artifact Project with 4000 Feeds (or less) are supported.
- Azure DevOps artifact Feed with 800 packages or artifacts (or less) is supported.
- Azure DevOps pipeline with 200 packages or artifacts (or less) publishing is supported.

## GitHub integration with DevOps Change Velocity

Connect your GitHub instance to discover repositories and pipeline definitions and configure real-time notifications or polling to enable change traceability and automation.

### Overview

DevOps Change Velocity supports both Code (Repository) and Orchestration (Actions) capabilities for the GitHub tool.

Both GitHub and GitHub Enterprise are supported and you can connect using one of the following authentication:

- Basic authentication
- [OAuth 2.0 credentials](#)

The following operations are performed as part of integrating GitHub:

- Connect: Discover repositories and pipeline definitions by connecting your GitHub instances to DevOps Change Velocity.
- Configure: Enable sending real-time notifications for commits and pipelines by automatically creating a Webhook (push and workflow\_job) in GitHub so that this data can be used to create change policies.

Custom actions by ServiceNow are available in GitHub Marketplace for the orchestration capability, to push information from Actions (workflows) and to pause or resume workflows from DevOps Change Velocity. For more information on custom actions, see [Additional information - GitHub Actions](#).

To capture the workflow data in DevOps Change Velocity, you need to configure Secrets in your GitHub tool. For more information, see [Additional information - GitHub Actions](#).

There are a few limitations for GitHub Actions support, see [Additional information - GitHub Actions](#).

## Get started

Use one of the following options to onboard GitHub. For a guided experience, use the workspace to onboard a tool. Alternatively, you can use the Service Catalog or Classic experience.

### Onboard GitHub to DevOps Change Velocity — Workspace

Connect your GitHub instance to discover, configure, and import repositories and pipelines.

#### Before you begin

Complete the tasks in [Getting started with DevOps Change Velocity](#).

Role required: sn\_devops.admin or sn\_devops.tool\_owner

## Procedure

1. Navigate to **Workspaces > DevOps Change Workspace** and use one of the following options to open the Playbook to onboard GitHub.

Option	Steps
Homepage	<p>a. Select the <b>Connect tool</b> widget</p> <p>b. From the Connect to a tool modal, select the tool from the appropriate category. For example, if you want to connect to GitHub as coding tool, select GitHub under the <b>Code</b> category.</p>
Applications module	<p>a. Select <b>Applications ()</b>.</p> <p>b. Select an existing application, or create one. To create an application, see <a href="#">Create an application in DevOps Change Velocity</a>.</p> <p>c. From the# Recommended actions# pane, select the#<b>Connect a tool</b> card.</p> <p>d. From the Connect to a tool modal, select the tool from the appropriate category. For example, if you want to connect to GitHub as coding tool, select GitHub under the <b>Code</b> category.</p>
Tools module	<p>a. Select <b>Tools ()</b>.</p> <p>b. From the Tools list, select the appropriate category. For example, if you want to connect to GitHub as coding tool, you'd select the <b>Coding tools</b> category.</p> <p>c. Select <b>Connect coding tool</b>.</p> <p>d. On the# Connect to a tool# modal, select GitHub.</p>

2. In the **Tool name** field, enter a name for the tool.

3. Select **Next**.

The DevOps playbook opens to help you complete the onboarding tasks.

4. Complete the connection and configuration using the playbook.

- a. In the **Credential type** field, select one.

- Basic Auth
- OAuth 2.0

- b. Enter the credentials.

- Basic Auth: Enter the username and password/access token of your GitHub instance
- OAuth 2.0: Enter your OAuth credential.

If the OAuth credential is created using GitHub Apps, the **GitHub app slug name** field is enabled. Enter a value for this field to check the permission requirements of the tool before connecting. You can find the GitHub app slug name on the settings page of your GitHub app.

For more information on OAuth2.0 credentials, see [Setting up GitHub OAuth 2.0 credentials for DevOps Change Velocity](#).

- c. Optional: If your GitHub instance is attached to a MID Server, select the **MID Server** option and enter its details.

(Optional) For more information about MID server, see [MID Server selection](#)

- d. Select **Connect**.

Permission check is run and the permission requirements are displayed in a pop-up modal.

For OAuth 2.0, if you haven't entered the GitHub app slug name, the tool is connected without checking permission requirements.

- e. Select **Continue** or **Connect Anyway**.

After successful connection, the repositories and pipelines are automatically discovered.

- f. Specify the access for the tool.

- i. If you want to control access to the tool, add the groups that must be given access to the tool in the **Maintained by** field.

The tasks these users in the groups can perform depends on the role assigned to them.

- DevOps Tool Owner role: Can view and edit the tool.
- DevOps App Owner role: Can view the tool and can associate, discover, import historical data, and modify pipeline steps (if applicable) of the tool's objects (such as plans, repositories, and pipelines).
- DevOps Administrator role: Can edit all tools.
- Other DevOps roles: Can view the tool.

**Note:** If you don't select a group and skip this step, all users with the DevOps Tool Owner role will be able to edit the tool.

- ii. If you choose to control access to the tool, the **All App Owners can view and associate tool objects to applications** option becomes available for selection.

This option enables all users having the DevOps App Owner role to access the tool. If selected, they'll be able to view, associate, discover, import historical data, and modify pipeline steps (if applicable) of the tool's objects.

**iii. Select **Assign**.**

**g. From the list, select the items you want to configure and select **Configure**.**

This action configures the following webhooks:

- *push*: To gather repository commits, branches, and tags
- *workflow\_job*: To gather pipeline data

**i Note:** Completing the configuration as part of this task is recommended as real-time notifications are ideal to maintain current information, particularly for automating change requests. Otherwise, you can set up the webhook by configuring it manually later by enabling nightly polling to fetch data system for any tracked repositories or pipelines by setting the *Enable Polling* property to **Yes**.

**h. From the Select repositories to track and Select pipelines activities in the playbook, select the repositories and pipeline that you want to track.**

For each selected pipeline, all steps are imported for the last successful execution.

**i. Optional: In the Assign services to pipeline steps activity, specify **Step type** and **Service** for each pipeline step.**

(Optional) Completing this step as part of this onboarding task enables the DevOps Insights dashboards to show more meaningful data immediately.

**j. Select **Next**.**

**5. From the **Summary** page, select **View tool record** to review the details of the connected GitHub tool.**

For GitHub Actions pipelines, you must perform some additional steps like creating secrets, defining the workflow configuration in GitHub, and so on. For more information, see [Additional information - GitHub Actions](#).

### **Result**

You've successfully onboarded your GitHub tool to DevOps Change Velocity.

### **Onboard GitHub to DevOps Change Velocity — Service Catalog**

Create, connect, discover, and configure your GitHub instance using the ServiceNow Service Catalog.

### **Before you begin**

Role required: sn\_devops.admin or sn\_devops.tool\_owner

## Procedure

1. Navigate to **All > Service Catalog > Catalog Definitions > Maintain Items** and search for DevOps.

**Note:** You can also access the service catalog from Employee Center or Service portal.

2. From the DevOps catalog items, select and activate **DevOps App Onboarding** and **DevOps Tool Onboarding**.
3. After activating, select **DevOps Tool Onboarding** and select **Try it**.
4. In the DevOps Tool Onboarding form, enter the tool details:

Field	Description
Tool name	Name for your GitHub integration.
Tool integration	Select GitHub or GitHub Enterprise.
Tool URL	Enter <a href="https://api.github.com">https://api.github.com</a> .
Credential type	Select Basic Auth or OAuth, depending on the credentials you're using. <ul style="list-style-type: none"> <li>◦ For Basic Auth, enter the username and password or access token.</li> <li>◦ For OAuth, enter the credentials.</li> </ul>
Use MidServer	Optional. Select #MID Server# for an on-premises tool that is attached to a #MID Server. Application is automatically set to #DevOps and capability is set to REST.

5. Select **Order Now**.

A request is created. When the request is approved, the tool is created, connected, and discovered.

6. From the DevOps catalog items, select **DevOps App Onboarding**.

7. In the DevOps App Onboarding form, enter the details:

Are you creating a new app or adding to an existing one?	Select from the options whether to create a new app or use an existing app.
App	Enter the name for the app that you're creating or using.
Onboarding pipelines	Leave empty.
Onboarding repositories	Enter the connected GitHub tool name.
Repositories	Select the repositories for which you want to configure webhooks and import historical data.
Import from and Import to	Select the dates for which you want to import the data. By default, the last 30 days

	are selected. You can choose to import data for a maximum of 90 days.
Do you wish to configure webhook for the tool?	Select the check box if you want to configure webhooks for the selected repositories.
Onboarding plans	Leave empty.

## 8. Select Order Now.

A request is created. When the request is approved, the repository and pipeline objects are associated to the app record and webhooks are configured for real-time tracking. Historical data is imported for the selected repositories. The **Track** field is automatically enabled for imported repositories and pipelines. For repositories the **Track file changes** is also automatically enabled.

### Onboard GitHub to DevOps Change Velocity — Classic

Connect your GitHub instance to discover, configure, and import repositories and pipelines.

#### Before you begin

Role required: sn\_devops.admin or sn\_devops.tool\_owner

Complete the steps specified in the [Getting started](#) section before connecting to a tool.

#### About this task

- **Connect** to GitHub and get the webhook URL when you submit a DevOps tool record.
- **Discover** repositories.
- **Configure** the webhook in the GitHub repository.
- **Import** branch and commit records.

#### Procedure

1. Enter the GitHub instance details to connect to DevOps Change Velocity by navigating to **All > DevOps > Tools > Create New**.
2. Enter a value in the **Tool Name** field and fill in the tool details.

#### Create DevOps Tool form

Field	Description
Tool integration	Tool to integrate. In this case, select <b>GitHub</b> .
Tool URL	URL of the existing GitHub instance to integrate. For e.g, <a href="https://api.github.com">https://api.github.com</a>
GitHub credential type	<ul style="list-style-type: none"> <li>◦ Basic Auth <ul style="list-style-type: none"> <li>▪ GitHub username</li> <li>▪ Personal access token (classic)</li> </ul> </li> </ul> <p>Only a personal access token (classic) is supported with basic authentication. When you generate a personal access</p>

Field	Description
	<p>token (classic) for GitHub, you must specify the scopes to authorize if you are not granting complete access. The minimum scopes that you must select for authorization are repo, admin:repo_hook, and user:email.</p> <ul style="list-style-type: none"> <li>◦ OAuth</li> </ul> <p>GitHub Tool Credential. See <a href="#">Setting up GitHub OAuth 2.0 credentials for DevOps</a>.</p>

3. Optional: Select **MID Server** for an on-premises tool that is attached to a MID Server. Application is automatically set to DevOps and capability is set to REST.

4. Select **Submit**.

On successful tool creation, you are taken to the tool record page.

5. If you want to control access to the tool, add the groups that must be given access to the tool in the **Maintained by** field on the **Access** tab.

The tasks the users in the groups can perform depends on the roles assigned to them.

- DevOps Tool Owner role: Can view and edit the tool.
- DevOps App Owner role: Can view the tool and can associate, discover, import historical data, and modify pipeline steps (if applicable) of the tool's objects (such as plans, repositories, and pipelines).
- DevOps Administrator role: Can edit all tools.
- Other DevOps roles: Can view the tool.

**Note:** Only groups containing users with DevOps roles are available for selection in the **Maintained by** field.

The **All App Owners can view and associate tool objects to applications** option becomes available for selection if you choose to restrict access to the tool. This option enables all users having the DevOps App Owner role to access the tool. If selected, they'll be able to view, associate, discover, import historical data, and modify pipeline steps (if applicable) of the tool's objects.

6. Select **Discover** to discover existing repositories or pipelines for the tool. Repository records are added to the Repositories related list.
7. Automatically configure the webhook URL in a GitHub repository to send notifications to the DevOps tool by selecting **Configure**. Alternatively, you can choose to enable nightly polling to fetch data system for any tracked repositories or pipelines by setting the **Enable Polling** property to **Yes**.

**Note:** If you do not have admin privileges for your GitHub tool (to allow automatic configuration of the webhook URL), you might need to have the tool admin user configure it for you (create and configure the webhook URL manually in your tool instance). Once the webhook is configured in the tool, Enter Manual Configuration Mode to connect to the tool manually, then exit.

8. Click **Import** to import historical data from the repository.

Imported branch records and commit records from the repository are added to the corresponding related lists.

9. In the **App** field, click the lookup list and select an App record to associate with the repository, or click **New** to create one.

### Setting up GitHub OAuth 2.0 credentials for DevOps Change Velocity

Create OAuth 2.0 credentials for GitHub Apps or OAuth apps and use them to connect your GitHub instance.

If you want to use Basic Authentication credentials instead of OAuth 2.0, skip this section and proceed to onboard GitHub using one of the following options:

- [Onboard GitHub to DevOps Change Velocity — Workspace](#).
- [Onboard GitHub to DevOps Change Velocity — Classic](#).

**Note:** DevOps connection and credential aliases (DevOpsAlias1 through DevOpsAlias10) are used to connect automatically when you set up your GitHub instance using OAuth 2.0. To connect to more than 10 tools, or if you receive an error saying all DevOpsAlias connection and credential aliases are being used, an admin can create additional aliases in the sn\_devops application scope. For more information, see [Create a Connection & Credential alias](#). If necessary, you can reuse a DevOpsAlias alias that is no longer in use by inactivating the HTTP connection.

### OAuth 2.0 credentials for GitHub Apps - JWT

Perform the following steps to integrate your GitHub Apps using the JWT bearer token.

Role required: oauth\_admin.

#### Configure the GitHub App in your GitHub account (JWT)

Create a custom GitHub App from your GitHub account to enable OAuth 2.0 authentication with your ServiceNow instance.

#### Before you begin

GitHub requirement: GitHub App configured to integrate with ServiceNow.

Role required: No instance role required

#### About this task

Complete these steps from your GitHub account. See [Building GitHub Apps](#) on the GitHub Developer site for instructions on creating and configuring custom applications.

#### Procedure

1. From your GitHub account, create your GitHub App by navigating to **Developer Settings > GitHub Apps**.
2. In the **Homepage URL** field, enter `https://<instance-name>.service-now.com`.
3. In the **User authorization callback URL** field, enter `https://<instance-name>.service-now.com/oauth_redirect.do`.
4. In the **Identifying and authorizing users** section, deselect the **Expire user authorization tokens** field.
5. In the **Webhook** section, deselect the **Active** field.
6. Leave the remaining fields empty (default).
7. In the **Repository permissions** section, configure the following settings.

Action	Read-only
Contents	Read-only
Deployments	Read-only
Environments	Read-only
Metadata	Read-only
Secrets	Read-only
Webhooks	Read and write  <b>i Note:</b> Read and write permissions are required to configure webhooks from ServiceNow.
Pull requests	Read-only

8. Leave the remaining permissions as No access (default).
9. After creating the GitHub App, generate a new private key and save it to your machine.
10. Install the newly created GitHub App on the accounts of your choice.

- a. From the GitHub Apps settings page, select your app.
- b. In the left sidebar, select **Install App**.
- c. Select **Install** next to the organization or personal account containing the correct repository.
- d. Install the app on all repositories or select repositories.  
For more information, see [Installing GitHub Apps](#).

### Generate the JKS certificate for GitHub

Generate a JKS certificate for the JWT authentication.

#### Before you begin

Role required: admin

#### Procedure

1. Create a CA signed certificate using the GitHub App private key:

```
openssl req -new -x509 -key <file-name>.pem -out <certificate-name>.pem
-days 1095
```

2. Enter the required details.

3. Create the PKCS 12 file using the GitHub App private key and CA signed certificate:

```
openssl pkcs12 -export -in <certificate-name>.pem -inkey
<file-name>.pem -certfile <certificate-name>.pem -out
<PKCS-12-file-name>.p12
```

4. Provide the export password.

**5. Create the JKS file:**

```
keytool -importkeystore -srckeystore <PKCS-12-file-name>.p12
-srcstoretype pkcs12 -destkeystore <JKS-certificate-filename>.jks
-deststoretype JKS
```

**6. Provide the destination and source keystore passwords.**

**Attach the GitHub Java Key Store certificate to your instance**

Enable the JWT Bearer Grant token authentication by attaching the valid GitHub Java KeyStore (JKS) certificate to your ServiceNow instance.

**Before you begin**

Ensure the availability of a valid Java KeyStore certificate.

Role required: admin

**Procedure**

1. Navigate to **All > System Definition > Certificates**.
2. Select **New**.
3. On the form, fill in the fields.

**X.509 Certificate form fields**

Field	Description
Name	Name to uniquely identify the record. For example, My GitHub App Certificate.
Notify on expiration	Option to specify the users to be notified when the certificate expires.
Warn in days to expire	Number of days to send a notification before the certificate expires.
Active	Option to enable the certificate.
Type	Option to select the type of the certificate. Select <b>Java Key Store</b> .
Expires in days	Number of days until the certificate expires.
Key store password	Password associated with the certificate (hint: the destination KeyStore password previously created).
Short description	Summary about the certificate.

4. Select the attachments icon () and attach a JKS certificate.

5. Select **Validate Stores/Certificates**.

**Create a JWT signing key for the GitHub JKS certificate**

Create a JSON Web Token (JWT) signing key to assign to your GitHub Java KeyStore certificate.

**Before you begin**

Role required: admin, sn\_devops.admin

## Procedure

1. Navigate to **All > System OAuth > JWT Keys**.
2. Select **New**.
3. On the form, fill in the fields.

### JWT Keys form fields

Field	Description
Name	Name to uniquely identify the JWT signing key. For example, My GitHub App JWT Key.
Signing Keystore	Valid JKS certificate attached in the previous task. For example, My GitHub App Certificate.
Key Id	Unique Id to identify which key is used when multiple keys are used to sign tokens.
Signing Algorithm	Algorithm to sign with the JWT key (hint: RSA 256).
Signing Key Password	Password associated with the signing key (hint: the source keystore password previously created).
Active	Option to enable the key.

4. Select **Submit**.

### Create a JWT provider for your GitHub signing key

Add a JSON Web Token (JWT) provider to your ServiceNow instance for GitHub.

#### Before you begin

Role required: admin, sn\_devops.admin

## Procedure

1. Navigate to **All > System OAuth > JWT Providers**.
2. Select **New**.
3. On the form, fill in the fields.

### JWT Provider form fields

Field	Description
Name	Name to uniquely identify the JWT provider. For example, My GitHub App JWT Provider.
Expiry Interval (sec)	Number in seconds to set the lifespan of JWT provider tokens (Hint: You can leave it as default).
Signing Configuration	Valid JWT signing key previously created. For example, My GitHub App JWT Key.

4. Right-click the form header, and select **Save**.

5. Enter your GitHub App **App ID** (available in the **About** section of your GitHub App configuration in GitHub ) as the value of the **iss** claim, in the Standard Claims related list.
6. Leave **aud** and **sub** values blank (default).

### Register GitHub as an OAuth Provider (JWT)

Use the information generated during GitHub App account configuration to register GitHub as an OAuth provider and allow the instance to request OAuth 2.0 tokens.

#### Before you begin

Role required: admin, sn\_devops.admin

#### Procedure

1. Navigate to **All > System OAuth > Application Registry**.
2. Select **New**.  
The **What kind of OAuth application?** message is displayed.
3. Select **Connect to a third party OAuth Provider**.
4. On the form, fill in the fields.

#### Application Registry form fields

Field	Description
Name	Name to uniquely identify the record. For example, enter <b>My GitHub App Provider</b> .
Client ID	Client ID of your GitHub App (hint: available in the <b>About</b> section of your GitHub App configuration in GitHub ).
Client Secret	Client secret of your GitHub App (hint: available in the <b>About</b> section of your GitHub App configuration in GitHub ).
OAuth API script	Option that enables you to reference an amended OAuthUtil script include. Select <b>OAuthDevOpsGitHubJWTHandler</b> .
Default Grant type	Type of grant associated with application registry. Select <b>JWT Bearer</b> .
Token URL	The location of the token endpoint that the instance uses to retrieve and refresh tokens. For cloud version, enter: <code>https://api.github.com/app/installations/&lt;installation_id&gt;/access_tokens</code> For enterprise version, enter: <code>https://&lt;HOST_URL&gt;/api/v3/app/installations/&lt;installation_id&gt;/access_tokens</code> (hint: go to Install App section in your GitHub App configuration in GitHub and click on gear icon to configure your app. The installation id will be in the webpage URL. <code>https://github.com/settings/installations/&lt;installation_id&gt;</code> )

5. Leave the rest of the form fields as default.
6. Right-click the form header, and select **Save**.
7. Open the default profile created in the **OAuth Entity Profiles** related list.
8. Populate the **JWT Provider** field with the JWT provider previously created and save the form.

**9.** Navigate to **Key Management > Module Access Policies > All**.

**10.** In the **Result** field, select **Track** and save the changes.

#### Create a credential record for GitHub App provider (JWT)

Create a credential record to the GitHub App provider previously created to authorize actions.

#### Before you begin

Role required: admin, sn\_devops.admin

#### Procedure

**1.** Navigate to **All > Connections & Credentials > Credentials**.

**2.** Select **New**.

The **What type of Credentials would you like to create?** message is displayed.

**3.** Select **OAuth 2.0 Credentials**.

**4.** On the form, fill in the fields.

#### OAuth 2.0 Credentials form fields

Field	Value required
Name	Name to uniquely identify the record. For example, enter <code>My GitHub App Credential</code> .
Active	Option to enable the record.
OAuth Entity Profile	Default OAuth Entity profile created in the Application Registry.

**5.** Save the record.

**6.** Select the **Get OAuth Token** related link to generate the OAuth token.

#### OAuth 2.0 credentials for GitHub Apps - Authorization Code

Perform the following steps to integrate your GitHub Apps using Authorization code.

Role required:

- oauth\_admin
- Admin account in GitHub

**Note:** Only user-level repos are supported.

#### Configure the GitHub App in your GitHub account (Authorization Code)

Create a custom GitHub App from your GitHub account to enable OAuth 2.0 authentication with your ServiceNow instance.

#### Before you begin

GitHub requirement: GitHub App configured to integrate with ServiceNow.

Role required: No instance role required

## About this task

Complete these steps from your GitHub account. See [Building GitHub Apps](#) on the GitHub Developer site for instructions on creating and configuring custom applications.

## Procedure

1. From your GitHub account, create your GitHub App by navigating to **Developer Settings > GitHub Apps**.
2. In the **Homepage URL** field, enter `https://<instance-name>.service-now.com`.
3. In the **User authorization callback URL** field, enter `https://<instance-name>.service-now.com/oauth_redirect.do`.
4. In the **Identifying and authorizing users** section, deselect the **Expire user authorization tokens** field.
5. In the **Webhook** section, deselect the **Active** field.
6. Leave the remaining fields empty (default).
7. In the **Repository permissions** section, configure these settings.

Action	Read-only
Contents	Read-only
Deployments	Read-only
Environments	Read-only
Metadata	Read-only
Secrets	Read and write

8. Leave the remaining permissions as `No access` (default).
9. Install the newly created GitHub App on the accounts of your choice.

## Register GitHub as an OAuth Provider (Authorization Code)

Use the information generated during GitHub App account configuration to register GitHub as an OAuth provider and allow the instance to request OAuth 2.0 tokens.

## Before you begin

Role required: `admin, sn_devops.admin`

## Procedure

1. Navigate to **All > System OAuth > Application Registry**.
2. Click **New**.  
The system displays the message **What kind of OAuth application?**
3. Select **Connect to a third party OAuth Provider**.  
The system displays an empty Application Registries form.
4. Complete the form.

Field	Value required
Name	Enter any name to uniquely identify the record. For example, enter <code>My GitHub App Provider</code> .

Field	Value required
Client ID	Enter the client ID of your GitHub App (hint: available in the <b>About</b> section of your GitHub App configuration in GitHub ).
Client Secret	Enter the client secret of your GitHub App (hint: available in the <b>About</b> section of your GitHub App configuration in GitHub ).
OAuth API script	Select <b>OAuthDevOpsGitHubHandler</b> .
Default Grant type	Select <b>Authorization Code</b> .
Authorization URL	Enter <code>https://github.com/login/oauth/authorize</code> . For an on-premises deployment, use the proper GitHub host URL.
Token URL	Enter <code>https://github.com/login/oauth/access_token</code> . For an on-premises deployment, use the proper GitHub host URL.

5. Leave the rest of the form fields as default.

6. Right-click the form header, and click **Save**.

- The system validates the OAuth credentials and populates the **Redirect URL** (Hint: It should match the **User authorization callback URL** previously provided in your GitHub App configuration).
- The system populates **OAuth Entity Profile** with **Grant Type** as **Authorization Code**. For example, **OAuth Entity Profile** is created with default **Name, My GitHub App Provider default\_profile**

#### Create a credential record for GitHub App provider (Authorization Code)

Create a credential record in ServiceNow to the GitHub App provider previously created to authorize actions.

#### Before you begin

Role required: admin, sn\_devops.admin

#### Procedure

1. Navigate to **All > Connections & Credentials > Credentials**.

2. Click **New**.

The system displays the message **What type of Credentials would you like to create?**.

3. Select **OAuth 2.0 Credentials**.

The pop-up window displays an empty OAuth 2.0 Credentials form.

4. Fill in these values.

Field	Value required
Name	Enter any name to uniquely identify the record. For example, enter <code>My GitHub App Credential</code> .
Active	Enable

Field	Value required
OAuth Entity Profile	Select the default OAuth Entity profile you created previously.
Applies to	Select the MID Servers that can use this credential. For example, select <b>All MID Servers</b> .
Order	Select the order to apply this credential. For example, enter 100.

5. Save the record.

6. Click the **Get OAuth Token** related link to generate the OAuth token.

### OAuth 2.0 credentials for OAuth Apps

Perform the following steps to integrate GitHub using your OAuth Apps.

Role required: oauth\_admin.

### Configure the OAuth App in your GitHub account

Create a custom OAuth App from your GitHub account to enable OAuth 2.0 authentication with your ServiceNow instance.

#### Before you begin

GitHub requirement: GitHub OAuth App configured to integrate with ServiceNow.

Role required: No instance role required.

#### About this task

Complete these steps from your GitHub account. See [Building OAuth Apps](#) on the GitHub Developer site for instructions on creating and configuring custom applications.

#### Procedure

- From your GitHub account, create your OAuth App by navigating to **Developer Settings > OAuth Apps**.
- In the **Homepage URL** field, enter `https://<instance-name>.service-now.com`.
- In the **Authorization callback URL** field, enter `https://<instance-name>.service-now.com/oauth_redirect.do`.

#### Register GitHub as an OAuth Provider

Use the information generated during GitHub OAuth App account configuration to register GitHub as an OAuth provider and allow the instance to request OAuth 2.0 tokens.

#### Before you begin

Role required: admin, sn\_devops.admin

#### Procedure

- Navigate to **All > System OAuth > Application Registry**.
- Click **New**.  
The system displays the message **What kind of OAuth application?**
- Select **Connect to a third party OAuth Provider**.  
The system displays an empty Application Registries form.
- Complete the form.

Field	Value required
Name	Enter any name to uniquely identify the record. For example, enter <code>My GitHub OAuth App Provider</code> .
Client ID	Enter the client ID of your OAuth App.
Client Secret	Enter the client secret of your OAuth App.
OAuth API script	Select <b>OAuthDevOpsGitHubHandler</b> .
Default Grant type	Select <b>Authorization Code</b> .
Authorization URL	<p>Enter <code>https://github.com/login/oauth/authorize</code>. For an on-premises deployment, use the proper GitHub host URL.</p>
Token URL	<p>Enter <code>https://github.com/login/oauth/access_token</code>. For an on-premises deployment, use the proper GitHub host URL.</p>

5. Leave the rest of the form fields as default.
6. Right-click the form header, and click **Save**.
  - The system validates the OAuth credentials and populates the **Redirect URL**.

**Tip:** The URL should match the **Authorization callback URL** that was previously provided in your OAuth App configuration

  - The system populates **OAuth Entity Profile** with **Grant Type** as **Authorization Code**. For example, **OAuth Entity Profile** is created with default **Name**, **My GitHub OAuth App Provider default\_profile**
7. In the **OAuth Entity Scopes** related list, create an entry where **Name** is set to **scope** and **OAuth scope** is set to **repo**, and save the form.
8. In the **OAuth Entity Profiles** related list, open the default profile.
9. In the **OAuth Entity Profiles Scopes** related list, select the **OAuth Entity Scope** you created, and save the form.

#### Create a credential record for OAuth App provider

Create a credential record in ServiceNow to the OAuth provider previously created to authorize actions.

#### Before you begin

Role required: admin, sn\_devops.admin

#### Procedure

1. Navigate to **All > Connections & Credentials > Credentials**.
2. Click **New**.  
The system displays the message **What type of Credentials would you like to create?**.
3. Select **OAuth 2.0 Credentials**.

The pop-up window displays an empty OAuth 2.0 Credentials form.

4. Fill in these values.

Field	Value required
Name	Enter any name to uniquely identify the record. For example, enter <b>My OAuth App Credential</b> .
Active	Enable
OAuth Entity Profile	Select the default OAuth Entity profile you created previously.
Applies to	Select the MID Servers that can use this credential. For example, select <b>All MID Servers</b> .
Order	Select the order to apply this credential. For example, enter 100.

5. Save the record.

6. Click the **Get OAuth Token** related link to generate the OAuth token.

#### Additional information - GitHub Actions

Additional information on GitHub Actions, such as configuring secrets in GitHub Actions, workflows in the GitHub repository, and limitations.

#### Secrets in GitHub Actions

Create secrets (credentials) in GitHub repository or GitHub organization. Secrets are environment variables (encrypted) that you create in an organization or repository. These secrets are available to use in GitHub Actions workflows. For more information, see [Encrypted secrets](#).

Secret	Description
SN_INSTANCE_URL	ServiceNow instance URL. For example, <a href="https://&lt;instance_name&gt;.service-now.com">https://&lt;instance_name&gt;.service-now.com</a> .
SN_ORCHESTRATION_TOOL_ID	Sys_id for the GitHub tool created in ServiceNow instance.
SN_DEVOPS_USER	DevOps integration user name.
SN_DEVOPS_PASSWORD	Password for the specified DevOps integration user name.

#### Workflows in the GitHub repository

Create a YAML file to define workflow configuration in your GitHub repository.

The following points must be considered while defining the workflow:

- All workflows of your repository must have either a .yml or .yaml file extension. All workflows must be under `.github/workflows` directory and follow the syntax defined in the [Workflow syntax for GitHub Actions](#).
- The name of the workflow must match with the workflow file name.

- The names of the workflows under **All workflows** on the **Actions** tab must match with the workflows saved under the `.github/workflows` directory of your repository.
- A display name must be given for every job and must be unique for every job in the workflow.
- Use the `workflow_dispatch` event to trigger a workflow manually.

## GitHub Actions workflow run details in DevOps

After you configure the webhooks, the notifications from GitHub Actions are sent to ServiceNow DevOps whenever a workflow is executed or triggered.

The following details are sent to the ServiceNow instance when a workflow is manually executed or triggered automatically on a GitHub repository.

- The `workflow_job` webhook notifies the ServiceNow instance with status of the job (queued, `in_progress`, completed) when the workflow is manually run or automatically triggered on a GitHub Repository.
- Inbound Events are created in the ServiceNow instance for status of the job (queued, `in_progress`, and completed) and `in_progress` events are ignored.
- Queued and completed inbound events are processed and ServiceNow DevOps Pipeline Steps and Orchestration Tasks are created for jobs configured in the workflow.
- Pipeline Execution is created for every workflow run with Task Executions and Step Executions records created for each job executed in the workflow run.
- You can use the Pipeline UI to visualize interactions and results across a pipeline execution.

## GitHub Actions limitations for DevOps Change Velocity integration

The following are supported from DevOps Change Velocity for GitHub Actions.

- Only User repositories are discovered for integration with the GitHub Actions.
- For GitHub Organizations, use a specific account (with access to the required organizations) with personal access token for integrating with ServiceNow DevOps.
- Only the latest GitHub Actions scan results can be pulled from the instance for a workflow run.
- ServiceNow DevOps Change Automation using custom action or environment is not supported for parallel jobs.
- You must authorize the code for OAuth 2.0 or JWT.
- Only the JUnit and TestNG test results XMLs generated using Maven and Gradle tools and Selenium test result XMLs generated using JUnit/TestNG framework are currently supported by the ServiceNow DevOps Test Report custom action.

Limitations of the GitHub Actions integration:

- No REST APIs are supported to pull SonarQube scan results, unit test results, run commits, artifacts details, and package details from your ServiceNow instance.
- No Callback URL to pause and resume workflow run from the ServiceNow instance.

- GitHub environments are available to private repositories only in GitHub Enterprise Cloud.
- The GitHub Actions and GitHub environments are supported in the GitHub Enterprise Server from version 3.3 onwards.
- GitHub environments can only be created in GitHub Repository.
- User who creates GitHub tool in the ServiceNow instance must be a reviewer to approve the workflow for GitHub Environments.
- Each workflow that is run with GitHub Environment is limited to 30 days.
- For parallel jobs, the webhook notification payload does not contain information on the jobs executed in parallel with a sequence number. Due to this limitation, the sequence of jobs depends on the execution order returned by the response of the (/repos/{owner}/{repo}/actions/runs/{run\_id}/jobs) API.

### ServiceNow DevOps custom actions from GitHub marketplace

Use the custom actions from the GitHub marketplace to collect SonarQube scan data, pause or resume workflow, or resume workflow until a change request is approved or rejected in your instance.

#### ServiceNow DevOps SonarQube custom action

Save the SonarQube scan results of a project initiated by the workflow run in your ServiceNow instance, you must create the SonarQube tool in your instance and use the ServiceNow DevOps SonarQube custom action at the steps level of a job in the workflow.

View the SonarQube analysis results using one of the following methods:

- Navigate to **DevOps > Software Quality Results > Software Quality Summary**.
- Navigate to **DevOps > Orchestrate > Task Execution** and selecting a relevant **Task Execution** record.

Perform the following tasks to use the custom action.

- Create SonarQube tool in your instance.

Generate token from **User > My Account > Security** page in your SonarQube tool and create tool in the ServiceNow instance using the generated token. For more information, see [Onboard SonarQube to DevOps Change Velocity — Workspace](#) and [Onboard SonarQube to DevOps Change Velocity — Classic](#).

- Create the following Secrets to save SonarQube scan results in the ServiceNow instance.
  - SONAR\_HOST\_URL: SonarQube instance URL. For example, `https://sonarcloud.io`
  - SONAR\_PROJECT\_KEY: The key to identify a project in the SonarQube instance. For example, `org.examples:demo`
- Configure the ServiceNow DevOps SonarQube custom action in the workflow.

The custom action `servicenow-devops-sonar` must be configured at steps level of job in the workflow with **uses** keyword. The inputs of this custom action must not be tampered to save SonarQube analysis results in the ServiceNow instance. For more information, see [ServiceNow DevOps Sonar](#).

#### ServiceNow DevOps Test Report custom action

Save unit test results of the project initiated by the workflow run in ServiceNow instance. The ServiceNow DevOps Test Report custom action must be used at the steps level of job in the workflow.

View the unit test results saved in your ServiceNow instance by navigating to **DevOps > Test Results > Test Summaries**.

You can also view by navigating to **DevOps > Orchestrate > Task Execution** and clicking a relevant **Task Execution** record.

Perform the following tasks to use the custom action.

- Create Secrets in the GitHub Repository.
- Configure the DevOps Test Report custom action in the workflow.

The custom action `servicenow-devops-test-report` must be configured at steps level of job in the workflow with **uses** keyword. The inputs of this custom action must not be tampered to save the unit test results in your ServiceNow instance. For more information, see [ServiceNow DevOps Test Report ↗](#).

#### ServiceNow DevOps Register Artifact custom action

Save artifacts created or deployed by the workflow run in your ServiceNow instance. Use the ServiceNow DevOps Register Artifact custom action at steps level of job in the workflow.

View the artifacts details saved in your instance by navigating to **DevOps > Artifact > Artifacts**.

Perform the following tasks to use the custom action:

- Create Secrets in the GitHub Repository.
- Configure ServiceNow DevOps Register Artifact custom action in the workflow.

The custom action `servicenow-devops-register-artifact` must be configured at steps level of job in the workflow with **uses** keyword. The inputs of this custom action must not be tampered to save artifact details in your instance. For more information, see [ServiceNow DevOps register artifacts ↗](#).

#### ServiceNow DevOps Register Package custom action

Save package created or deployed by the workflow run in the ServiceNow instance. Use the ServiceNow DevOps Register Package custom action at steps level of job in the workflow.

View the package details saved in the ServiceNow instance by navigating to **DevOps > Artifact > Packages**.

Perform the following tasks to use the custom action.

- Create Secrets in the GitHub Repository.
- Configure ServiceNow DevOps Register Artifact custom action in the workflow.

The custom action `servicenow-devops-register-package` must be configured at steps level of job in the workflow with **uses** keyword. The inputs of this custom action must not be tampered to save artifact details in your instance. For more information, see [ServiceNow DevOps register package ↗](#).

#### ServiceNow DevOps Change Automation custom action

Create a change request in the ServiceNow instance to pause and resume the workflow run from the ServiceNow instance. Use the ServiceNow DevOps Change Automation custom action at steps level of job in the workflow. If the change is not created within the time period specified in the threshold (`changeCreationTimeOut`), and the **abortOnChangeCreationFailure** parameter is enabled, the pipeline will be aborted.

This custom action creates a change request in ServiceNow, enables the **Change Control** option, and polls the ServiceNow instance at the defined time interval for change status until either the change is approved or rejected or the timeout threshold (`timeout`) is reached. GitHub aborts the workflow run if the timeout threshold is reached and the **abortOnChangeStepTimeout** parameter is enabled.

This custom action immediately resumes the workflow run when change receipt is enabled for pipeline in ServiceNow without waiting for the change to be approved or rejected in the ServiceNow instance.

View the change created for the workflow run in the ServiceNow instance by navigating to **DevOps > Orchestrate > Pipeline Change Requests**.

The change number with status **pending\_decision** is displayed in the GitHub Actions console while polling the ServiceNow instance for change status. The details like change comments, approved by, approved on, and status are logged in GitHub Actions console after the change is approved or rejected or canceled by the user in the ServiceNow instance.

Perform the following tasks to use the custom action.

- Create Secrets in the GitHub Repository.
- Configure ServiceNow DevOps Change Automation custom action in the workflow.

The custom action `servicenow-devops-change` must be configured at steps level of job in the workflow with **uses** keyword. The inputs of this custom action must not be tampered to create change in your instance and poll for change status every `<x>` second. For more information, see [ServiceNow DevOps change automation](#).

#### ServiceNow DevOps Get Change Github Action

The [ServiceNow DevOps Get Change Github Action](#) custom action must be added at the step level in a pipeline job to retrieve the change request number from a ServiceNow instance by specifying change details.

#### ServiceNow DevOps Update Change Github Action

The [ServiceNow DevOps Update Change Github Action](#) custom action must be added at the step level in a job to update the change request in a ServiceNow instance. The change request number whose details need to be updated and the change request details to be updated must be specified as input.

#### GitHub Environments for ServiceNow DevOps Change

To resume the workflow that is waiting for review approval from the ServiceNow instance until the ServiceNow change is approved or rejected, you must create an environment secret in GitHub Repository and use the environment at job level in the workflow.

This custom action creates change in the ServiceNow instance, enables the **Change Control** option, creates a Callback URL, and resumes the workflow run using the Callback URL after the change is approved or rejected or canceled by a user in the ServiceNow instance.

When the change receipt is enabled for pipeline in your instance, the workflow is immediately resumed without waiting for change to be approved or rejected in the ServiceNow instance.

View the change created for the workflow run in ServiceNow instance by navigating to **DevOps > Orchestrate > Pipeline Change Requests**.

The details like change comments, approved by, approved on, and status are logged in GitHub tool after the workflow run is resumed from ServiceNow.

When a reviewer manually approves or rejects the workflow run in GitHub, ServiceNow is notified with completed job event. The Pipeline Execution, Step Execution, and Task Execution record relevant for the workflow run are updated to reflect the job of the status in your instance.

When the workflow run is canceled by a user in GitHub, your instance is notified with completed job event. The Pipeline Execution, Step Execution, and Task Execution record relevant for the workflow run are updated to reflect the job of the status in your instance.

Perform the following tasks to use the custom action.

- Configure Environments in GitHub Repository.
  - Navigate to **Settings > Environments** from a GitHub Repository and click **New environment** to create an environment.
  - Configure the Environment protection rules for the environment that require manual approval for deployments. For more information, see [Environment protection rules](#).
  - Ensure that the user who creates the GitHub tool in your instance must be a reviewer to approve the workflow.
- Create Secret for GitHub environment.

By default, environment that require manual approval is paused by GitHub and waits for reviewer approval. A secret with name SERVICENOW\_DEVOPS\_CHANGE must be created to resume the workflow run from your instance after the change is approved or rejected .

- Configure Environment in the workflow.

Use environment name for keyword **environment** at job level in the workflow, so that the workflow run can be resumed from your instance with the Callback URL.

```
<job_name>:
  environment: <environment_name>
  name: <display_name_of_job>
  runs-on: <github_hosted_runner>
  steps:
```

```
- name: Deployment Scripts
  run: echo Deployment started
```

Ensure that the environment name is the same as the **environment** keyword that is created and configured for review approval with environment secret SERVICENOW\_DEVOPS\_CHANGE.

## Bitbucket integration with DevOps Change Velocity

Integrate Bitbucket coding tool with DevOps to track Bitbucket repositories.

Use one of the following options to onboard Bitbucket. For a guided experience, use the workspace to onboard a tool. Alternatively, you can use the Service Catalog or Classic experience.

### Onboard Bitbucket to DevOps Change Velocity — Workspace

Connect your Bitbucket instance using the playbook in DevOps Change workspace to discover repositories.

#### Before you begin

Role required: sn\_devops.admin or sn\_devops.tool\_owner

Complete the steps specified in the [Getting started](#) section before connecting to a tool.

#### Procedure

1. Navigate to **Workspaces > DevOps Change Workspace** and use one of the following options to open the Playbook to onboard Bitbucket.

Option	Steps
Homepage	<ol style="list-style-type: none"> <li>Select the <b>Connect tool</b> widget</li> <li>On the# Connect to a tool# modal, select Bitbucket from the <b>Code</b> category.</li> </ol>
Applications module	<ol style="list-style-type: none"> <li>Select <b>Applications ()</b>.</li> <li>Select an existing application, or create one. To create an application, see <a href="#">Create an application in DevOps Change Velocity</a>.</li> <li>From the# Recommended actions# pane, select the#<b>Connect a tool</b> card.</li> <li>On the# Connect to a tool# modal, select Bitbucket from the <b>Code</b> category.</li> </ol>
Tools module	<ol style="list-style-type: none"> <li>Select <b>Tools ()</b>.</li> <li>From the Tools list, select <b>Coding tools</b>.</li> <li>Select <b>Connect coding tool</b>.</li> <li>On the# Connect to a tool# modal, select Bitbucket.</li> </ol>

2. Specify a name for the tool in the **Tool name** field, and select **Next**.

3. On the Bitbucket instance details playbook activity:
  - a. Enter the URL of your Bitbucket instance.
  - b. Enter the login credentials of the global admin for the Bitbucket instance.
  - c. If your Bitbucket instance is attached to a MID Server, select the MID Server option and enter its details. A MID server is required if your tool instance is hosted on-prem. For more information about MID server, see [MID Server selection](#).

**4. Select Connect.**

5. Permission checks are run and the permission requirements are displayed in a pop-up.

**6. Select Continue or Connect Anyway.**

If the connection is successful, the tool is created in ServiceNow instance and connected to your Bitbucket instance.

**7. Specify the access for the tool.**

- a. If you want to control access to the tool, add the groups that must be given access to the tool in the **Maintained by** field.

The tasks these users in the groups can perform depends on the role assigned to them.

- DevOps Tool Owner role: Can view and edit the tool.
- DevOps App Owner role: Can view the tool and can associate, discover, import historical data, and modify pipeline steps (if applicable) of the tool's objects (such as plans, repositories, and pipelines).
- DevOps Administrator role: Can edit all tools.
- Other DevOps roles: Can view the tool.

**i Note:** If you don't select a group and skip this step, all users with the DevOps Tool Owner role will be able to edit the tool.

- b. If you choose to control access to the tool, the **All App Owners can view and associate tool objects to applications** option becomes available for selection.

This option enables all users having the DevOps App Owner role to access the tool. If selected, they'll be able to view, associate, discover, import historical data, and modify pipeline steps (if applicable) of the tool's objects.

**c. Select Assign.**

8. After the connection is successful, the repositories are automatically discovered. Select the items that you want to configure from the list.

**9. Optional: Select Configure.**

This will automatically configure webhooks. This is recommended as real-time notifications are ideal to maintain the most up-to-date information particularly for automating change requests.

(Optional) You can set up webhook by configuring it manually later. Alternatively, you can choose to enable nightly polling to fetch data system for any tracked repositories by setting the **Enable Polling** property to **Yes**.

10. Select the repositories that you want to track from the **Select repositories to track** playbook activity.

**11.** Select **Next** to review the details of the successfully connected Bitbucket tool.

**Note:** If your tool credential has changed, you must update the credentials in your ServiceNow instance. For more information, see [Update third-party tool credentials in DevOps Change Velocity](#).

**12.** From the **Summary** page, select **View tool record** to review the details of the connected instance and the repositories discovered from it.

## Result

You've successfully onboarded your Bitbucket tool to DevOps Change Velocity.

### Onboard Bitbucket to DevOps Change Velocity — Service Catalog

Create, connect, discover, and configure your Bitbucket instance using the ServiceNow Service Catalog.

#### Before you begin

Role required: sn\_devops.admin or sn\_devops.tool\_owner

#### Procedure

**1.** Navigate to **All > Service Catalog > Catalog Definitions > Maintain Items** and search for DevOps.

**Note:** You can also access the service catalog from Employee Center or Service portal.

**2.** From the DevOps catalog items, select and activate **DevOps App Onboarding** and **DevOps Tool Onboarding**.

**3.** After activating, select **DevOps Tool Onboarding** and select **Try it**.

**4.** In the DevOps Tool Onboarding form, enter the tool details:

Field	Description
Tool name	Name for your Bitbucket integration.
Tool integration	Select Bitbucket.
Tool URL	URL of your Bitbucket instance.
Tool username	Username for your Bitbucket account.
Tool password/ Access token	Bitbucket password or access token.
Use MidServer	Optional. Select#Mid Server# for an on-premises tool that is attached to a#Mid Server. Application is automatically set to #DevOps and capability is set to REST.

**5.** Select **Order Now**.

A request is created. When the request is approved, the tool is created, connected, and discovered.

6. From the DevOps catalog items, select **DevOps App Onboarding**.

7. In the DevOps App Onboarding form, enter the details:

Are you creating a new app or adding to an existing one?	Select from the options whether to create a new app or use an existing app.
App	Enter the name for the app that you're creating or using.
Onboarding pipelines	Leave empty.
Onboarding repositories	Enter the connected Bitbucket tool name.
Repositories	Select the repositories for which you want to configure webhooks and import historical data.
Import from and Import to	Select the dates for which you want to import the data. By default, the last 30 days are selected. You can choose to import data for a maximum of 90 days.
Do you wish to configure webhook for the tool?	Select the check box if you want to configure webhooks for the selected repositories.
Onboarding plans	Leave empty.

#### 8. Select **Order Now**.

A request is created. When the request is approved, the repository objects are associated to the app record and webhooks are configured for real-time tracking. Historical data is imported for the selected repositories. The **Track** field is automatically enabled for imported repositories. For repositories the **Track file changes** is also automatically enabled.

### Onboard Bitbucket to DevOps Change Velocity — Classic

Connect your Bitbucket instance to discover, configure, and import repositories.

#### Before you begin

Role required: sn\_devops.admin or sn\_devops.tool\_owner

Complete the steps specified in the [Getting started](#) section before connecting to a tool.

#### About this task

- **Connect** by using your Bitbucket instance details.
- **Discover** repositories.
- **Configure** to create the webhook in the Bitbucket repository automatically.
- **Import** branch and commit records.

#### Procedure

1. Enter the Bitbucket instance details to connect to DevOps Change Velocity by navigating to **All > DevOps > Tools > Create New**.
2. Enter a value in the **Tool Name** field and fill in the tool details.

## Create DevOps Tool form

Field	Description
Tool integration	Tool to integrate. In this case, select <b>Bitbucket</b> .
Tool URL	URL of the existing Bitbucket instance to integrate.
Tool username, Tool password / Access token	Login credentials of the existing Bitbucket instance.

**3.** Select **MID Server** for an on-premises tool that is attached to a MID Server.

Application is automatically set to DevOps and capability is set to REST.

**4.** Select **Submit**.

The tool is connected successfully.

On successful tool creation, you are taken to the tool record page.

**5.** If you want to control access to the tool, add the groups that must be given access to the tool in the **Maintained by** field on the **Access** tab.

The tasks the users in the groups can perform depends on the roles assigned to them.

- DevOps Tool Owner role: Can view and edit the tool.
- DevOps App Owner role: Can view the tool and can associate, discover, import historical data, and modify pipeline steps (if applicable) of the tool's objects (such as plans, repositories, and pipelines).
- DevOps Administrator role: Can edit all tools.
- Other DevOps roles: Can view the tool.

**Note:** Only groups containing users with DevOps roles are available for selection in the **Maintained by** field.

The **All App Owners can view and associate tool objects to applications** option becomes available for selection if you choose to restrict access to the tool. This option enables all users having the DevOps App Owner role to access the tool. If selected, they'll be able to view, associate, discover, import historical data, and modify pipeline steps (if applicable) of the tool's objects.

**6.** Select **Discover** to discover existing repositories for the coding tool.

Repository records are added to the **Repositories** related list.

**7.** Automatically configure the webhook URL in a Bitbucket repository to send notifications to the DevOps tool by selecting **Configure**.

Alternatively, you can choose to enable nightly polling to fetch data system for any tracked repositories by setting the *Enable Polling* property to **Yes**.

**8.** Select **Import** to import historical data from the repository.

**Note:** If your tool credential has changed, you must update the credentials in your ServiceNow instance. For more information, see [Update third-party tool credentials in DevOps Change Velocity](#).

Imported branch records and commit records from the repository are added to the corresponding related lists.

9. In the **App** field, click the lookup list and select an App record to associate with the repository, or click **New** to create one.

## GitLab integration with DevOps Change Velocity

Connect your GitLab instance to discover repositories and pipeline definitions and configure real-time notifications or polling to enable change traceability and automation.

### Overview

DevOps Change Velocity supports both Code (Repository) and Orchestration (Pipelines) capabilities for the GitLab tool. In case of pipelines, only basic pipelines are supported but not the multi-project ones.

- Connect: Discover repositories and pipeline definitions by connecting your GitLab instances to DevOps Change Velocity.
- Configure: Enable sending real-time notifications for commits and pipelines by automatically creating a Webhook in GitLab so that this data can be used to create change policies.

### Key points

- A pipeline must be run and completed at least once before enabling change control.
- If a manual job in GitLab is canceled, or timed out before completion, the corresponding change remains in **Open** state until the change request approval process is completed manually.
- Pipeline discovery is limited to the first 100 results using the project search filter. To discover additional pipelines, modify the search filter (which appears when you click **Discover**) to expand the results for the discovery request.
- Repositories or pipelines created or updated after the project has already been discovered and configured are tracked manually.
- GitLab Issues is not supported.
- JUnit test type integration is supported for GitLab. [Test tool integration](#) lets you view test results in DevOps for GitLab unit, functional, and performance tests.

### Get started

Use one of the following options to onboard GitLab. For a guided experience, use the workspace to onboard a tool. Alternatively, you can use the Service Catalog or Classic experience.

#### Onboard GitLab to DevOps Change Velocity – Workspace

Connect your GitLab instance using the playbook in DevOps Change workspace to discover repositories and pipeline.

#### Before you begin

Role required: sn\_devops.admin or sn\_devops.tool\_owner

## Procedure

1. Navigate to **Workspaces > DevOps Change Workspace** and use one of the following options to open the Playbook to onboard GitLab.

Option	Steps
Homepage	<ul style="list-style-type: none"> <li>a. Select the <b>Connect tool</b> widget</li> <li>b. From the Connect to a tool modal, select the tool from the appropriate category. For example, if you want to connect to GitLab as coding tool, you'd select GitLab under the <b>Code</b> category.</li> </ul>
Applications module	<ul style="list-style-type: none"> <li>a. Select <b>Applications ()</b>.</li> <li>b. Select an existing application, or create one. To create an application, see <a href="#">Create an application in DevOps Change Velocity</a>.</li> <li>c. From the# Recommended actions# pane, select the#<b>Connect a tool</b> card.</li> <li>d. From the Connect to a tool modal, select the tool from the appropriate category. For example, if you want to connect to GitLab as coding tool, you'd select GitLab under the <b>Code</b> category.</li> </ul>
Tools module	<ul style="list-style-type: none"> <li>a. Select <b>Tools ()</b>.</li> <li>b. From the Tools list, select the appropriate category. For example, if you want to connect to GitLab as coding tool, you'd select the <b>Coding tools</b> category.</li> <li>c. Select <b>Connect coding tool</b>.</li> <li>d. On the# Connect to a tool# modal, select <b>GitLab</b>.</li> </ul>

2. In the **Tool name** field, enter a name for the tool.

3. Select **Next**.

The DevOps playbook opens to help you complete the onboarding tasks.

4. Complete the connection and configuration using the playbook.

- a. Enter your GitLab instance URL.
- b. In the **Credential type** field, select one.
  - Basic Auth
  - OAuth 2.0
- c. Enter the credentials.

- Basic Auth: Enter the username and password/access token of your GitLab instance.

**i Note:** Only personal access token is supported. When you generate the token, select the scope API and grant read/write access including all the groups and projects, container registry, and package registry.

- OAuth 2.0: Enter your OAuth credential.

For more information on OAuth2.0 credentials, see [Setting up GitLab OAuth 2.0 credentials for DevOps](#).

- Optional: If your GitLab instance is attached to a MID Server, select the **MID Server** option and enter its details.

(Optional) For more information about MID server, see [MID Server selection](#)

- Select **Connect**.

Permission checks are run and the permission requirements are displayed in a pop-up modal.

## 5. Select **Continue** or **Connect Anyway**.

If the connection is successful, the tool is created in ServiceNow and connected to your GitLab instance. Projects, repositories, and pipelines are automatically discovered.

## 6. Specify the access for the tool.

- If you want to control access to the tool, add the groups that must be given access to the tool in the **Maintained by** field.

The tasks these users in the groups can perform depends on the role assigned to them.

- DevOps Tool Owner role: Can view and edit the tool.
- DevOps App Owner role: Can view the tool and can associate, discover, import historical data, and modify pipeline steps (if applicable) of the tool's objects (such as plans, repositories, and pipelines).
- DevOps Administrator role: Can edit all tools.
- Other DevOps roles: Can view the tool.

**i Note:** If you don't select a group and skip this step, all users with the DevOps Tool Owner role will be able to edit the tool.

- If you choose to control access to the tool, the **All App Owners can view and associate tool objects to applications** option becomes available for selection.

This option enables all users having the DevOps App Owner role to access the tool. If selected, they'll be able to view, associate, discover, import historical data, and modify pipeline steps (if applicable) of the tool's objects.

- Select **Assign**.

## 7. Select the items you want to configure from the list.

8. Optional: Select **Configure** the tool to send real-time data.  
You can also choose to enable nightly polling to retrieve data for any tracked pipelines by selecting the **Enable Polling** property option in the **Administration** module. See [DevOps Change Velocity Properties](#).
9. Select the repositories and pipeline that you want to track from the **Select repositories to track** and **Select pipelines** playbook activity respectively.

For each selected pipeline, all steps are imported for the last successful execution.

10. Optional: In the Assign services to pipeline steps activity, specify **Step type** and **Service** for each pipeline step.  
(Optional) Completing this step as part of tool onboarding enables the DevOps Insights dashboards to show more meaningful data immediately.
11. Click **Next** to review the details of the successfully connected GitLab tool.

From the Summary page, select **View tool record** to review the details of the connected instance.

## Result

You've successfully onboarded your GitLab tool to DevOps Change Velocity.

### Onboard GitLab to DevOps Change Velocity – Classic

Connect your GitLab instance to discover, configure, and import repositories and pipelines.

#### Before you begin

Complete the steps specified in the [Getting started](#) section before connecting to a tool.

Role required: sn\_devops.admin or sn\_devops.tool\_owner

#### About this task

- **Connect** by using your GitLab instance details.
- **Discover** repositories and pipelines.
- **Configure** webhooks in GitLab.
- **Import** branch and commit records, task execution and step execution records.

#### Procedure

1. Enter the GitLab instance details to connect to DevOps Change Velocity.
  - a. Navigate to **DevOps > Tools > Create New** and create a record.
  - b. Enter a **Tool Name** and fill in the tool details.

---

Tool URL

GitLab tool URL.

For example:

<https://gitlab.com>

Tool Username	GitLab username
Tool Password / Access Token	GitLab access token
	<b>i Note:</b> Only personal access token is supported. When you generate the token, select the scope api and grant read/write access including all groups and projects, the container registry, and the package registry.

**MID Server** is optional. Select **MID Server** for an on-premises tool that is attached to a MID Server. The **Application** value is automatically set to DevOps and the **Capability** value is set to REST.

c. Click **Submit**.

The tool is connected successfully.

On successful tool creation, you're taken to the tool record page.

2. If you want to control access to the tool, add the groups that must be given access to the tool in the **Maintained by** field on the **Access** tab.

The tasks the users in the groups can perform depends on the roles assigned to them.

- DevOps Tool Owner role: Can view and edit the tool.
- DevOps App Owner role: Can view the tool and can associate, discover, import historical data, and modify pipeline steps (if applicable) of the tool's objects (such as plans, repositories, and pipelines).
- DevOps Administrator role: Can edit all tools.
- Other DevOps roles: Can view the tool.

**i Note:** Only groups containing users with DevOps roles are available for selection in the **Maintained by** field.

The **All App Owners can view and associate tool objects to applications** option becomes available for selection if you choose to restrict access to the tool. This option enables all users having the DevOps App Owner role to access the tool. If selected, they'll be able to view, associate, discover, import historical data, and modify pipeline steps (if applicable) of the tool's objects.

3. Click **Discover** to discover existing repositories and pipelines, and fill in the fields in the project Search Filter window.

**i Note:** GitLab repos and pipelines discovered are determined by these filter conditions. To discover additional repos, modify the project filter to expand the results.

Owned by me (recommended)	Searches for the repositories in the project that the current user owns.
Currently member of	Searches for the repositories in the project that the current user is a member of.

Search

Searches for the repositories in projects with the specified text string.

**Note:** GitLab repositories and pipelines discovered are also determined by the credentials (level of access) configured in the connection.

Records are added to the corresponding related lists.

4. Automatically configure the webhook URL in a GitLab repository to send notifications to the DevOps tool by selecting **Configure**. Alternatively, you can choose to enable nightly polling to fetch data system for any tracked repositories or pipelines by setting the **Enable Polling** property to **Yes**.
5. For discovered repositories, import historical data for the tool, and associate the repository with an app.

- a. Open the repository record from the Repositories related list and click **Import**. Imported branch records and commit records from the repository are added to the corresponding related lists.
- b. In the **App** field, click the lookup list and select an App record to associate with the repository, or click **New** to create one.

Imported historical data records are added to the corresponding related lists.

### Setting up GitLab OAuth 2.0 credentials for DevOps

Integrate your GitLab account with your ServiceNow instance by creating a custom OAuth application in GitLab and authenticating requests from ServiceNow DevOps.

Configure your GitLab account, register GitLab in the application registry, and create a credential record for the GitLab App provider.

Role required: oauth\_admin.

### Configure the GitLab App in your GitLab account (Authorization Code)

Create a custom GitLab App from your GitLab account to enable OAuth 2.0 authentication with your ServiceNow instance.

#### Before you begin

Role required: admin

#### About this task

Configure GitLab as an OAuth 2.0 authentication identity provider. For more information, see [GitLab documentation](#).

#### Procedure

1. From your GitLab account, create your App by navigating to **Edit Profile > Applications**.
2. In the **Add new application** form, specify a **Name**, and in **Redirect URI** field, enter `https://<instance-name>.service-now.com/oauth_redirect.do`.
3. In the **Scopes** section, ensure that you select the **api** check box.
4. Leave the remaining fields empty (default).
5. Click **Save application**.

The application is created. You can open the application to access the Application ID, Secret key and Callback URL.

6. Install the newly created GitLab App on the accounts of your choice.

### Register GitLab as an OAuth Provider (Authorization Code)

Use the information generated during GitLab App account configuration to register GitLab as an OAuth provider and allow the instance to request OAuth 2.0 tokens.

#### Before you begin

Role required: admin

#### Procedure

1. Navigate to **System OAuth > Application Registry**.
2. Click **New**.  
The system displays the message **What kind of OAuth application?**
3. Select **Connect to a third party OAuth Provider**.  
The system displays an empty Application Registries form.
4. Complete the form.

Field	Value required
Name	Enter any name to uniquely identify the record. For example, enter <code>My GitLab App Provider</code> .
Client ID	Enter the Application ID of your GitLab App.
Client Secret	Enter the secret key of your GitLab App .
OAuth API script	Select <b>OAuthDevOps GitLab Handler</b> .
Default Grant type	Select <b>Authorization Code</b> .
Authorization URL	Enter <code>https://gitlab.com/oauth/authorize</code> . For an on-premise deployment, use the proper GitLab host URL.
Token URL	Enter <code>https://gitlab.com/oauth/token</code> . For an on-premise deployment, use the proper GitLab host URL.

5. Leave the rest of the form fields as default.
6. Right-click the form header, and click **Save**.

- The system validates the OAuth credentials and populates the **Redirect URL** (Hint: It should match the **Redirect URI** value previously provided in your GitLab App configuration).
- The system populates **OAuth Entity Profile** with **Grant Type** as **Authorization Code**. For example, **OAuth Entity Profile** is created with default **Name, My GitLab App Provider default\_profile**

7. Validate that the OAuth Entity Scope related list contains the **api** scope.

#### Create a credential record for GitLab App provider (Authorization Code)

Create a credential record to the GitLab App provider previously created to authorize actions.

#### Before you begin

Role required: admin

#### Procedure

1. Navigate to **Connections & Credentials > Credentials**.
2. Click **New**.  
The system displays the message **What type of Credentials would you like to create?**.
3. Select **OAuth 2.0 Credentials**.  
The pop-up window displays an empty OAuth 2.0 Credentials form.
4. Fill in these values.

Field	Value required
Name	Enter any name to uniquely identify the record. For example, enter <b>My GitLab App Credential</b> .
Active	Enable
OAuth Entity Profile	Select the default OAuth Entity profile you created previously.
Applies to	Select the MID Servers that can use this credential. For example, select <b>All MID Servers</b> .
Order	Select the order to apply this credential. For example, enter 100.

5. Save the record.
6. Click the **Get OAuth Token** related link to generate the OAuth token.  
A successful token generation indicates that you can now authenticate connection between ServiceNow DevOps and GitLab via OAuth.

#### Model a GitLab basic CI pipeline in DevOps

Model a GitLab basic CI pipeline by mapping the pipeline to an app, and mapping DevOps pipeline steps to GitLab pipeline jobs.

#### Before you begin

Role required: sn\_devops.admin

## Procedure

1. Map your pipeline to an app.

a. Navigate to **DevOps > Apps & Pipelines > Apps** and open the application record to associate with the pipeline.

b. In the Pipelines related list, click **Edit...** to select a pipeline to associate with the app, or click **New** to create the pipeline.

**Note:** While associating a pipeline with an app, the pipeline steps are also fetched during import.

For a new pipeline, fill in the **Orchestration pipeline** field using the group name, subgroup name (if applicable), and project name as specified in GitLab.

For example, `My Group/My SubGroup/My Project`.

If a project is not under a group, simply specify `My Project`.

c. Click **Submit**.

2. Open the pipeline record again and create DevOps steps to map to each GitLab pipeline job so an orchestration task can be created.

Steps can be created in one of the following ways.

- Automatically create and map pipeline steps in DevOps by running your GitLab pipeline.

Pipeline steps are automatically created, mapped, and associated when DevOps receives step notifications from your GitLab pipeline during the run.

- Manually create and map each pipeline step to a GitLab pipeline job.

In the Steps related list, click **New** to create a DevOps step for each GitLab pipeline job (**Orchestration stage** field).

**Note:** The **Orchestration stage** field value of each step is case-sensitive and must match the original name of the corresponding GitLab pipeline job.

Name	Name of the pipeline step.
Pipeline	Pipeline in which the step is configured.
Type	<p>Pipeline step type.</p> <ul style="list-style-type: none"> <li>▪ Build and Test</li> <li>▪ Test</li> <li>▪ Deploy</li> <li>▪ Deploy and Test</li> <li>▪ Manual</li> <li>▪ Prod Deploy</li> </ul>
Order	Order in which the steps are run.

	<p><b>Note:</b> The step order determines the order of the cards in the Pipeline UI.</p> <p>The order of the cards in the Pipeline UI is by task execution.</p>
Orchestration stage	<p>GitLab pipeline job name (case-sensitive).</p> <p><b>Note:</b> For step association with GitLab CI pipeline jobs, the <b>Orchestration stage</b> field must be configured.</p>
Business service	Configuration service that applies to the step.

Once orchestration tasks are created, associate each orchestration task in the Orchestration Tasks related list with a DevOps pipeline step.

3. Optional: Select the **Change control** check box in a step to enable [change acceleration](#) and the corresponding configuration fields.

**Note:** The [Change Management](#) feature must be installed for change acceleration.

Change receipt	<p>(Optional) Select to enable change receipt for the step so the pipeline doesn't pause when a change request is created.</p> <p>All pipeline data is included in the change, but approval is not required for the pipeline to proceed.</p>
Change approval group	<p>(Optional) Approval group for the change request.</p> <p>The change approval group becomes the <b>Assignment group</b> in the DevOps change request.</p> <p><b>Note:</b> Ensure that the selected group has members and a group manager so approver field is not empty.</p>
Change type	<p>Change request type to create.</p> <ul style="list-style-type: none"> <li>◦ Normal (default)</li> <li>◦ Standard</li> <li>◦ Emergency</li> </ul>
(Optional) Template	(Optional) List of templates to use to auto populate fields for Normal or Emergency change requests.

<b>i Note:</b> This field is shown only when <b>Change type</b> is Normal or Emergency.	Select a template or create a new one.
(Optional) Standard change template  <b>i Note:</b> This field is shown only when <b>Change type</b> is Standard.	List of standard change templates to use for Standard change requests.  <b>i Note:</b> This field is required for Standard change type.
Change controlled branches	(Optional) (Multibranch only) Comma-separated list of branches under change control. Wildcards are supported.

You can set up change control in GitLab for manual GitLab jobs.

**i Note:** A pipeline must be run and completed at least once before enabling change control.

#### Example:

##### DevOps pipeline

#### What to do next

[Configuring the GitLab pipeline for DevOps](#)

#### Configuring the GitLab pipeline for DevOps

Webhooks are required in GitLab to send job and push notifications to DevOps. Change control can be configured in GitLab for a manual job.

#### Configure webhooks in GitLab manually

Configure webhooks in GitLab manually to send job and push notifications to the DevOps application if you did not select automatic configuration at the time of onboarding.

#### Before you begin

Verify that you have copied the **Webhook URL** field value from the DevOps GitLab tool form into your clipboard. The **Webhook URL** field is not available by default on the tool record. To add the field, navigate to **Additional actions () > Configure > Form Layout** in the tool record.

Role required: GitLab admin

#### About this task

Create webhooks for these two triggers in GitLab for every project you want to track.

- Push events
- Job events

In GitLab, navigate to **Project > Settings > Webhooks** to create webhooks.

## Procedure

1. Create a webhook for push events in GitLab.

- a. Paste the DevOps GitLab tool webhook in the **URL** field, and modify the URL to specify only the `code` parameter.

That is, replace `{code | plan | artifact | orchestration | test}` with `code`.

For example:

```
http://<your-instance>.service-now.com/api/sn_devops/v2/devops/tool/
code?toolId={sys_id of the GitLab tool record}.
```

- b. Enter a value in the **Secret Token** field.

**Note:** You can obtain the secret token from your ServiceNow admin.

- c. Select the **Push events** check box.

2. Create a webhook for job events in GitLab.

- a. For job events, paste the DevOps GitLab tool webhook in the **URL** field, and modify the URL to specify only the `orchestration` parameter.

That is, replace `{code | plan | artifact | orchestration | test}` with `orchestration`.

For example:

```
http://<your-instance>.service-now.com/api/sn_devops/v2/devops/tool/
orchestration?toolId={sys_id of the GitLab tool record}.
```

- b. Enter a value in the **Secret Token** field.

**Note:** You can obtain the secret token from your ServiceNow admin.

- c. Select the **Job events** check box.

## Example:

### GitLab webhooks for DevOps integration

#### Change acceleration in GitLab

Change acceleration is supported in DevOps for manual GitLab jobs.

The GitLab job under change control must have these instructions for the pipeline execution to be resumed or canceled via the change request:

- `when: manual`
- `allow_failure: false`

For example:

```

deploy:
  stage: deploy
  tags:
    - local-runner1
  when: manual
  allow_failure: false
  script:
    - echo 'Deploy'

```

Refer to the [CI/CD pipeline configuration reference](#)  for more information on how to configure a GitLab job.

Additional considerations:

- If `allow_failure` is set to `true`, the pipeline continues even when the change is rejected.
- A user with the appropriate role access in GitLab can unblock and continue a pipeline regardless of the change request state.

#### GitLab change acceleration behavior

Manual execution	Change acceleration in step	Change request approved	Result
Yes	Yes	N/A	If the manual job is under change control, the change is automatically created.
		Yes	The manual job is automatically executed.
		No	The manual job is automatically rejected/failed.
	No	N/A	The manual job waits for manual intervention from the pipeline owner via the GitLab UI (default behavior).
No	Yes	N/A	The change request is not created.

**Note:** Parallel jobs are displayed sequentially, based on the order in which the jobs are queued for execution.

#### Additional information - GitLab

Additional information about GitLab such as bulk commits in GitLab.

## Bulk commits in GitLab

Bulk commits are supported with GitLab.

To support large commits, perform these actions:

- Install the ServiceNow IntegrationHub Action Template - Data Stream (com.glide.hub.action\_type.datastream) plugin.
- For optimal performance, disable flow logging by setting the Flow Designer com.snc.process\_flow.reporting.level property to **Off**.
- For MID Server settings, view the [MID Server support for Data Stream actions](#) section.

GitLab code push webhook sends a maximum of 20 commits in a notification. If the number of commits in the push are less than 20, a single inbound event is created and processed in the ServiceNow instance.

If the number of pushed commits are equal or greater than 20, multiple inbound events are created from the original event with each new event containing a batch of 19 commits. The original inbound event is marked as ignored.

Currently, GitLab Data stream action can process up to 10,000 commits in a single push.

## Jenkins integration with DevOps Change Velocity

Connect your Jenkins instance to discover pipeline definitions and configure real-time notifications or polling to enable change traceability and automation.

### Overview

DevOps Change Velocity supports Orchestration capability for the Jenkins tool and various kinds of pipeline types like Single Pipeline, Multibranch Pipeline, Folder, and Freestyle project types. For Folder, three levels are supported by default. Integrating Jenkins with DevOps Change Velocity enables you retrieve the data from your Jenkins pipeline for end-to-end traceability and policy creation for change automation.

### Get started

Onboard Jenkins to DevOps Change velocity

Connect your Jenkins instance to DevOps change velocity to retrieve data like artifacts, test results, scan results, etc. from your Jenkins pipelines. We recommend that you onboard using Workspace [Onboard Jenkins to DevOps Change Velocity – Workspace](#) for an intuitive experience.

Setting up Jenkins plugin to send real-time data

Automatically configure Jenkins to send real time notifications for your pipeline executions to this ServiceNow DevOps Change Velocity. Real time notifications are ideal to maintain the most up to date information particularly for automating change requests.

**1.** Install Jenkins plugin for ServiceNow DevOps Change Velocity. You must install the Jenkins plugin for DevOps Change Velocity to capture data that is coming from Jenkins in two ways:

- Download from the [ServiceNow Store](#).
- Install from the Jenkins Marketplace. Navigate to **Manage Jenkins > Manage Plugins**, search for the ServiceNow DevOps plugin and install.

**2.** Enable the ServiceNow DevOps plugin by navigating to **Manage Jenkins > Configure System**.

**3.** Configure the plugin and test the connection.

**a.** In Jenkins, navigate to **Manage Jenkins > Configure System**, select the **ServiceNow DevOps Enabled** check box in the ServiceNow DevOps Configuration section, and fill in the fields. When enabled, Jenkins starts sending events to DevOps as inbound events.

 **Note:** Values are case-sensitive.

Field	URL format/value
Instance URL	<code>https://&lt;your-instance&gt;.service-now.com</code>
API Version	v1
Orchestration Tool ID	The <code>sys_id</code> of the orchestration tool.  You can copy this value from the webhook URL ( <code>toolid:value</code> ), or obtain it directly using the <code>Copy sys_id</code> command on the Orchestration Tool form.
Artifact Tool ID	The <code>sys_id</code> of the artifact tool.
Credentials	To add credentials, click <b>Add</b> .  Username: Enter your DevOps integration account. By default, it will be your <code>devops.integration.user</code> .  Password: Password of the integration user that was set by the DevOps admin.
Log Level	The level of log messages you want to store in Jenkins logs/ ServiceNow log recorder.  Select from one of the following options: <ul style="list-style-type: none"><li>▪ inherit</li><li>▪ off</li><li>▪ severe</li><li>▪ info</li><li>▪ config</li><li>▪ fine</li></ul>

Field	URL format/value
	<ul style="list-style-type: none"> <li>▪ finer</li> <li>▪ finest</li> <li>▪ all</li> </ul> <p>For more information on Log levels and log recorders, see <a href="#">Jenkins log levels and Log Recorders</a></p>
Force Tracking Check	<ul style="list-style-type: none"> <li>▪ Select the check box to make a REST (POST) API call to Jenkins for each pipeline execution to determine whether the pipeline is tracked or not.</li> <li>▪ Clear the check box to store details in the <code>snPipelineInfo.json</code> file and stop making API calls for every pipeline execution.</li> </ul>
Pull Request Pipeline Tracking check	

- b.** Click **Test Connection**. Verify that the connection successful message displays.

**Note:** Jenkins also supports testing capabilities with JUnit. [Test tool integration](#) lets you view test results in DevOps for Jenkins unit, functional, and performance tests.

Use one of the following options to onboard Jenkins. For a guided experience, use the workspace to onboard a tool. Alternatively, you can use the Service Catalog or Classic experience.

### Onboard Jenkins to DevOps Change Velocity – Workspace

Connect your Jenkins instance using the playbook in DevOps Change workspace to discover pipelines.

#### Before you begin

Complete the tasks in [Getting started with DevOps Change Velocity](#).

Role required: sn\_devops.admin or sn\_devops.tool\_owner

#### Procedure

1. Navigate to **Workspaces > DevOps Change Workspace** and use one of the following options to open the playbook to onboard Jenkins.  
You can connect a tool from an application if you also know which specific pipelines should be associated and tracked under that application to streamline the setup. This will allow you to easily associate and import data from Jenkins as well.

Option	Steps
Homepage	<ol style="list-style-type: none"> <li>Select the <b>Connect tool</b> widget</li> <li>From the Connect to a tool modal, select the tool from the appropriate category.</li> </ol>

Option	Steps
Applications module	<p>For example, if you want to connect to Jenkins as an orchestration tool, you'd select Jenkins under the <b>Orchestration</b> category.</p> <ul style="list-style-type: none"> <li>a. Select <b>Applications ()</b>.</li> <li>b. Select an existing application, or create one. To create an application, see <a href="#">Create an application in DevOps Change Velocity</a>.</li> <li>c. From the# Recommended actions# pane, select the<b>Connect a tool</b> card.</li> <li>d. From the Connect to a tool modal, select the tool from the appropriate category. For example, if you want to connect to Jenkins as an orchestration tool, you'd select Jenkins under the <b>Orchestration</b> category.</li> </ul>
Tools module	<ul style="list-style-type: none"> <li>a. Select <b>Tools ()</b>.</li> <li>b. From the Tools list, select the appropriate category. For example, if you want to connect to Jenkins as an orchestration tool, you'd select the <b>Orchestration tools</b> category.</li> <li>c. Select <b>Connect orchestration tool</b>.</li> <li>d. On the# Connect to a tool# modal, select <b>Jenkins</b>.</li> </ul>

2. In the **Tool name** field, enter a name for the tool.

3. On the **Jenkins instance details** playbook activity:

- a. Enter the URL of your Jenkins instance.
- b. Enter the login credentials (password or access token or API token) of the Jenkins instance.  
To generate API token, see [Connect to Jenkins using API token authentication](#).
- c. If your Jenkins instance is attached to a MID Server, select the MID Server option and enter its details.  
A MID server is required if your tool instance is hosted on-prem. For more information about MID server, see [MID Server selection](#) ↗

4. Select **Connect**.

The Permission check is run and the permission requirements are displayed in a pop-up.

5. Select **Continue** or **Connect anyway**.

6. Specify the access for the tool.

- a. If you want to control access to the tool, add the groups that must be given access to the tool in the **Maintained by** field.

The tasks these users in the groups can perform depends on the role assigned to them.

- DevOps Tool Owner role: Can view and edit the tool.
- DevOps App Owner role: Can view the tool and can associate, discover, import historical data, and modify pipeline steps (if applicable) of the tool's objects (such as plans, repositories, and pipelines).
- DevOps Administrator role: Can edit all tools.
- Other DevOps roles: Can view the tool.

**i Note:** If you don't select a group and skip this step, all users with the DevOps Tool Owner role will be able to edit the tool.

- b. If you choose to control access to the tool, the **All App Owners can view and associate tool objects to applications** option becomes available for selection.

This option enables all users having the DevOps App Owner role to access the tool. If selected, they'll be able to view, associate, discover, import historical data, and modify pipeline steps (if applicable) of the tool's objects.

- c. Select **Assign**.

7. Configure the Jenkins instance to send data by performing the steps specified in the [Jenkins integration with DevOps Change Velocity](#) topic.

You can also choose to enable nightly polling to retrieve data for any tracked pipelines by selecting the **Enable Polling** property option in the **Administration** module. See [DevOps Change Velocity Properties](#).

8. Select the pipelines that you want to track from **Select pipelines playbook** activity.  
For each selected pipeline, all steps are imported for the last successful execution.

9. Optional: In the Assign services to pipeline steps activity, specify **Step type** and **Service** for each pipeline step.

(Optional)

Completing this step as part of tool onboarding enables the DevOps Insights dashboards to show more meaningful data immediately.

10. Select **Next** and from the **Summary** page, select **View tool record** to review the details of the connected Jenkins tool.

**i Note:** If your tool credential has changed, you must update the credentials in your ServiceNow instance. For more information, see [Update third-party tool credentials in DevOps Change Velocity](#).

**Result**

You've successfully onboarded your Jenkins tool to DevOps Change Velocity.

**Onboard Jenkins to DevOps Change Velocity — Service Catalog**

Connect your Jenkins instance using the ServiceNow Service Catalog to retrieve data like artifacts, test results, scan results, and so on.

**Before you begin**

Role required: sn\_devops.admin or sn\_devops.tool\_owner

**Procedure**

1. Navigate to **All > Service Catalog > Catalog Definitions > Maintain Items** and search for DevOps.

**Note:** You can also access the service catalog from Employee Center or Service portal.

2. From the DevOps catalog items, select and activate **DevOps App Onboarding** and **DevOps Tool Onboarding**.
3. After activating, select **DevOps Tool Onboarding** and select **Try it**.
4. In the DevOps Tool Onboarding form, enter the tool details:

Field	Description
Tool name	Name for your Jenkins integration.
Tool integration	Select Jenkins.
Tool URL	URL for your Jenkins instance.
Tool username	Username for your Jenkins instance.
Tool password/ Access token	Access credentials for your Jenkins instance.
Use MidServer	Optional. Select#Mid Server# for an on-premises tool that is attached to a#Mid Server. Application is automatically set to #DevOps and capability is set to REST.

5. Select **Order Now**.

A request is created. When the request is approved, the tool is connected and discovered.

6. From the DevOps catalog items, select **DevOps App Onboarding**.

7. In the DevOps App Onboarding form, enter the details:

Are you creating a new app or adding to an existing one?	Select from the options whether to create a new app or use an existing app.
App	Enter the name for the app that you're creating or using.
Onboarding pipelines	Enter the connected Jenkins tool name.

Pipelines	Select the pipelines for which you want to configure webhooks and import historical data.
Import from and Import to	Select the dates for which you want to import the data. By default, the last 30 days are selected. You can choose to import data for a maximum of 90 days.
Onboarding repositories	Leave empty.
Onboarding plans	Leave empty.

#### 8. Select Order Now.

A request is created. When the request is approved, the pipeline objects are associated to the app record, which enables real-time tracking. Historical data is also imported for the selected pipelines. The **Track** field is automatically enabled for imported pipelines.

### Onboard Jenkins to DevOps Change Velocity – Classic

Connect your Jenkins instance to discover and track your pipelines, and import your orchestration tasks for end to end traceability and change automation.

#### Before you begin

Role required: sn\_devops.admin or sn\_devops.tool\_owner

#### About this task

**i Note:** You follow the procedure described in this section after you have used the playbook pages to perform initial setup.

Actions:

- **Connect** to Jenkins and get the webhook URL when you submit a DevOps tool record.
- **Discover** orchestration tasks and pipelines.
- **Import** task execution and step execution records.

**i Note:** You can authenticate your connection with Jenkins using Jenkins API tokens. For more information, see [Connect to Jenkins using API token authentication](#).

#### Procedure

1. Create a tool record in DevOps to automatically connect to Jenkins and get the webhook URL.
  - a. Navigate to **DevOps > Tools > Create New** and create a record.
  - b. Enter a **Tool Name** and fill in the tool details.

Tool Integration	Jenkins
Tool URL	<p>Jenkins tool URL</p> <p>For example:</p> <p><code>https://jenkins.com</code></p>

Tool Username	Jenkins user name
Tool Password / Access Token	Jenkins password, access token or the API token you generate.
	<p><b>i Note:</b> To generate API token, see <a href="#">Connect to Jenkins using API token authentication</a>.</p>

- c. Select MID Server for an on-premises tool that is attached to a MID Server. The Application value is automatically set to DevOps and the Capability value is set to REST.
- d. Click **Submit**.  
The tool is connected successfully.

On successful tool creation, you're taken to the tool record page.

- 2. If you want to control access to the tool, add the groups that must be given access to the tool in the **Maintained by** field on the **Access** tab.

The tasks the users in the groups can perform depends on the roles assigned to them.

- DevOps Tool Owner role: Can view and edit the tool.
- DevOps App Owner role: Can view the tool and can associate, discover, import historical data, and modify pipeline steps (if applicable) of the tool's objects (such as plans, repositories, and pipelines).
- DevOps Administrator role: Can edit all tools.
- Other DevOps roles: Can view the tool.

**i Note:** Only groups containing users with DevOps roles are available for selection in the **Maintained by** field.

The **All App Owners can view and associate tool objects to applications** option becomes available for selection if you choose to restrict access to the tool. This option enables all users having the DevOps App Owner role to access the tool. If selected, they'll be able to view, associate, discover, import historical data, and modify pipeline steps (if applicable) of the tool's objects.

- 3. Click **Discover** to discover the existing orchestration tasks (Jenkins stages) and pipelines.

**i Note:** Orchestration tasks and pipelines are discovered for folders nested to the level specified in the `sn_devops.discover.folder.depth` property. For more information, see [Properties installed with DevOps](#).

Records are added to the corresponding related lists.

- 4. Open a discovered record from the Orchestration Tasks related list and click the **Import** related link to import historical data from the orchestration task.  
Imported task execution records and step execution records are added to the corresponding related lists.

### Model a Jenkins pipeline in DevOps

Model a Jenkins pipeline by mapping the pipeline to an app, and mapping DevOps pipeline steps to Jenkins stages.

## Before you begin

The Jenkins plugin for ServiceNow DevOps is provided to enable change acceleration so your orchestration tool can communicate with ServiceNow DevOps and control certain aspects of pipeline executions.

Visit the Ancillary Software section on the [ServiceNow Store](#) website to download the Jenkins plugin for ServiceNow DevOps.

Role required: sn\_devops.admin

## About this task

You follow the procedure described in this section after you have used the playbook pages to perform initial setup.

Both [scripted pipelines](#) (Jenkinsfile) and freestyle jobs are supported.

For Jenkinsfile pipelines, pipeline steps are created, mapped, and associated to orchestration tasks automatically, instead of manually.

## Procedure

**1.** Map your pipeline to an app in DevOps.

**a.** Navigate to **DevOps > Apps & Pipelines > Apps** and open the application record to associate with the pipeline.

**b.** In the Pipelines related list, click **Edit...** to select a pipeline to associate with the app, or click **New** to create the pipeline.

For a new pipeline, fill in the **Orchestration pipeline** field using the full project name as specified in Jenkins.

**i Note:** While associating a pipeline with an app, the pipeline steps are also fetched during import.

**c.** Click **Submit**.

**2.** Open the pipeline record again and select the **Track** check box so events from the pipeline are received.

**i Note:** The **Track** check box must be selected to integrate the pipeline with DevOps.

**3.** Create DevOps steps automatically or manually to map to each Jenkins pipeline stage so an orchestration task is created.

- For declarative or scripted pipelines, run your Jenkins pipeline to automatically create and map pipeline steps in DevOps.

Pipeline steps are automatically created, mapped, and associated when DevOps receives step notifications from your Jenkins pipeline during the run.

- For freestyle jobs, manually create and map each pipeline step to a Jenkins pipeline job.

In the Steps related list, click **New** to create a DevOps step for each Jenkins pipeline stage (**Orchestration stage** field)

**i Note:** The **Orchestration stage** field value of each step is case-sensitive and must match the original name of the corresponding Jenkins pipeline stage.

Name	Name of the pipeline step.
Pipeline	Pipeline in which the step is configured.
Type	<p>Pipeline step type.</p> <ul style="list-style-type: none"> <li>▪ Build and Test</li> <li>▪ Test</li> <li>▪ Deploy</li> <li>▪ Deploy and Test</li> <li>▪ Manual</li> <li>▪ Prod Deploy</li> </ul>
Order	<p>Order in which the steps are run.</p> <p><b>i Note:</b> The step order determines the order of the cards in the <a href="#">Pipeline UI</a>.</p> <p>The order of the cards in the Pipeline UI is by task execution.</p>
Orchestration stage	<p>Jenkins pipeline stage name (case-sensitive).</p> <p><b>i Note:</b> For step association with Jenkins pipeline stages, the <b>Orchestration stage</b> field must be configured.</p>
Business service	Configuration service that applies to the step.

Once orchestration tasks are created, associate each orchestration task in the Orchestration Tasks related list with a DevOps pipeline step.

#### 4. Optional: Enable change control automatically or manually based on the type of pipeline.

- For declarative or scripted pipelines, if you have used the snDevOpsChange script in your pipeline, run your Jenkins pipeline to automatically enable change control. You can also enable change control manually by following the steps provided for freestyle jobs.
- For freestyle jobs, select the **Change control** check box in a step to enable [change acceleration](#) and the corresponding configuration fields.

**i Note:** ServiceNow [Change Management](#) must be installed for change acceleration.

---

Change receipt

(Optional) Select to enable change receipt for the step so the pipeline doesn't pause when a change request is created.

	All pipeline data is included in the change, but approval is not required for the pipeline to proceed.
Change approval group	(Optional) Approval group for the change request.  The change approval group becomes the <b>Assignment group</b> in the DevOps change request.  <b>i Note:</b> Ensure that the selected group has members and a group manager so the approver field is not empty.
Change type	Change request type to create. <ul style="list-style-type: none"><li>▪ Normal (default)</li><li>▪ Standard</li><li>▪ Emergency</li></ul>
(Optional) Template  <b>i Note:</b> This field is shown only when <b>Change type</b> is Normal or Emergency.	(Optional) List of templates to use to auto populate fields for Normal or Emergency change requests.  Select a template or create a new one.
(Optional) Standard change template  <b>i Note:</b> This field is shown only when <b>Change type</b> is Standard.	List of standard change templates to use for Standard change requests.  <b>i Note:</b> This field is required for Standard change type.
Change controlled branches	(Optional) (Multibranch only) Comma-separated list of branches under change control. Wildcards are supported.

5. Navigate to **DevOps > Tools > Orchestration Tools** and in the Jenkins tool record, copy the DevOps **Webhook URL** field value.

The webhook URL contains the DevOps location for Jenkins to send messages, including the sys\_id for the tool:

```
https://<devops.integration.user>:<password>@<your-instance>.service-now.com/api/sn_devops/v1/devops/tool/event/{sys_id of the record}
```

### Example:

**DevOps app**

**DevOps pipeline**

## DevOps pipeline step

### Additional information - Jenkins

Additional information on Jenkins.

#### Connect to Jenkins using API token authentication

Connect to Jenkins using API token authentication instead of user name and password.

#### Before you begin

Role required:

- Jenkins: admin or any user with overall Read and Job Read roles

#### About this task

You can have multiple active API tokens at the same time and track the usage of your tokens. You can also revoke API tokens as needed. You can also use the Jenkins API token for authentication when you're using the Jenkins CLI..

#### Procedure

In the Jenkins banner frame, click your user name to open the user menu.

- a. Navigate to **Your Username > Configure > API Token**.
- b. Click **Add new Token**.
- c. Click **Generate**.
- d. Copy the API token that is generated

#### Note:

- Regenerate the tokens every 6 months (depending on your context). Jenkins displays an indicator concerning the age of the token
- Use a different token for each application so that if an application is compromised you can revoke its token individually.
- If your token expires, regenerate the token and update it in the Now Platform instance
- ServiceNow DevOps DevOps does not support using Legacy API Tokens because Jenkins does not recommend the use of Legacy API Token. For more information, see the [Jenkins blog post](#).

#### Reduce calls from Jenkins to ServiceNow DevOps to fetch pipeline information

Enable the Force Tracking Check field in the Jenkins configuration form to create a pipeline tracking file in Jenkins. ServiceNow DevOps makes a REST call to Jenkins to update the tracking file when the Track field in a pipeline is modified.

#### Force Tracking Check

The ServiceNow DevOps configuration section in Jenkins includes a **Force Tracking Check** check box to reduce the number of calls made from Jenkins to DevOps to fetch pipeline information such as pipelines being tracked. Base-system flows:

- DevOps Jenkins File Update- Track flow
- DevOps Jenkins File Update- Test Info flow

## How it works

In previous versions, a REST call fetched the pipeline information for every Jenkins build triggered. If you had multiple pipelines in your Jenkins environment and were tracking only a few of them, this meant that a call was made to fetch the tracking information for each pipeline even if you were tracking a few of them.

The first time you trigger a Jenkins build or pipeline execution, Jenkins makes a pipeline information API call and creates `snPipelineInfo.json` file in `/ {JENKINS_HOME} /jobs/{jobName}` directory. For each subsequent pipeline execution Jenkins checks the information available in the `snPipelineInfo.json` file before making a pipeline info API call.

If you disable the **Force Track Check** check box:

- The DevOps Jenkins File Update- Track flow triggers when you update the **Track** field on the pipeline form. The **Track** field info is updated in the `snPipelineInfo.json` file.
- DevOps Jenkins File Update- Test Info flow triggers when you update Test type mapping for Jenkins tool integration and verify that Test info is updated in `snPipelineInfo.json` file.

If you enable the **Force Track Check** check box, Jenkins makes pipeline info API calls to DevOps even if track/test information exists in the `snPipelineInfo.json` file.

## Using a declarative or scripted pipeline in DevOps

When you use a Jenkinsfile, steps are created, mapped, and associated to orchestration tasks automatically instead of manually.

Jenkinsfile is a text file that contains the definition of a Jenkins pipeline and is checked into source control.

Each root-level stage configured in the Jenkinsfile is discovered as a separate orchestration task in DevOps that is mapped to an individual step.

- Note:** The **Track** field for the pipeline must be set to **True** in DevOps to receive job notifications from Jenkins.

## DevOps Jenkinsfile commands

- Note:** The `snDevOpsStep` command is not required to set up Jenkins pipeline steps because the steps are created automatically when the pipeline is run.

- `snDevOpsStep(enabled:{true/false}, ignoreErrors:{true/false})`

Where:

- `enabled` specifies the setting for build notifications property (true/false)
- `ignoreErrors` specifies the setting to prevent job failure if there is an error (true/false)

- Note:** Step properties can also be set using check boxes in Jenkins (job configuration), which take precedence over the values specified here.

- `snDevOpsChange(ignoreErrors:{true/false}, changeRequestDetails:{setCloseCode:{true/false}, attributes:})`

Where `ignoreErrors` specifies the setting to prevent job failure if there is an error (true/false)

Where `changeRequestDetails` specifies closure code and change request fields from within the pipeline

Enables change control for each root-level stage that is mapped to a DevOps step.

- `snDevOpsArtifact`

Registers artifacts when configuring [DevOps change acceleration for releases](#).

- `snDevOpsPackage`

Creates a package for artifacts when configuring [DevOps change acceleration for releases](#).

**i Note:** Stage mapping is only supported for stages at the root level, not nested or parallel stages.

## Example Jenkinsfile

```
node {
    stage("build") {
        snDevOpsStep(enabled:false)
    }
    .
    .
    .
    stage("test") {
        snDevOpsStep(ignoreErrors:false)
    }
    .
    .
    .
    stage("deploy") {
        snDevOpsStep()
        snDevOpsChange()
    }
    .
    .
    .
}
```

**i Note:** Historical import of a declarative or scripted pipeline is not supported.

## Jenkins snippet generator for DevOps

You can use the Jenkins Snippet Generator utility to generate a template code for the orchestration tasks for scripted pipelines. You can use the snippet generator utility to create template for the following orchestration tasks.

- SnDevOpsArtifact
- SnDevOpsChange
- SnDevOpsPackage

To generate a step snippet, navigate to the Pipeline Syntax from a configured pipeline, select the step from the **Sample Step** list, and update the values for different variables in the step. Select the **ignore error** option to prevent job failure if there is an error. Select **Generate Pipeline Script** to create a snippet. You can copy and paste the snippet into a pipeline.

## Parallel and sub-stage support

When a stage (or set of parallel stages) is nested within a pipeline stage defined by the `snDevOpsStep()` command, these rules apply:

- Any action from the nested stage is processed as part of the parent root-level stage
- Only one change request is created (at the parent root level) even if multiple stages nested under the parent root-level stage trigger a change
- Orchestration tasks created are always associated with the parent root-level stage (not the nested stage)

### Example: Sub stage

In this sub-stage example, if a change request gets created from the sub stage (deploy PROD), the details of the parent root-level stage (deploy) are used in the change request, and orchestration tasks are also associated with the parent root-level stage (deploy).

```
stage("deploy") {
    stages{
        stage('deploy UAT') {
            when{
                branch 'dev'
            }
            steps{
                snDevOpsStep ()
                echo "deploy in UAT"
            }
        }
        stage('deploy PROD') {
            when {
                branch 'master'
            }
            steps{
                snDevOpsStep ()
                echo "deploy in prod"
                snDevOpsChange()
            }
        }
    }
}
```

### Example: Parallel stage

In this parallel stage example, if a change request is created from a sub stage (UAT test-1 and/or UAT static code test), only the first change request is created (using the details of the parent root-level stage, UAT test) regardless of whether both sub stages (UAT test-1 and UAT static code test) get triggered.

There is no indication of which parallel stage generated the change, and orchestration tasks are associated with the parent root-level stage (UAT test).

```
stage('UAT test') {
    parallel {
        stage('UAT test-1') {
```

```
        steps {
            snDevOpsStep()
            snDevOpsChange()
            // 'UAT test-1' tasks
        }
        post {
            success {
                // post success tasks. E.g.: junit
                '**/target/surefire-reports/*.xml'
            }
        }
    }
    stage('UAT static code test') {
        steps {
            snDevOpsStep()
            snDevOpsChange()
            // 'UAT static code test' tasks
        }
    }
}
```

## Configure SonarQube scans on Jenkins pipelines

Configure SonarQube scans on Jenkins pipelines.

## Before you begin

Ensure that you meet the following pre-requisites before configuring your Jenkins pipelines for SonarQube scans

- SonarCloud or SonarQube version 8.6.1 community edition
  - SonarQube Scanner for Jenkins 2.4 or higher is installed on your Jenkins instance/environment.
  - ServiceNow DevOps plugin version 1.27 or later for Jenkins.
  - Ensure that SonarQube scans are configured and exist on your Jenkins pipelines using the SonarQube Scanner plugin for Jenkins. For more information, see
    - [SonarQube Scanner for Jenkins](#)
    - [SonarScanner for Jenkins](#)
  - A SonarQube tool is created and is connected to the SonarQube server. For more information, see [Create a SonarQube tool record in DevOps](#)

Role required: sn devops.admin

## About this task

When you run a pipeline which has SonarQube scan executions, the details are fetched into ServiceNow DevOps from the Jenkins pipeline. Using the Jenkins plugin, from the ServiceNow Store we check if the scan execution is configured in the Jenkins pipeline and check for SonarQube scans on every orchestration stage, using the `withSonarQubeEnv` tag. If a SonarQube analysis has happened on any stage of the pipeline's execution, as part of our end notification we add a model with the `scanID` and `url` details for every scan that occurs in a particular stage. These scan analytics or details are correlated and displayed in the Software Quality Summary related list from Change requests and Task executions.

## Procedure

- In Jenkins, navigate to **Manage Jenkins > Configure System**, select the **Enable injection of SonarQube server configuration as build environment variables** check box in the **SonarQube Servers** section, and fill in the fields.

### SonarQube Installations

Field	Description
Name	A unique name for the configuration/project.
Server URL	URL or IP address of the SonarQube server.
Server authentication	Select the relevant authentication token type for the configuration.

**i Note:** Use these same SonarQube server details or values while creating the SonarQube tools in ServiceNow DevOps as well as in the Jenkins pipeline.

- Click **Test Connection**.

Verify that the connection successful message is shown.

- Click **Apply and Save**.

- Run the pipeline.

### Result

Based on the scan results on various stages of pipeline's execution, the results are shown in the corresponding step of the pipeline. Inbound events are created for notifications and subflows are triggered based on the notification type and capability.

### What to do next

**i Note:**

Navigate to the pipeline UI to view the scan details under the **Quality > Software Quality Results**.

- View scan details as part of Task Executions. View details of all the Sonar scans that are part of the task execution mapped to a build or release pipeline execution step.

1. Navigate to **DevOps > Orchestrate > Task Execution** click a relevant Task Execution record.

2. Click the Software Quality Summary related list.

3. Click a relevant Scan ID record.

The Software Quality Scan Summary and Scan Details are displayed.

- View scan details as part of Change Request. View all the scans that were part of this build/release pipeline in the **Software Quality Results > Software Quality Summary** related list.

1. Navigate to **DevOps > Orchestrate > Pipeline Change Requests**

2. Click the Software Quality Summary related list.

3. Click a relevant Scan ID record.

The Software Quality Scan Summary and Scan Details are displayed.

## Configure JFrog in Jenkins

Configure JFrog plugin for Jenkins to publish, deploy, or download artifacts.

### Before you begin

Role required: admin role in Jenkins

### Procedure

1. Install the [Artifactory](#) plugin, to integrate Jenkins with JFrog , to publish, deploy, or download artifacts.
2. After the installation, configure the JFrog server details from **Manage Jenkins > Configure System**.

#### Note:

To be able to integrate JFrog with Jenkins, the build info must be published for JFrog along with the artifacts.

3. Use the following scripts to download or upload artifacts to Jenkins.

Scripted pipeline to upload:

```
def server = Artifactory.server 'JFROG1'
def uploadSpec = """
    "files": [
        "pattern":
            "${env.WORKSPACE}/target/artifact-1.3.jar",
        "target": "default-docker-virtual/"
    ]
}"""

def buildInfo = server.upload(uploadSpec)
server.publishBuildInfo buildInfo
```

Scripted pipeline to download:

```
def server = Artifactory.server 'JFROG1'
def downSpec= """
    "files": [
        "pattern":
            "default-docker-local/artifact-1.3.jar",
        "target": "/var/jfrog_artifacts/"
    ]
}"""

def buildInfo = server.download(downSpec)
server.publishBuildInfo buildInfo
```

### Notify change request rejection or cancellation reason to Jenkins pipeline

Send change request rejection or cancellation reason along with approver name and the change request number to Jenkins pipeline logs.

## Before you begin

- Ensure that you have upgraded to ServiceNow DevOps version 1.28 or later.
- Have an active Jenkins integration.

Role required: sn\_devops.admin

## About this task

You can send change request rejection or cancellation reasons or comments to the Jenkins pipeline logs.

- Ensure that you enter appropriate reasons or comments when rejecting or canceling a Change Request manually.
- If you have loaded demo data during upgrade, and are using the DevOps Demo Change Automation flow or a custom flow based on it, a notification with default message values is sent to the Jenkins pipeline logs.

### **i Note:**

- The Change request number is also automatically sent to the Jenkins pipeline logs (for both scripted and freestyle pipelines) as soon as the change is created.
- The *Approver name* and time stamp of the cancellation/rejection is also automatically sent to the Jenkins pipeline logs.

## Procedure

1. To manually reject or cancel change requests, follow these steps:

- Navigate to **DevOps > Orchestrate > Pipeline Change Requests > Change Request record**.
- Open the required Change Request record.
  - From the context menu, click **Cancel Change**. In the **Cancel Change Request > Reason** field, enter an appropriate reason for cancelling the change, and click **Save**.
  - In the Approvers related list, provide your inputs in the **Comment** field, right-click the record, and click **Reject**.

The change request is canceled/rejected and the reason for canceling the change is added to the **Comments** field and sent to the Jenkins pipeline log.

2. To send custom messages (from auto-rejected change requests) to Jenkins, follow these steps:

- Navigate to **Flow Designer > DevOps Demo Change Automation flow > DevOps Demo Change Policy**.
- Navigate to the *DevOps Auto Reject* decision > *DevOps Apply Change Approval Definition* subflow > *Devops Create Auto Approval Record* action.
- Modify the action's input script for the `approval.comments` attribute value.  
By default, the auto-rejected change requests stores and sends the `approval.comments = 'Auto ' + state + ' via Change Policy';` variables as messages to the Jenkins pipeline as notifications.

3. In Jenkins, navigate to the pipeline (that is corresponding to the rejected change request) > **Console Output**.

The Change Request's rejection or cancellation comments that are stored as part of the step execution reflect in the Jenkins Console Output.

## Notify ServiceNow DevOps change request number to Jenkins pipelines

Send change request numbers to the Jenkins pipeline steps or logs when a change request is created in ServiceNow DevOps.

### Sending ServiceNow DevOps change request to Jenkins pipeline

Change requests that are created as part of Jenkins pipeline executions for both scripted and free-style pipelines are notified to the Jenkins pipeline console.

When you trigger a scripted Jenkins pipeline and a change request is created as part of the pipeline execution stage in ServiceNow DevOps, the change request number is notified to Jenkins. The change request number displays in the Jenkins pipeline job logs for scripted pipelines.

Upon triggering a free-style Jenkins pipeline, the change request that is created as part of the ServiceNow DevOps pipeline execution stage waits for the approval action to be performed on the change. The change request number and details display against the relevant build in the Jenkins console's **Build History** section.

**Note:** After an approval or rejection action is performed on the change request in ServiceNow DevOps, the change request displays in the Jenkins pipeline build logs, similar to scripted pipelines.

### Get change request number in Jenkins pipeline

Retrieve the change request number in a Jenkins pipeline based on specific change details by running the `snDevOpsGetChangeNumber` script.

#### Before you begin

Role required: Jenkins admin

#### Procedure

1. In your Jenkins dashboard, open the pipeline for which you want to retrieve the change request number.
2. Navigate to **Configure > Pipeline**.
3. In the Pipeline script section, update the `snDevOpsGetChangeNumber` script with the following input parameters:
  - Pipeline Name
  - Build Number
  - Stage Name
  - Branch Name (only for multi-branch pipeline)

**Note:** If you do not provide the change request details as input parameters, the change request number associated with the current pipeline and stage will be retrieved.

```
steps {
    snDevOpsChange()
    script{
        changeNumber = snDevOpsGetChangeNumber() //Returns
        Change Request Number of current Pipeline, current Stage
    }
}
```

```

changeNumber =
snDevOpsGetChangeNumber(changeDetails:
""" {"build_number": "${ciBuildNumber}", "pipeline_name": "${ciPipelineName}", "stage_name": "${stageName}" } """) // Returns Change Request Number
of provided pipeline_name, stage_name, build_number
echo '">> Deploy in prod'
sh "echo ${changeNumber}"
}

```

4. Save the script.
5. Navigate to **DevOps > Orchestrate > Pipeline Change Requests**.
6. Select the change record associated with the pipeline.
7. Approve the change request by selecting **#Approved#** in the **State** field.
8. In Jenkins, open the pipeline for which you are retrieving the change request number.
9. Select **Build Now**.

The change request number associated with the pipeline will be displayed as an output in the pipeline.

### Update change request details in Jenkins pipeline

Update the change request details associated with a Jenkins pipeline by running the `snDevOpsUpdateChangeInfo` script in the pipeline.

#### Before you begin

Role required: Jenkins admin

#### About this task

When you update the **state** parameter in a change request, only the following transitions are supported:

- **cancel**: Change request state must be **implement** to move the state to **cancel**. **reason** is a mandatory input to update the state to canceled.
- **closed**: Change request state must be **implement** or **post implement** to move the state to **close**. **close\_code** and **close\_notes** are mandatory input to update the state to closed.

Specify the change request state as an integer value:

- 4 - Cancel (Value set in the `sn_devops.change_request.cancel_state` property)
- 3 - Closed (Value set in the `sn_devops.change_request.closed_state` property)

When you update a choice field, you must specify a valid choice value that is available in the corresponding choice list. For example, the choice list values for the **Close code** field are **successful**, **successful\_issues**, and **unsuccessful**.

#### Procedure

1. In your Jenkins dashboard, open the pipeline for which you want to update the change request details.
2. Navigate to **Configure > Pipeline**.
3. In the Pipeline script section, update the `snDevOpsUpdateChangeInfo` script with the following input parameters:
  - Change request number whose details need to be updated.
  - Change request details to be updated as Key:Value pairs.

```
{
  "short_description": "Test description", "priority": "1",
  "start_date": "2021-02-05 08:00:00",
  "end_date": "2022-04-05 08:00:00", "justification": "test justification", "description": "test description",
  "cab_required": <true/false>, "comments": "This update for work notes is from jenkins file", "work_notes": "test work notes",
  "assignment_group": "<SYS_ID>", "state": "<STATE_CODE>",
  "close_code": "<successful/Successful_issues/unsuccessful>",
  "reason": "<As per Choice List>"}
```

4. Save the script.
5. Navigate to **DevOps > Orchestrate > Pipeline Change Requests**.
6. Select the change record associated with the pipeline.
7. Approve the change request by selecting **#Approved#** in the **State** #field.
8. In Jenkins, open the pipeline for which you are updating the change request details.
9. Select **Build Now**.

The change request details specified in step 3 will be updated for the pipeline.

#### **ServiceNow DevOps change request state in Jenkins pipeline logs**

You can use the Jenkins Snippet Generator utility to configure how and when the change state must be displayed in the Jenkins pipeline job logs. This enables developers to view the status of the change in the console logs of the pipeline itself.

To generate a step snippet, navigate to the Pipeline Syntax from a configured pipeline, select the **SnDevOpsChange** step from the **Sample Step** list, and update the values for the change state variables in the step. Select the **ignore error** option to prevent job failure if there is an error. Select **Generate Pipeline Script** to create a snippet. You can copy and paste the snippet into the pipeline to start receiving the change state notifications. Update the following variables in the **SnDevOpsChange** step to receive change state notifications:

- **Polling Interval:** Specifies the frequency (in seconds) at which Jenkins polls ServiceNow for the change status and updates the console logs with the status. The change status is updated in the console logs only when the **Change state transitions**, **Assignment Group updates**, **Approval updates**, or **Planned Start/End date** fields are updated.
- **Note:** If no value is entered in the field, polling interval check won't be run to update the change status in the console logs.
- **Change Creation Timeout:** Specifies the change creation timeout value in seconds. On timeout, Jenkins checks the change creation status in ServiceNow. If the change did not get created, the pipeline is resumed or aborted based on the **Abort on change creation failure** flag. By default, the pipeline is aborted when timeout is specified and the **Abort on change creation failure** flag is selected.
- **Note:** If no value is entered in the field, change creation timeout check won't be run to update the pipeline.

- **Abort on change creation failure:** Abort or resume the pipeline if change is not created till the change creation timeout.
  - Selected: Abort
  - Cleared: Resume

- **Change Step Timeout:** Specifies the change step timeout value in seconds. On timeout, Jenkins checks the change step status in ServiceNow. If the change step is still in progress, the pipeline is resumed or aborted based on the **Abort on change step timeout** flag. By default, the pipeline aborted when timeout is specified, and the **Abort on change step timeout** flag is selected.

**Note:** If no value is entered in the field, change step timeout check won't be run to update the pipeline.

- **Abort on change step timeout:** Abort or resume the pipeline if the change step is still in progress on change step timeout.

- Selected: Abort
- Cleared: Resume

After the required variables are configured for the change step, the state of the change is displayed in the Jenkins pipeline job logs in the following way:

### Nested and parallel stages in Jenkins pipelines

Use nested and parallel stages in scripted Jenkins pipelines to speed up your pipeline execution. Change requests are created for nested and parallel stages and not just for the parent stage.

### Support for nested and parallel stages in Jenkins pipelines

You can use nested and parallel stages in scripted Jenkins pipelines to automate and speed up tasks that can be run in parallel. For example, you have a scripted Jenkins pipeline with nested and parallel stages for various test cases such as different quality checks for different operating systems and browsers.

ServiceNow DevOps supports processing parallel and nested stages in Jenkins pipelines and displays the pipeline, in the DevOps pipeline UI. In effect, the ServiceNow DevOps pipeline UI renders or replicates the Jenkins pipeline UI in real time. From the **Pipeline Execution** view of the relevant pipeline, click the **Pipeline UI** related link to view the real-time state of the pipeline as it appears in Jenkins. The associated artifact details that are sourced from the build pipeline, Test Results, Software Quality Summary Results, and Change request details display on the pipeline UI.

**Important:** Support for parallel and nested stages is restricted to scripted pipelines in Jenkins. Freestyle pipelines continue to appear in a sequential or serial manner in the DevOps pipeline UI, even if parallel and nested stages are part of freestyle pipelines in Jenkins.

### Change requests in nested and parallel stages

Change requests are created for all nested and parallel stages, once all upstream events (prior to the change request) are received. In previous releases, nested or parallel stages in Jenkins pipelines were not identified nor processed in ServiceNow DevOps. Only parent stages were identified and processed in a linear or sequential manner. If change requests existed as part of any nested and parallel stages, those change requests were ignored and a single change request was processed as part of the parent stage. When you run a new pipeline after upgrading, new steps and steps executions are created for nested stages.

Nested and parallel stages were not processed previously, and approval groups were mapped only to the parent stage. Because nested and parallel stages are identified during processing, you must verify that relevant approval groups are mapped to the appropriate

nested or parallel stage. If subsequent steps of the pipeline are dependent on the change request's being approved, the pipeline execution is paused, and resumed when the change request is approved.

## Upgrade Considerations

If you are already using Jenkins with nested and parallel pipelines as your orchestration tool, consider the following while upgrading.

- Upgrade during off-peak hours.
- Ensure that you do not have any pipeline executions that are currently in progress by ServiceNow DevOps. If pipeline executions are being processed, step executions might not be created as expected for the in-progress pipeline runs. Rerun the pipeline to create appropriate step executions.

### Jenkins log levels and Log Recorders

Set log levels in the Jenkins plugin for ServiceNow DevOps based on the extent of log detail you need for debugging.

### Base system Log Recorder in Jenkins logs

You can set the level of detail that your log messages capture. In previous versions of the plugin, you could only enable or disable the **Debug** mode, which meant that you could either enable or disable all Jenkins logs without the ability to select the level of information or detail you want to save in your logs. Often, this leads to excessive information in your logs, used up more disk space, and collected information that might not always be relevant or useful.

The flexibility to set log levels allows you to select the right log level information that is appropriate for your organizational needs. You can select custom log levels based on the log levels or categories that Jenkins currently supports.

The log level is turned off by default for first-time DevOps installations. Consider the following items after successfully upgrading:

- if the **Debug** check box was previously selected, the log level is enabled and set to *Info* by default.
- If the **Debug** check box was cleared previously, by default, log levels are tuned off.

#### **Note:**

The Jenkins log does not record log messages at a level lower than *Info*. The base system ServiceNow DevOps log recorders are used to record this message.

You can select a custom level from the Jenkins plugin- ServiceNow DevOps Configuration form > Log level list. A base system ServiceNow DevOps log recorder is created in the Jenkins Log Recorders list. For more information, see [Jenkins documentation on viewing logs](#).

**Note:** Do not modify the log recorder from the Jenkins > Log Recorders > Configure log recorders. Editing the log recorder from Jenkins creates a duplicate entry for any changes to the log levels that you make. Please use the list of log levels in the ServiceNow DevOps Configuration form, to modify the log levels.

## Argo CD integration with DevOps Change Velocity

Integrate Argo CD with DevOps Change Velocity to automate the deployment of applications from GitHub repositories.

## Overview

This integration enables the ServiceNow platform to manage the change request closure process based on the sync status received from Argo CD for continuous deployment of applications.

You must activate the DevOps Integration with Argo CD plugin (`sn_devops_argocd`) before connecting your Argo CD instance in ServiceNow. For more information on activating a plugin, see [Activate a plugin](#).

## Workflow

Here is the workflow of how the continuous deployment process works through Argo CD in ServiceNow DevOps.

- Create an Argo CD tool connection in ServiceNow DevOps Change Velocity using the Classic or Workspace UI.
- Create a webhook in Argo CD manually.
- Update your Config file in the GitHub repository for deployment. While updating the Config file, specify the change request number in the commit tag (`sn_devops_change-<change request number>`).

**Note:** The change request number specified in the commit tag must already be created by the CI pipeline and in the implement state.

- Sync the required app associated with your Config file and repository in Argo CD.
- Once sync is successful, notifications are sent to ServiceNow DevOps and inbound events are created.
- Change request number is retrieved from the inbound events and updated with the Argo CD synchronization status.
- Change request is closed and based on the sync status, the close code, worknotes, and close notes fields are updated in the change request.

## Example

The following examples specify how changes made in Argo CD are notified to ServiceNow DevOps through the webhook.

- The Config file is updated in GitHub with the following commit tag format:
- Inbound events are created in ServiceNow when an app is synced in Argo CD:
- If sync is successful, the change request is closed and the close code, worknotes, and close notes fields are updated in the change request:

### Onboard Argo CD to DevOps Change Velocity – Workspace

Connect your Argo CD instance using the playbook in DevOps Change workspace.

#### Before you begin

Role required: `sn_devops.admin` or `sn_devops.tool_owner`

Complete the steps specified in the [Getting started](#) section before connecting to a tool.

## Procedure

1. Navigate to **Workspaces > DevOps Change Workspace** and use one of the following options to open the Playbook to onboard Argo CD.

Option	Steps
Homepage	<ul style="list-style-type: none"> <li>a. Select the <b>Connect tool</b> widget</li> <li>b. On the# Connect to a tool# modal, select Argo CD from the <b>Orchestration</b> category.</li> </ul>
Applications module	<ul style="list-style-type: none"> <li>a. Select <b>Applications ()</b>.</li> <li>b. Select an existing application, or create one. To create an application, see <a href="#">Create an application in DevOps Change Velocity</a>.</li> <li>c. From the# Recommended actions# pane, select the#<b>Connect a tool</b> card.</li> <li>d. On the# Connect to a tool# modal, select Argo CD from the <b>Orchestration</b> category.</li> </ul>
Tools module	<ul style="list-style-type: none"> <li>a. Select <b>Tools ()</b>.</li> <li>b. From the Tools list, select <b>Orchestration tools</b>.</li> <li>c. Select <b>Connect orchestration tool</b>.</li> <li>d. On the# Connect to a tool# modal, select Argo CD.</li> </ul>

2. Specify a name for the tool in the **Tool name** field, and select **Next**.

3. On the Argo CD instance details playbook activity:

- a. Enter the URL of your Argo CD instance.
  - b. Enter the login credentials of the Argo CD instance.
  - c. Optional: Select the MID Server option and enter its details if your Argo CD instance is attached to a MID Server.
- A MID server is required if your tool instance is hosted on-prem. For more information about MID server, see [MID Server selection](#) ↗.

4. Select **Connect**.

5. Specify the access for the tool.

- a. If you want to control access to the tool, add the groups that must be given access to the tool in the **Maintained by** field.

The tasks these users in the groups can perform depends on the role assigned to them.

- DevOps Tool Owner role: Can view and edit the tool.
- DevOps App Owner role: Can view the tool and can associate, discover, import historical data, and modify pipeline steps (if applicable) of the tool's objects (such as plans, repositories, and pipelines).
- DevOps Administrator role: Can edit all tools.
- Other DevOps roles: Can view the tool.

**i Note:** If you don't select a group and skip this step, all users with the DevOps Tool Owner role will be able to edit the tool.

- b.** If you choose to control access to the tool, the **All App Owners can view and associate tool objects to applications** option becomes available for selection.

This option enables all users having the DevOps App Owner role to access the tool. If selected, they'll be able to view, associate, discover, import historical data, and modify pipeline steps (if applicable) of the tool's objects.

- c. Select Assign.**

- 6.** From the **Summary** page, select **View tool record** to review the details of the connected instance.

## Result

You've successfully onboarded your Argo CD tool to DevOps Change Velocity.

### Onboard Argo CD to DevOps Change Velocity — Classic

Connect your Argo CD instance using the Classic UI in DevOps Change Velocity.

#### Before you begin

Role required: sn\_devops.admin or sn\_devops.tool\_owner

#### Procedure

1. Navigate to **DevOps > Tools > Create New** and create a record.
2. Enter a **Tool Name** and fill in the tool details.

#### Create DevOps Tool form

Field	Description
Tool integration	Tool to integrate. In this case, select <b>Argo CD</b> .
Tool URL	URL of the existing Argo CD instance to integrate.  For example: <a href="https://argocd.k8s.sndevops.xyz/">https://argocd.k8s.sndevops.xyz/</a>
Tool username	Username of your Argo CD instance.

Field	Description
Tool password	Password of your Argo CD instance.

3. Optional: Select **MID Server** for an on-premises tool that is attached to a MID Server. Application is automatically set to DevOps and capability is set to REST.
4. Select **Submit**.  
The tool is automatically Connected Successfully using a connection alias, and HTTP tool connection (basic authentication credential).

### Configure webhooks in Argo CD manually

Configure webhooks in Argo CD to send sync notifications to the DevOps Change Velocity application.

#### Before you begin

Role required: Argo CD admin

#### About this task

The subscription to Argo CD application events can be defined using the `notifications.argoproj.io/subscribe.<trigger>.<service>: <recipient>` annotation.

#### Procedure

1. In Argo CD, open the Argo CD app for which you are configuring webhooks.
2. Navigate to **APP DETAILS > SUMMARY**.
3. Add the `notifications.argoproj.io/subscribe.<trigger>.<service>: <recipient>` annotation with the trigger condition to sync change status for your instance.

### JFrog integration with DevOps Change Velocity

The JFrog integration enables you to track artifacts published to JFrog from Jenkins builds.

You must activate the Jenkins Artifactory plugin in your Jenkins server to generate the artifacts as part of your pipeline run and publish them in the JFrog artifact repositories.

- **Note:** Historical import of JFrog artifacts is not supported. You can import JFrog artifacts that are built by Jenkins using the Service Catalog import historical data process. For more information, see [Importing historical data for DevOps tools](#). Discover and configure is also not supported for JFrog.

#### Get started

Use one of the following options to onboard JFrog. For a guided experience, use the workspace to onboard a tool. Alternatively, you can use the Service Catalog or Classic experience.

#### Onboard JFrog to DevOps Change Velocity — Workspace

Connect your JFrog instance using the playbook in DevOps Change workspace to track artifact repositories published to JFrog from Jenkins builds.

#### Before you begin

Role required: `sn_devops.admin` or `sn_devops.tool_owner`

Complete the steps specified in the [Getting started](#) section before connecting to a tool.

## Procedure

1. Navigate to **Workspaces > DevOps Change Workspace** and use one of the following options to open the Playbook to onboard JFrog.

Option	Steps
Homepage	<ul style="list-style-type: none"> <li>a. Select the <b>Connect tool</b> widget</li> <li>b. On the# Connect to a tool# modal, select JFrog from the <b>Artifact</b> category.</li> </ul>
Applications module	<ul style="list-style-type: none"> <li>a. Select <b>Applications ()</b>.</li> <li>b. Select an existing application, or create one. To create an application, see <a href="#">Create an application in DevOps Change Velocity</a>.</li> <li>c. From the# Recommended actions# pane, select the#<b>Connect a tool</b> card.</li> <li>d. On the# Connect to a tool# modal, select JFrog from the <b>Artifact</b> category.</li> </ul>
Tools module	<ul style="list-style-type: none"> <li>a. Select <b>Tools ()</b>.</li> <li>b. From the Tools list, select <b>Artifact tools</b>.</li> <li>c. Select <b>Connect artifact tool</b>.</li> <li>d. On the# Connect to a tool# modal, select JFrog.</li> </ul>

2. Specify a name for the tool in the **Tool name** field, and select **Next**.

3. On the JFrog instance details playbook activity:

- a. Enter the URL of your JFrog instance.
- b. Enter the login credentials of the global admin for the JFrog instance.
- c. Optional: Select the MID Server option and enter its details if your JFrog instance is attached to a MID Server.  
A MID server is required if your tool instance is hosted on-prem. For more information about MID server, see [MID Server selection](#) ↗.

4. Select **Connect**.

The permission check is run and the permission requirements are displayed in a pop-up. The credentials that you specified must have the Administer Platform role in JFrog for seamless discovery and import of tool objects. This is a limitation from JFrog.

5. Select **Continue** or **Connect Anyway**.

**Note:** If your tool credential has changed, you must update the credentials in your ServiceNow instance. For more information, see [Update third-party tool credentials in DevOps Change Velocity](#).

#### 6. Specify the access for the tool.

- a. If you want to control access to the tool, add the groups that must be given access to the tool in the **Maintained by** field.

The tasks these users in the groups can perform depends on the role assigned to them.

- DevOps Tool Owner role: Can view and edit the tool.
- DevOps App Owner role: Can view the tool and can associate, discover, import historical data, and modify pipeline steps (if applicable) of the tool's objects (such as plans, repositories, and pipelines).
- DevOps Administrator role: Can edit all tools.
- Other DevOps roles: Can view the tool.

**Note:** If you don't select a group and skip this step, all users with the DevOps Tool Owner role will be able to edit the tool.

- b. If you choose to control access to the tool, the **All App Owners can view and associate tool objects to applications** option becomes available for selection.

This option enables all users having the DevOps App Owner role to access the tool. If selected, they'll be able to view, associate, discover, import historical data, and modify pipeline steps (if applicable) of the tool's objects.

- c. Select **Assign**.

#### 7. From the **Summary** page, select **View tool record** to review the details of the connected instance.

### Result

You've successfully onboarded your JFrog tool to DevOps Change Velocity.

### Onboard JFrog to DevOps Change Velocity — Service Catalog

Connect your JFrog instance using the ServiceNow Service Catalog.

#### Before you begin

Role required: sn\_devops.admin or sn\_devops.tool\_owner

#### Procedure

1. Navigate to **All > Service Catalog > Catalog Definitions > Maintain Items** and search for DevOps.

**Note:** You can also access the service catalog from Employee Center or Service portal.

2. From the DevOps catalog items, select and activate **DevOps App Onboarding** and **DevOps Tool Onboarding**.

3. After activating, select **DevOps Tool Onboarding** and select **Try it**.

4. In the DevOps Tool Onboarding form, enter the tool details:

Field	Description
Tool name	Name for your JFrog integration.
Tool integration	Select JFrog.
Tool URL	URL of your JFrog instance.
Tool username, password or access token	Login credentials of the global admin for the existing JFrog instance.
Use MidServer	Optional. Select#MID Server#for an on-premises tool that is attached to a#MID Server. Application is automatically set to DevOps #and capability is set to REST.

5. Select **Order Now**.

A request is created. When the request is approved, the tool is connected.

### Onboard JFrog to DevOps Change Velocity — Classic

Connect to an instance of the JFrog artifact tool to enable you to track artifacts published to JFrog from Jenkins builds.

#### Before you begin

Role required: sn\_devops.admin or sn\_devops.tool\_owner

Complete the steps specified in the [Getting started](#) section before connecting to a tool.

#### Procedure

1. Enter the JFrog instance details to connect to DevOps Change Velocity by navigating to **All > DevOps > Tools > Create New**.
2. Enter a value in the **Tool Name** field and fill in the tool details.

#### Create DevOps Tool form

Field	Description
Tool integration	Tool to integrate. In this case, select <b>JFrog</b> .
Tool URL	URL of the existing JFrog instance to integrate.
Tool username, Tool password / Access token	Login credentials of the global admin for the existing JFrog instance.

3. Select **MID Server** for an on-premises tool that is attached to a MID Server.

Application is automatically set to DevOps and capability is set to REST.

4. Click **Submit**.

**Note:** If your tool credential has changed, you must update the credentials in your ServiceNow instance. For more information, see [Update third-party tool credentials in DevOps Change Velocity](#).

If the connection is successful, the tool is created in ServiceNow and connected to your JFrog instance.

On successful tool creation, you're taken to the tool record page.

5. If you want to control access to the tool, add the groups that must be given access to the tool in the **Maintained by** field on the **Access** tab.

The tasks the users in the groups can perform depends on the roles assigned to them.

- DevOps Tool Owner role: Can view and edit the tool.
- DevOps App Owner role: Can view the tool and can associate, discover, import historical data, and modify pipeline steps (if applicable) of the tool's objects (such as plans, repositories, and pipelines).
- DevOps Administrator role: Can edit all tools.
- Other DevOps roles: Can view the tool.

**Note:** Only groups containing users with DevOps roles are available for selection in the **Maintained by** field.

The **All App Owners can view and associate tool objects to applications** option becomes available for selection if you choose to restrict access to the tool. This option enables all users having the DevOps App Owner role to access the tool. If selected, they'll be able to view, associate, discover, import historical data, and modify pipeline steps (if applicable) of the tool's objects.

## SonarQube integration with DevOps Change Velocity

Connect SonarQube that is integrated with your CI/CD pipelines to DevOps Change Velocity, to retrieve code quality and code security results.

### Overview

Sonar scans that are configured on GitHub Actions, Jenkins, and Azure DevOps pipelines are supported in DevOps Change Velocity. Both SonarCloud and SonarQube (on-premises) are supported.

You can view the code quality and code security summary results either in the related list of a Change Request or the Task Execution of the pipeline in your ServiceNow instance. You can also use code quality and code security results in defining change policies and conditions for change automation.

DevOps Change Velocity captures both overall and new code metrics.

### Get started

You can onboard your Sonar instance using an admin or a non-admin Sonar user.

Pre-requisite configuration for admin user

In SonarQube: Project-level access to your SonarQube instance to configure scans for all your projects.

Pre-requisite configuration for non-admin users

- In SonarQube:
  - The non-admin user must have access to a branch created using Branch Analysis. For more information, see [Branch Analysis](#).
  - The non-admin user must have Browse and Execute Analysis permissions for the projects (both private and public) for which you want to run and view scan results.

**i Note:** You can set up branch analysis to enable SonarCloud to analyze branches in your projects apart from the main branch. You can't set up or perform branch analysis on SonarQube community edition licenses. Upgrade to SonarQube Developer or Enterprise editions to set up branch analysis on SonarQube on-premises implementations.

- In DevOps Change Velocity

1. Navigate to **All > DevOps > Administration > Properties**.
2. Enable the **DevOps Non-Admin Software Quality Summary Flag** property by selecting the **Yes** option.

Sonar custom action and extension are available in the GitHub and Azure DevOps marketplace respectively. For Jenkins, the Sonar scan results are retrieved using ServiceNow Jenkins plugin.

For more information on the scan results captured in ServiceNow, see [Software Quality Results](#).

Use one of the following options to onboard SonarQube. For a guided experience, use the workspace to onboard a tool. Alternatively, you can use the Service Catalog or Classic experience.

### Onboard SonarQube to DevOps Change Velocity — Workspace

Connect to your Sonar instance using the playbook in DevOps Change Workspace.

#### Before you begin

Role required: sn\_devops.admin or sn\_devops.tool\_owner

#### Procedure

1. Navigate to **Workspaces > DevOps Change Workspace** and use one of the following options to open the Playbook to onboard SonarQube.

Option	Steps
Homepage	<ol style="list-style-type: none"> <li>Select the <b>Connect tool</b> widget</li> <li>On the# Connect to a tool# modal, select SonarQube from the <b>Software quality</b> category.</li> </ol>
Applications module	<ol style="list-style-type: none"> <li>Select <b>Applications ()</b>.</li> <li>Select an existing application, or create one. To create an application, see <a href="#">Create an application in DevOps Change Velocity</a>.</li> </ol>

Option	Steps
	<p>c. From the# Recommended actions# pane, select the<b>Connect a tool</b> card.</p> <p>d. On the<b>Connect to a tool</b># modal, select SonarQube from the <b>Software quality</b> category.</p>
Tools module	<p>a. Select <b>Tools</b> () .</p> <p>b. Select <b>Software quality tools</b>.</p> <p>c. Select <b>Connect software quality tool</b>.</p> <p>d. On the<b>Connect to a tool</b> modal, select <b>SonarQube</b>.</p>

**2.** Enter a name to identify your tool and click **Next**.

**3.** On the Enter Sonar instance details playbook activity section, enter the following details:

- a.** In the **URL of the SonarQube instance** field, enter the SonarQube or SonarCloud instance URL.

For example, <https://sonarcloud.io>.

- b.** In the **SonarQube username** field, enter the user name for your Sonar account.
- c.** In the **Password or access token** field, enter the token generated in Sonar.
- d.** If your Sonar instance is attached to a MID Server, select the **Use MID Server** option and enter its details.

A MID server is required if your tool instance is hosted on-prem. For more information about MID server, see [MID Server selection](#) ↗.

**4.** Click **Connect** and review the details of the successfully connected Sonar instance.

**5.** Specify the access for the tool.

- a.** If you want to control access to the tool, add the groups that must be given access to the tool in the **Maintained by** field.

The tasks these users in the groups can perform depends on the role assigned to them.

- DevOps Tool Owner role: Can view and edit the tool.
- DevOps App Owner role: Can view the tool and can associate, discover, import historical data, and modify pipeline steps (if applicable) of the tool's objects (such as plans, repositories, and pipelines).
- DevOps Administrator role: Can edit all tools.
- Other DevOps roles: Can view the tool.

**Note:** If you don't select a group and skip this step, all users with the DevOps Tool Owner role will be able to edit the tool.

- b.** If you choose to control access to the tool, the **All App Owners can view and associate tool objects to applications** option becomes available for selection.

This option enables all users having the DevOps App Owner role to access the tool. If selected, they'll be able to view, associate, discover, import historical data, and modify pipeline steps (if applicable) of the tool's objects.

**c. Select Assign.**

6. From the **Summary** page, select **View tool record** to review the details of the connected instance.

**Result**

You've successfully onboarded your SonarQube tool to DevOps Change Velocity.

**Onboard SonarQube to DevOps Change Velocity — Service Catalog**

Connect your Sonar instance using the ServiceNow Service Catalog.

**Before you begin**

Role required: sn\_devops.admin or sn\_devops.tool\_owner

**Procedure**

1. Navigate to **All > Service Catalog > Catalog Definitions > Maintain Items** and search for DevOps.

**Note:** You can also access the service catalog from Employee Center or Service portal.

2. From the DevOps catalog items, select and activate **DevOps App Onboarding** and **DevOps Tool Onboarding**.
3. After activating, select **DevOps Tool Onboarding** and select **Try it**.
4. In the DevOps Tool Onboarding form, enter the tool details:

Field	Description
Tool name	Name for your Sonar integration.
Tool integration	Select SonarQube.
Tool URL	SonarQube/SonarCloud instance URL.  For example,https://sonarcloud.io.
Tool password/ Access token	SonarQube user token.
Use MidServer	Optional. Select#MID Server#for an on-premises tool that is attached to a# MID Server. Application is automatically set to #DevOps and capability is set to REST.

5. Select **Order Now**.

A request is created. When the request is approved, the tool is connected.

## Onboard SonarQube to DevOps Change Velocity — Classic

Connect to your Sonar instance to retrieve scan results.

### Before you begin

Role required: sn\_devops.admin or sn\_devops.tool\_owner

### Procedure

1. Navigate to **All > DevOps > Tools > Create New**.
2. In the Create DevOps tool form, enter the tool details:

Tool name	Name for the tool you're integrating.
Tool integration	Tool to integrate. Select SonarQube.
Tool URL	<p>SonarQube/SonarCloud instance URL. For example, https://sonarcloud.io</p> <p><b>i Note:</b> Don't enter the slash (/) character at the end of your Sonar tool URL.</p>
Tool password / Access token	<p>SonarQube user token</p> <p><b>i Note:</b> Only user tokens generated from <b>User &gt; My Account &gt; Security</b> in your Sonar instance are supported.</p> <p>Before generating the token for a SonarQube user, ensure that you grant Browse permission for all your Sonar projects.</p> <pre>api: Grants complete read/write access to the API, including all groups and projects, the container registry, and the package registry.</pre>
Use MID Server	MID Server is optional. Select MID Server for an on-premises tool that is attached to a MID Server. The Application value is automatically set to DevOps and the Capability value is set to REST. For more information, see <a href="#">MID Server selection</a> .

3. Click **Submit** to connect to your Sonar instance.

On successful tool creation, you're taken to the tool record page.

4. If you want to control access to the tool, add the groups that must be given access to the tool in the **Maintained by** field on the **Access** tab.

The tasks the users in the groups can perform depends on the roles assigned to them.

- DevOps Tool Owner role: Can view and edit the tool.
- DevOps App Owner role: Can view the tool and can associate, discover, import historical data, and modify pipeline steps (if applicable) of the tool's objects (such as plans, repositories, and pipelines).
- DevOps Administrator role: Can edit all tools.
- Other DevOps roles: Can view the tool.

**i Note:** Only groups containing users with DevOps roles are available for selection in the **Maintained by** field.

The **All App Owners can view and associate tool objects to applications** option becomes available for selection if you choose to restrict access to the tool. This option enables all users having the DevOps App Owner role to access the tool. If selected, they'll be able to view, associate, discover, import historical data, and modify pipeline steps (if applicable) of the tool's objects.

## What to do next

- [Configure SonarQube scans on Azure DevOps pipelines](#)
- [Configure SonarQube scans on Jenkins pipelines](#)

## Split.io integration with DevOps Change Velocity

Onboard your Split.io tool so that you can manage feature flags requests directly from your ServiceNow instance.

### Overview

Feature flags are used to turn code off and on in your production environment.

This integration extends the ServiceNow platform to manage the CHG approval process for feature flags and segments of Split. This capability enables managing updates to feature flags from DevOps Change Velocity.

- Split.io feature flag tool integration support enables discovery of workspaces, environments, segments, and feature flags.
- Users can set CHG request fields to enable Split.io for CHG control.
- Upon approval/rejection of a CHG request, the callback URL in Split.io for the split or segment is invoked to resume implementation of the update to the split and segment.

### Enable integration with ServiceNow DevOps in Split tool

Enable integration with ServiceNow DevOps in Split tool and restrict approvers to only the enabled integration for your environment.

#### Before you begin

Role required: admin in Split organization

#### Procedure

1. In Split tool marketplace, navigate to **Admin Settings > Integrations**.
2. Add ServiceNow DevOps in the **Integration Name** field, and select **Save**.  
A token is generated, which must be used while onboarding the Split.io tool.
3. Navigate to **Admin Settings > Workspaces > Choose the required workspace > Edit the Environment**.

4. In the **Change permissions** field, select **Require approvals for changes > Restrict who can approve**.
5. Select **Integrations** from the drop-down and select the integration configured in step 2 as, and select **Save**.

### Onboard Split.io to DevOps Change Velocity — Classic

Create a Split tool record to connect and discover workspaces, environments, segments, and feature flags from the connected Split tool.

#### Before you begin

Role required: sn\_devops.admin

#### Procedure

1. Create a tool record in DevOps to automatically connect to Split and get the webhook URL.
  - a. Navigate to **DevOps > Tools > Create New** and create a record.
  - b. Enter a **Tool Name** and fill in the tool details.

Tool Integration	Split
Tool URL	Split tool URL. For example: <code>https://api.split.io</code>
Tool Username	Split username
Tool Password / Access Token	Split password or access token. For information on creating a Split token, see <a href="#">Enable integration with ServiceNow DevOps in Split tool</a> .

**MID Server** is optional. Select MID Server for an on-premises tool that is attached to a MID Server. Application is automatically set to DevOps and capability is set to REST.

- c. Select **Submit**.  
 The tool is automatically **Connected Successfully** using a connection alias, and HTTP tool connection (basic authentication credential).

**i Note:** If you do not have global admin privileges for your tool (to allow automatic configuration of the webhook URL), you may need to have the tool admin user configure it for you (cut and paste the webhook URL into the tool configuration). Once the webhook is configured in the tool, **Enter Manual Configuration Mode** to connect to the tool manually, then exit.

On successful tool creation, you're taken to the tool record page.

2. If you want to control access to the tool, add the groups that must be given access to the tool in the **Maintained by** field on the **Access** tab.

The tasks the users in the groups can perform depends on the roles assigned to them.

- DevOps Tool Owner role: Can view and edit the tool.
- DevOps App Owner role: Can view the tool and can associate, discover, import historical data, and modify pipeline steps (if applicable) of the tool's objects (such as plans, repositories, and pipelines).
- DevOps Administrator role: Can edit all tools.
- Other DevOps roles: Can view the tool.

**i Note:** Only groups containing users with DevOps roles are available for selection in the **Maintained by** field.

The **All App Owners can view and associate tool objects to applications** option becomes available for selection if you choose to restrict access to the tool. This option enables all users having the DevOps App Owner role to access the tool. If selected, they'll be able to view, associate, discover, import historical data, and modify pipeline steps (if applicable) of the tool's objects.

3. Discover workspaces, environments, segments, and feature flags from the connected Split tool by selecting **Discover**.
4. Automatically configure the webhook URL in Split tool by selecting **Configure**. ServiceNow Integration will be activated on Split. Webhooks are auto-configured and notifications are sent from Split tool to DevOps.

#### **Example:**

The following example specifies how changes made in the Split tool are notified to ServiceNow DevOps through the webhook.

- Inbound events are created in ServiceNow for status (Requested, Approved, Rejected or Withdrawn) of the event.
- Feature Flag requests (**DevOps > Feature Flag > Feature Flag Requests**) are created or updated based on the status.
- A change request is created for every Feature Flag request, and work notes on the change request is updated with basic change details on the feature flag.

## **User-created integrations in DevOps Change Velocity**

User-created integrations are for integrating additional planning, coding, and test tools that are not available by default in the DevOps Change Velocity application.

The DevOps Change Velocity application includes tool definitions for integrating some common planning, coding, and [test tools](#), but you can also set up user-created integrations for additional tools in your DevOps environment.

**i Note:** User-created integrations for orchestration tools is not supported.

## **Integration objects**

DevOps tool integration consists of these objects.

## Tool capability actions

- **Connect** action:

When connecting, the subflow for the specific tool is called and the connection state is updated. The connection status message is shown on the form.

See [Connect capability subflow](#) for more details.

- **Discover** action:

When discovering, an import request record is created and the subflow for the specific tool is called (as defined in the Integration Capability record). **Detail** and **Status** fields in the Import Request record are updated with the number of items discovered, updated, and failed.

The transformed payload consists of an array of objects as a JSON string. Elements vary depending on the tool type.

See [Discover capability subflow](#) for more details.

**Note:** Pagination is not supported in the Discover action for user-created integrations.

- **Import** action:

Import action does not support historical import functionality.

- **Lookup** action:

The Lookup main flow is provided to support artifact tool type in a subflow created by your integration developer.

- **Notification** (webhook) action:

The source tool is configured manually (by your integration developer) to send raw data to the ServiceNow instance. The raw payload is then transformed into a standard JSON object using a subflow.

See the [Notification capability subflow](#) and the [DevOps - POST /devops/tool/{capability}](#) endpoint of the [DevOps API](#) for more details.

**Note:** If a subflow is not specified, default handling of notifications occurs (*original payload* is automatically copied to *transformed payload*).

This behavior is useful when the transactional data of the tool is supported by ServiceNow DevOps as is.

See the expected standard payloads in the [Notification capability subflow](#) for more details.

## DevOps integration configuration overview

Tool integration configuration can be completed by your integration developer and your DevOps admin.

Integration developer

- Create a tool integration record in DevOps to define the tool you are integrating (source tool).
- Create a Flow Designer [subflow](#) to collect and transform data from the tool you are integrating (source tool).
- Create a tool capability mapping record in DevOps to map the tool integration record to the tool type capability.

**i Note:** Notifications (webhook) capability is supported. Connect and discover capabilities are also supported.

- Create an integration capability record in DevOps to specify the action for the tool type capability.

DevOps admin

- Create a (planning, coding, or test) tool record in DevOps to connect to the tool you are integrating (source tool).

**i Note:** The tool integration record must be specified in the **Tool** field of the tool record.

- Configure the source tool with the webhook and credentials.

## Inbound events

An inbound event serves as a staging area for the notifications flow that supports reprocessing of failed payloads. Meaning, a record in an error state from a failed integration or transformation can be retried.

If an Inbound Event record is in the **Error** state, the flow was not able to insert the record successfully into the core DevOps tables.

Common errors can be resolved with these actions.

### Inbound event error states

Error	Action
Missing required fields	The transformed payload does not match the standard payload. Refer to the standard and JSON payloads provided.
Repository not marked for tracking	The commit cannot be inserted. The DevOps admin needs to track the repository.
[Subflow] has not been published within application scope [app_scope]	The subflow is created but not published yet.

## Inbound event error states (continued)

Error	Action
Timeout exception	<p>The subflow takes more time than the value set in the property:  <code>com.glide.hub.flow_api.default_execution_time</code></p> <p>See <a href="#">FlowAPI - executeSubflowQuick(String name, Map inputs, Number timeout)</a> for more details.</p> <p><b>Note:</b> The execution of the subflow exceeds the value set in the <b>Timeout</b> field in the Integration Capability record.</p>
Did not find a matching subflow for notification capability and [tool_integration_sys_id] tool integration	<p>The flow was not able to find the matching subflow.</p> <p>Verify the integration setup procedure.</p>
Payload does not match the expected capability.	<p>The <b>Original payload</b> (payload being sent) is a different <b>Capability</b> type than the tool type capability configured in your tool capability mapping.</p> <p>The payload type must match the tool type capability configured in your tool integration.</p>

**Note:** An inbound event record is not created when any of the following conditions occur:

- Source tool has not passed the tool ID as a query parameter.
- Source tool has passed on a tool ID, but there is no matching tool ID in the instance.

## Tool mappings

A tool can be mapped to multiple capabilities.

Tool Integration	Tool Type Capability	Tool Capability Mapping
<ul style="list-style-type: none"> <li>• Agile Development 2.0</li> <li>• Azure DevOps</li> <li>• Bitbucket</li> <li>• GitHub</li> <li>• GitHub Enterprise</li> <li>• GitLab</li> </ul>	<ul style="list-style-type: none"> <li>• Plan</li> <li>• Code</li> <li>• Orchestration</li> <li>• Artifact</li> <li>• Test</li> </ul>	<p>Plan</p> <ul style="list-style-type: none"> <li>• Agile Development 2.0 - Plan</li> <li>• Azure DevOps - Plan</li> <li>• Jira - Plan</li> <li>• Rally - Plan</li> </ul> <p>Code</p>

Tool Integration	Tool Type Capability	Tool Capability Mapping
<ul style="list-style-type: none"> <li>• Jenkins</li> <li>• Jira</li> <li>• Rally</li> </ul>		<ul style="list-style-type: none"> <li>• Azure DevOps - Code</li> <li>• Bitbucket - Code</li> <li>• GitHub - Code</li> <li>• GitHub Enterprise - Code</li> <li>• GitLab - Code</li> </ul> <p>Orchestration</p> <ul style="list-style-type: none"> <li>• Azure DevOps - Orchestration</li> <li>• Jenkins - Orchestration</li> <li>• GitLab - Orchestration</li> </ul> <p>Test</p> <ul style="list-style-type: none"> <li>• Azure DevOps - Test</li> <li>• Jenkins - Test</li> </ul>

A tool capability mapping can be mapped to multiple actions.

Tool Capability Mapping	Tool Action	Integration Capability
<ul style="list-style-type: none"> <li>• Agile Development 2.0 - Plan</li> <li>• Azure DevOps - Plan</li> <li>• Azure DevOps - Code</li> <li>• Azure DevOps - Orchestration</li> <li>• Bitbucket - Code</li> <li>• GitHub - Code</li> <li>• GitHub Enterprise - Code</li> <li>• GitLab - Code</li> <li>• GitLab - Orchestration</li> <li>• Jenkins - Orchestration</li> <li>• Jira- Plan</li> <li>• Rally-Plan</li> </ul>	<ul style="list-style-type: none"> <li>• Connect</li> <li>• Discover</li> <li>• Import</li> <li>• Lookup</li> <li>• Notification</li> </ul>	<p>Agile Development 2.0</p> <ul style="list-style-type: none"> <li>• Agile Development 2.0 - Plan - Connect</li> <li>• Agile Development 2.0 - Plan - Discover</li> <li>• Agile Development 2.0 - Plan - Import</li> <li>• Agile Development 2.0 - Plan - Notification</li> </ul> <p>Azure DevOps</p> <ul style="list-style-type: none"> <li>• Azure DevOps - Plan - Connect</li> <li>• Azure DevOps - Plan - Discover</li> <li>• Azure DevOps - Plan - Notification</li> <li>• Azure DevOps - Code - Discover</li> <li>• Azure DevOps - Code - Notification</li> <li>• Azure DevOps - Orchestration - Discover</li> <li>• Azure DevOps - Orchestration - Notification</li> </ul> <p>Bitbucket</p> <ul style="list-style-type: none"> <li>• Bitbucket - Code - Connect</li> <li>• Bitbucket - Code - Discover</li> <li>• Bitbucket - Code - Import</li> <li>• Bitbucket - Code - Notification</li> </ul>

Tool Capability Mapping	Tool Action	Integration Capability
		<p>GitHub</p> <ul style="list-style-type: none"> <li>• GitHub - Code - Connect</li> <li>• GitHub - Code - Discover</li> <li>• GitHub - Code - Import</li> <li>• GitHub - Code - Notification</li> </ul> <p>GitHub Enterprise</p> <ul style="list-style-type: none"> <li>• GitHub - Code - Connect</li> <li>• GitHub - Code - Discover</li> <li>• GitHub - Code - Import</li> <li>• GitHub - Code - Notification</li> </ul> <p>GitLab</p> <ul style="list-style-type: none"> <li>• GitLab - Code - Connect</li> <li>• GitLab - Code - Discover</li> <li>• GitLab - Code - Notification</li> <li>• GitLab - Orchestration - Notification</li> </ul> <p>Jenkins</p> <ul style="list-style-type: none"> <li>• Jenkins - Orchestration - Connect</li> <li>• Jenkins - Orchestration - Discover</li> <li>• Jenkins - Orchestration - Import</li> <li>• Jenkins - Orchestration - Notification</li> </ul> <p>Jira</p> <ul style="list-style-type: none"> <li>• Jira - Plan - Connect</li> <li>• Jira - Plan - Discover</li> <li>• Jira - Plan - Import</li> <li>• Jira - Plan - Notification</li> </ul> <p>Rally</p> <ul style="list-style-type: none"> <li>• Rally-Plan - Discover</li> <li>• Rally-Plan - Import</li> <li>• Rally-Plan - Notification</li> <li>• Rally-Plan - Connect</li> <li>• Rally-Plan - Validate</li> </ul>

Multiple test types can be mapped to each tool integration.

Test Type	Tool Integration	Test Type Mapping
Unit: JUnit Functional: <ul style="list-style-type: none"> <li>• Integration</li> <li>• Regression</li> <li>• Smoke</li> <li>• System</li> <li>• User Acceptance</li> </ul> Performance: Load	<ul style="list-style-type: none"> <li>• Azure DevOps</li> <li>• Jenkins</li> </ul>	Azure DevOps <ul style="list-style-type: none"> <li>• Azure DevOps - JUnit</li> <li>• Azure DevOps - Integration</li> <li>• Azure DevOps - Regression</li> <li>• Azure DevOps - Smoke</li> <li>• Azure DevOps - System</li> <li>• Azure DevOps - User Acceptance</li> <li>• Azure DevOps - Load</li> </ul> Jenkins <ul style="list-style-type: none"> <li>• Jenkins - JUnit</li> <li>• Jenkins - Integration</li> <li>• Jenkins - Regression</li> <li>• Jenkins - Smoke</li> <li>• Jenkins - System</li> <li>• Jenkins - User Acceptance</li> <li>• Jenkins - Load</li> </ul>

## DevOps test tool integration

Test tool integration lets you view test results in DevOps for Jenkins, Azure DevOps, and GitLab unit, functional, and performance tests.

JUnit test type integration is supported for Jenkins and Azure DevOps.

JUnit test type integration is supported for GitLab.

**Note:** For other test types, use the [DevOps - POST /devops/tool/{capability}](#) endpoint of the [DevOps API](#).

- Selenium tests run and published using TestNG are reported by the Jenkins plugin for ServiceNow DevOps.
- Test type categorization is supported.
- Additional tests results reported by tools, such as Apache JMeter, can be processed in DevOps using custom Flow Designer subflows (no pipeline changes required).

Category	Test type
Unit	JUnit (default)  You can change the default test type by modifying the

Category	Test type
	<code>[sn_devops.default_test_type]</code> DevOps property.
Functional	<ul style="list-style-type: none"> <li>Integration</li> <li>Regression</li> <li>Smoke</li> <li>System</li> <li>User Acceptance</li> </ul>
Performance	Load

## Test type mapping

The test type mapping connects the test type and entity being tested with the DevOps tool (**DevOps > Integrations > Test Type Mappings** module.)

An accurate test type mapping ensures that the test type always appears as intended in the test summary results.

Field	Description
Test type	<ul style="list-style-type: none"> <li>JUnit</li> <li>Integration</li> <li>Regression</li> <li>Smoke</li> <li>System</li> <li>User Acceptance</li> <li>Load</li> </ul>
DevOps Entity Id	<p>Table name</p> <p>DevOps table name that contains the entity linked to the test results (in the test report payload).</p> <ul style="list-style-type: none"> <li>Step [sn_devops_step]</li> <li>Pipeline [sn_devops_pipeline]</li> </ul> <p><b>Note:</b> Only DevOps Step and Pipeline tables are supported.</p>
Document	Name of the entity specified in the selected table.

Field	Description
	For example, the name of the step, pipeline, artifact, or package.
Test File Paths  (Jenkins tests only)	Path to the generated test result file on the Jenkins server.  This is useful so test reports with attributes that don't conform to JUnit or TestNG implementation, such as JMeter for example, can still be leveraged by DevOps.  Separate multiple files by a comma.
	<p><b>Note:</b> You must use a Flow Designer subflow to transform a raw test payload.</p>
Tool integration	Tool that's running the test.
DevOps Table	DevOps table that corresponds to the table name in the <b>DevOps Entity Id</b> setting.

### DevOps test type mapping

## Transforming a raw test payload

You can transform a raw test report (a report containing attributes that do not conform to JUnit or TestNG implementation), such as JMeter for example, by configuring the **DevOps Test Subflow Policy** decision table to call a custom subflow (**Decision Tables > Decision Tables** module).

**Note:** You must create the custom Flow Designer subflow that transforms the raw test payload.

If there is more than one test type in a performance stage, you can use the **DevOps Test Type Policy** decision table to configure the test type for each test so the test result payloads are transformed correctly.

### DevOps decision tables

## DevOps decision tables

Decision table	Purpose	Configuration
DevOps Test Subflow Policy	<p>To automatically call a custom subflow that transforms the raw payload received by the tool.</p> <p>Decision inputs:</p> <ul style="list-style-type: none"> <li>• Test Result Payload</li> <li>• Test Type</li> </ul>	<p>Create a decision that specifies the custom subflow to call when the raw payload is received.</p> <p>Set the conditions to contain fields that would be included in the raw payload.</p> <p>For example, to call Jenkins BZ Performance Test custom subflow:</p> <p>Conditions:</p> <ul style="list-style-type: none"> <li>• Test Type is Load (Load is the test type configured for performance tests)</li> <li>• Test Result Payload contains throughput</li> <li>• Test Result Payload contains concurrency</li> </ul> <p>Answer: Flow: Jenkins BZ Performance Test</p>
DevOps Test Type Policy	<p>To automatically set test types when more than one type of test is configured in a performance test stage.</p> <p>This is necessary so the results for the second test type get transformed correctly.</p> <p>For example, when both a Load performance test and a JUnit performance test are mapped in the same DevOps step, the JUnit test results won't get formatted correctly unless a decision is created.</p> <p>Decision inputs:</p>	<p>Create a decision for each type of test in the performance test stage to set the test type.</p> <p>Load test:</p> <ul style="list-style-type: none"> <li>• Conditions: <ul style="list-style-type: none"> <li>◦ Step is Perf Tests</li> <li>◦ Test Result Payload contains throughput</li> <li>◦ Test Result Payload contains concurrency</li> </ul> </li> <li>• Answer: TestType: Load</li> </ul> <p>JUnit test:</p>

## DevOps decision tables (continued)

Decision table	Purpose	Configuration
	<ul style="list-style-type: none"> <li>• Step</li> <li>• Test Result Payload</li> <li>• Tool Integration</li> <li>• Pipeline</li> </ul>	<ul style="list-style-type: none"> <li>• Conditions: <ul style="list-style-type: none"> <li>◦ Step is Perf Tests</li> <li>◦ Test Result Payload does not contain throughput</li> <li>◦ Test Result Payload does not contain concurrency</li> </ul> </li> <li>• Answer: TestType: JUnit</li> </ul>

### DevOps multiple performance test types

### DevOps multiple test type mappings

### DevOps decision table decision

## Test summary results

You can view test summary results these ways.

- **DevOps > Test Results** module (Test Summaries and Performance Test Summaries).
- [DevOps change request](#) - Test Results related list.
- [DevOps Pipeline UI](#) - Quality tile.

### DevOps performance test summary example

## Expected standard JSON Notification capability payload - Test tool

Functional:

```
{
  "name": "CorpSite-selenium#55",
  "duration": 78.802,
  "passedTests": 4,
  "failedTests": 0,
  "skippedTests": 0,
  "blockedTests": 0,
  "totalTests": 4,
  "startTime": "2020-06-30T18:12:31Z",
  "finishTime": "2020-06-30T18:12:31Z",
  "passingPercent": 100,
}

// Use Artifact OR Package OR Build + Stage + PipelineName Attributes
```

Send only one Attribute combination. For example, send Attributes of either Artifact or Package, or the combination of Build + Stage + PipelineName.

If you send more than one Attribute, priority is given in the following order and the low priory one is ignored. For example, if you send attribute for both packages and artifacts, then attribute of package is considered and the attribute of artifacts is ignored.

- 1.packages
- 2.artifcats
- 3.buildNumber + stageName + pipelineName

```
"packages": [{"name": "CorpSite-pkg1"}],  
"artifacts": [{"name": "CorpSite-artifact", "version": "1.0.0"}],  
"buildNumber": "55",  
"stageName": "test",  
"pipelineName": "CorpSite-selenium",  
}
```

#### Notes:

- The pipelineName attribute value must be same as the value in the **Orchestration pipeline** field of the Pipeline [sn\_devops\_pipeline] table.
- The stageName attribute value must be same as the value in the **Orchestration stage** field of the Step [sn\_devops\_step] table.

#### Performance:

```
{
"name": "Performance Tests",
"url": "http://abc.com",
"startTime": "2020-06-30T18:12:31Z",
"finishTime": "2020-06-30T18:12:31Z",
"duration": 78.802,
"maximumVirtualUsers": "",  

"throughput": "",  

"maximumTime": "",  

"minimumTime": "",  

"averageTime": "",  

"ninetyPercent": "",  

"standardDeviation": "",

// Use Artifact OR Package OR Build + Stage + PipelineName Attributes
```

Send only one Attribute combination. For example, send Attributes of either Artifact or Package, or the combination of Build + Stage + PipelineName.

If you send more than one Attribute, priority is given in the following order and the low priory one is ignored. For example, if you send attribute for both packages and artifacts, then attribute of package is considered and the attribute of artifacts is ignored.

- 1.packages
- 2.artifcats
- 3.buildNumber + stageName + pipelineName

```

"packages": [{"name": "CorpSite-pkg1"}],
"artifacts": [{"name": "CorpSite-artifact", "version": "1.0.0"}],
"buildNumber": "55",
"stageName": "test",
"pipelineName": "CorpSite-Performance",
}

```

**Notes:**

- The pipelineName attribute value must be same as the value in the **Orchestration pipeline** field of the Pipeline [sn\_devops\_pipeline] table.
- The stageName attribute value must be same as the value in the **Orchestration stage** field of the Step [sn\_devops\_step] table.

## Configure a test tool in DevOps

Configure a test tool in DevOps to view unit, functional, and performance test results.

### Before you begin

Role required: sn\_devops.admin

### Procedure

1. Navigate to **All > DevOps > Integrations > Test Type Mappings** and create a record to map the test type to the integration tool.
2. In the Test Type Mapping record, use the search icon () to select **Test type** and **Tool integration** field values.

#### DevOps test type mapping

3. Click the search icon () in the **DevOps Entity Id** field, and fill in the test details.

Table name	DevOps table name that contains the entity linked to the test results.
Document	Name of the entity specified in the selected table.

#### DevOps test mapping entity ID

See the [Test type mapping](#) section for more details.

4. Optional: To capture the raw test payload of non-JUnit or -TestNG tests, enter an .xml filename and path (comma separate multiple files).  
(Optional) You can transform a raw payload using decision tables and a custom Flow Designer subflow.

**i Note:** You must create the custom subflow that transforms the raw payload.

See the [Transforming a raw test payload](#) section for more details on configuring decision tables.

- Run the test and view test results by navigating to **DevOps > Test Results** (Test Summaries and Performance Test Summaries).

### DevOps Test Summaries example

#### Example:

#### Scripted pipeline example with test configuration

#### Create a DevOps tool integration

To create a DevOps tool integration, your integration developer configures DevOps tool integration settings, and a Flow Designer subflow to collect and transform data from the source tool. Then your DevOps admin configures your DevOps tool connections.

#### Before you begin

**Note:** When creating an integration as a scoped app, the system admin must assign these roles to the integration developer so the integration developer is able to create tool integration and integration capability records for the specific scope.

- Developer role for the scoped app
- DevOps admin role

Role required: sn\_devops.admin

#### About this task

Creating a DevOps tool integration procedure involves configuration by both your integration developer and your DevOps admin.

- Your integration developer creates a tool integration record in DevOps, a Flow Designer [subflow](#), a tool capability mapping, and an integration capability record in DevOps to map the capabilities and actions together.

**Note:** Notification, connect, and discover capabilities are supported.

- Your DevOps admin sets up DevOps connections (planning or coding tool), and configures the source tool with the webhook and credentials.

This procedure provides detailed steps to create your DevOps tool integration.

#### Procedure

##### 1. Integration developer:

Configure the source tool integration capabilities and actions, and a subflow.

- Navigate to **DevOps > Integrations > Tool Integrations** and create a record to define the tool you are integrating (source tool).

**Note:** Do not edit the tool integration records provided with the DevOps application.

#### DevOps tool integration

Tool Label	Sample Code Tool
------------	------------------

Table	Code Tool [sn_devops_tool]
Use packageable integrations	Selected
Integration version	1.0
Active	Selected

- b. Navigate to **Flow Designer > Designer** and [create a subflow](#) to collect and transform data from the tool you are integrating (source tool).

**Note:** The **Run As** field must be set to System User, and the **Inputs** label must be set to current variable.

#### Notification subflow properties

Name	Code Tool Notification
Application	Sample Integration App
Accessible From	All application scopes
Description	Code tool for integration app
Run As	System User

The subflow must contain Get More Data via API calls, and/or transform the original payload. Copy the transformed payload into the inbound events record.

**Note:** Do not edit the DevOps main flow.

- c. Navigate to **DevOps > Integrations > Tool Capability Mappings** and create a record to map the tool integration record to the tool type capability.

#### DevOps tool capability mapping

Tool integration	Sample tool
Tool type capability	Code

- d. Navigate to **DevOps > Integrations > Integration Capabilities** and create a record to specify the action for the tool capability mapping.

**Note:** Do not edit the integration capability records provided with the DevOps application.

#### DevOps integration capability

Tool integration	Sample Code Tool
Capability mapping	Sample Code Tool-Code  <b>Note:</b> Do not edit tool type capability records.
Action	Notification

	<p><b>Note:</b> Do not edit tool action records.</p>
Active	Selected
Timeout (ms)	<p>Timeout for the corresponding subflow. If execution of the subflow exceeds this value, a timeout exception occurs.</p> <p>Value is in milliseconds (ms). Default is 45,000 (45 seconds).</p>
Subflow name	<p>x_snc_sample_integ.code_tool_notification</p> <p>The name is prefixed by the scope name and a dot (.) before the actual subflow name.</p> <p>For example, given:</p> <ul style="list-style-type: none"> <li>▪ connect_code_tool subflow</li> <li>▪ my_app_scope scope</li> </ul> <p>The value for this field is my_app_scope.connect_code_tool.</p> <p><b>Note:</b> If the <b>Subflow name</b> field is left blank for a Notification capability, default handling of notifications occurs.</p> <p>See <a href="#">Notification tool capability action</a>.</p>
Domain	global

## 2. DevOps admin:

Configure the connection from DevOps to the source tool.

### a. The tool record must contain:

- Reference to the tool integration record created by the integration developer (in the **Tool** field)
- Tool label
- Connection alias (connection and credential)

### b. Copy the notification (webhook) created on the DevOps planning tool to the source tool service hook of the notification endpoint and set the credentials to devops.integration.user.

You can view the state of integration events in the Inbound Event list (**DevOps > Administration > Inbound Events**).

The inbound event record state is set to **Processed** once the object has been inserted into the DevOps Core table. Event states include New, In Progress, Processed, Unmatched, and Error.

3. If the tool integration record and the subflow are created in a different scope, the DevOps admin must create two new Cross scope privileges records to allow the app to access the inbound events table.

Navigate to **System Applications > Application Cross-Scope Access** and create read and write cross scope privileges records to allow your app to access the inbound events table.

Field	Read	Write
Source Scope	Auto populated based on the current application	
Target Scope	DevOps	
Target Name	sn_devops_inbound_event	
Target Type	Table	
Operation	Read	Write
Status	Allowed	

### Example:

**DevOps tool integration**

**Flow Designer subflow properties**

**Flow Designer subflow**

**Flow Designer DevOps Integration - Notification flow**

**DevOps tool capability mapping**

**Integration capability**

**Planning tool**

**Cross scope access records (read and write)**

### Creating DevOps subflows

For user-created integrations, create a DevOps Flow Designer subflow to collect and transform data from the tool you are integrating.

ServiceNow [Flow Designer](#) is a Now Platform feature your integration developer can use to automate processes using a sequence of actions. Trigger conditions start the flow, and variables pass information between actions.

See Flow Designer [Subflows](#) for information on how to create a subflow.

## Notification capability subflow

Item	Expected value
Inputs	Label: current Type: Reference.Inbound Event
Outputs	N/A

Transform the original payload, and save the final payload in the **transformed\_payload** field.

Expected standard JSON Notification capability payload - Planning tool

**i Note:** The version attribute in the JSON payload is optional. Even if you do not provide the version attribute, the payload is processed successfully.

```
{
  "id": "STR1234",
  "type": "Story",
  "shortDescription": "Planning API Spec",
  "state": "In-progress",
  "createdDateTime": "1970-01-01T08:15:30-05:00",
  "assignedTo": {
    "name": "Leo Neo",
    "userName": "lenn",
    "id": "3fa85f64-5717-4562-b3fc-2c963f66afa6",
    "email": "lenn@smithworksinc.com"
  },
  // The Version attribute is optional
  "version": {
    "id": "REL1234",
    "shortDescription": "APIs Release",
    "createdDateTime": "1970-01-01T08:15:30-05:00",
    "app": {
      "id": "PRODUCT1234",
      "shortDescription": "Mobile UI",
      "createdDateTime": "1970-01-01T08:15:30-05:00",
      "url":
        "https://jira.com/mycompany/browse/PRODUCT-125"
    },
    "url": "https://jira.com/mycompany/browse/REL-125"
  },
  "app": {
    "id": "PRODUCT1234",
    "shortDescription": "Mobile UI",
    "createdDateTime": "1970-01-01T08:15:30-05:00",
    "url": "https://jira.com/mycompany/browse/PRODUCT-125"
  },
  "url": "https://jira.com/mycompany/browse/HALOKEY-25"
}
```

Expected standard JSON Notification capability payload - Coding tool

```
{
  "id": "3fa85f64-5717-4562-b3fc-2c963f66afa6",
```

Expected standard JSON Notification capability payload - Orchestration tool

```
{  
  "toolId": "bc1d9454dbdb0810ae77f3c61d9619d1",  
  "buildNumber": "100",  
  "nativeId": "HILR/Prod #100",  
  "name": "HILR/Prod",  
  "id": "HILR/Prod #100".
```

```

    "url": "https://dev.azure.com/lenn/CorpSite-ADO/_build/results?buildId=100#Prod/",
    "isMultiBranch": "false",
    "orchestrationTaskUrl": "https://dev.azure.com/lenn/CorpSite-ADO/_build?name=HILR#Prod",
    "orchestrationTaskName": "CorpSite-ADO/HILR#Prod",
    "upstreamTaskUrl": "https://dev.azure.com/lenn/CorpSite-ADO/_build/results?buildId=100#UAT/",
    "upstreamId": "CorpSite-ADO/HILR#UAT",
    "result": "building",
    "startDateTime": "2020-03-20 22:59:27"
}

```

Expected standard JSON Notification capability payload - Test tool

Functional:

```

{
  "name": "CorpSite-selenium#55",
  "duration": 78.802,
  "passedTests": 4,
  "failedTests": 0,
  "skippedTests": 0,
  "blockedTests": 0,
  "totalTests": 4,
  "startTime": "2020-06-30T18:12:31Z",
  "finishTime": "2020-06-30T18:12:31Z",
  "passingPercent": 100,

  // Use Artifact OR Package OR Build + Stage + PipelineName
  Attributes
  "packages": [{"name": "CorpSite-pkg1"}],
  "artifacts": [{"name": "CorpSite-artifact", "version": "1.0.0"}],
  "buildNumber": "55",
  "stageName": "test",
  "pipelineName": "CorpSite-selenium",
}

```

Performance:

```

{
  "name": "Performance Tests",
  "url": "http://abc.com",
  "startTime": "2020-06-30T18:12:31Z",
  "finishTime": "2020-06-30T18:12:31Z",
  "duration": 78.802,
  "maximumVirtualUsers": "",
  "throughput": "",
  "maximumTime": "",
  "minimumTime": "",
  "averageTime": "",
  "ninetyPercent": "",
  "standardDeviation": ""
}

```

```
// Use Artifact OR Package OR Build + Stage + PipelineName
Attributes
"packages": [{"name": "CorpSite-pkg1"}],
"artifacts": [{"name": "CorpSite-artifact", "version": "1.0.0"}],
"buildNumber": "55",
"stageName": "test",
"pipelineName": "CorpSite-Performance",
}
```

Expected standard JSON Notification capability payload - Artifact tool

Pipelines:

```
{
  "artifacts": [
    {
      "name": "acm.jar",
      "version": "1.82",
      "semanticVersion": "1.82.0",
      "repositoryName": "acm-repo"
    }
  ],
  "pipelineName": "testMultiBranch/master",
  "taskExecutionNumber": "82",
  "stageName": "buildmbmaster",
  "branchName": "master"
}
```

Jenkins Freestyle/Maven Project:

```
{
  "artifacts": [
    {
      "name": "mav1.jar",
      "version": "1.11",
      "semanticVersion": "1.11.0",
      "repositoryName": "mav-repo"
    }
  ],
  "projectName": "maven-test-proj",
  "taskExecutionNumber": "11"
}
```

**DevOps Notification capability flow diagram example - Coding tool**

## Connect capability subflow

The Connect capability is supported.

Item	Expected value
Inputs	Label: current Type: Reference.DevOps Tool

Item	Expected value
	Tool record for which the <b>Connect</b> button action is clicked.
Outputs  See <a href="#">subflow outputs</a> .	<ul style="list-style-type: none"> <li>Label: connected</li> </ul> <p>Flag indicating the success or failure of the connection made to the target tool (true/false).</p> <ul style="list-style-type: none"> <li>Label: errormessage</li> </ul> <p>String message displayed on the form for connection failure. The variable is an empty string if the connection is successful.</p>

Connect errors shown on the DevOps tool form:

Connection failed

Subflow was executed successfully but the connection could not be made.

Error: Failed to get failure details from the tool specific connect flow

Subflow execution failed for an unknown reason.

Error updating the tool connect status

The connection\_state attribute could not be updated for an unknown reason.

#### DevOps Connect capability flow diagram example - Orchestration tool

## Discover capability subflow

The Discover capability is supported.

Item	Expected value
Inputs	<p>Label: current</p> <p>Type: Reference.DevOps Tool</p>
Outputs  <b>Note:</b> Pagination is not supported for the discover action.  See <a href="#">subflow outputs</a> .	<p>Label: discoverpayload</p> <p>Type: Array of objects as JSON string.</p> <pre>(JSON.stringify([{}, {}]))</pre> <p>Planning tool:</p> <pre>[   {     "id": "REL1234567",     "name": "REL NUMBERS",      "url": "https://jira.com/vult/browse/REL1234567",   } ]</pre>

Item	Expected value
	<pre>     "nativeId": "1790e6cc-085b-4529-9cb8-47f393182226", }, {     "id": "TOR67",     "name": "TOR 67",      "url": "https://jira.com/welp/browse/TOR67",     "nativeId": "482ce864-085b-4529-9cb8-47f393767eb2" } ]</pre>

Coding tool:

```
[ {
    "name": "nvm_repo",
    "url": "https://github.com/nvm_repo/"
},
{
    "name": "golang_util",
    "url": "https://github.com/golang_util/"
}];
```

The Discover main flow is triggered during import request record creation. An import request has these states and messages.

State	Message
Requested	--
Processing	--
Completed	Updated <number> object(s) Found <number> objects with invalid toolId Found <number> objects failed validation
Error	<ul style="list-style-type: none"> <li>• Unable to parse payload string from Subflow</li> <li>• discoverpayload is expected to be an array of objects</li> <li>• ImportRequest record does not have reference to tool table</li> </ul>

State	Message
Paused	--
Canceled	--
Unmatched	--

Expected standard JSON Discover capability payload - Planning tool

```
[
{
  "id": "REL1234567",
  "name": "REL NUMBERS",
  "url": "https://jira.com/vult/browse/REL1234567",
  "nativeId": "1790e6cc-085b-4529-9cb8-47f393182226"
},
{
  "id": "TOR67",
  "name": "TOR 67",
  "url": "https://jira.com/welp/browse/TOR67",
  "nativeId": "482ce864-085b-4529-9cb8-47f393767eb2"
}
];
```

Expected standard JSON Discover capability payload - Coding tool

```
[
{
  "name": "nvm_repo",
  "url": "https://github.com/nvm_repo/",
  "externalCreatedDate": "2019-06-19 00:37:16"
},
{
  "name": "golang_util",
  "url": "https://github.com/golang_util/",
  "externalCreatedDate": "2019-06-19 00:37:16"
}
];
```

Expected standard JSON Discover capability payload - Orchestration tool

```
{
  "orchestrationTasks": [
    {
      "name": "Build_AP1_1",
      "url": "https://jenkins.wsf.xyz/job/Build_AP1_1",
      "projectName": "Build_AP1_1"
    },
    {
      "name": "CI_CD_Jenkins",
      "url": "https://pt1.jenkins.com/job/CI_CD_Jenkins",
      "projectName": "CI_CD_Jenkins"
    },
    {
      "name": "CI_deploy",
      "url": "https://pt2.jenkins.com/job/CI_deploy",
      "projectName": "CI_deploy"
    }
  ],
}
```

```

"pipelines": [
    {
        "name": "Build_AP_C_1",
        "url": "https://jenkins.wsf.xyz/job/Build_AP_C_1",
        "projectName": "Build_AP_C_1"
    },
    {
        "name": "CI_CD_Jenkins",
        "url": "https://pt1.jenkins.com/job/CI_CD_Jenkins",
        "projectName": "CI_CD_Jenkins"
    },
    {
        "name": "CI_deploy",
        "url": "https://pt2.jenkins.com/job/CI_deploy",
        "projectName": "CI_deploy"
    }
]
}

```

### DevOps Discover capability flow diagram example - Planning tool

## Update Import Request Flow Designer action

You can use the Update Import Request action in your Discover subflow to modify the Import Request record state, if desired.

Input label	Type
current	Reference.Import Request
state	<p>(String)</p> <ul style="list-style-type: none"> <li>• requested</li> <li>• processing</li> </ul> <p><b>i Note:</b> Flow execution is stopped (once the return is received from the subflow) when the state is set to processing.</p> <ul style="list-style-type: none"> <li>• completed</li> <li>• error</li> <li>• paused</li> <li>• canceled</li> <li>• unmatched</li> </ul>
details	(String)

## Update third-party tool credentials in DevOps Change Velocity

If your tool credential has changed, you must update the credentials in your ServiceNow instance to avoid getting disconnected.

### Before you begin

Role required: sn\_devops.admin

### Procedure

Update your tool credentials using one of the following methods.

Option	Steps
From Workspace	<ul style="list-style-type: none"> <li>a. Navigate to <b>Workspaces &gt; DevOps Change Workspace &gt; Tools &gt; &lt;tool type&gt;</b>.</li> <li>b. Select the tool for which you want to update your credentials.</li> <li>c. In the tool record, select <b>More Actions( ) &gt; Update credentials</b>.</li> <li>d. Enter the updated credentials in the corresponding field.</li> </ul> <p><b>Note:</b> You can change your password/token, username, or authentication type.</p> <ul style="list-style-type: none"> <li>e. Select <b>Check permissions</b> to verify if the required permissions are available.</li> <li>f. Select <b>Update credentials</b> to update the credentials. The tool will be in the disconnected state.</li> <li>g. Select <b>Connect</b> to establish the connection.</li> </ul>
From Classic	<ul style="list-style-type: none"> <li>a. Navigate to <b>All &gt; DevOps &gt; Tools &gt; &lt;tool type&gt;</b>.</li> <li>b. Select the tool for which you want to update your credentials.</li> <li>c. From the Tool connections related list, select the credential record.</li> </ul>

Option	Steps
	<p>d. Enter the updated credentials in the corresponding field, and select <b>Save</b>. The tool will be in the disconnected state.</p> <p>e. Select <b>Connect</b> to establish the connection.</p>

### Notifications on tool credential expiration

Notifications are sent to tool users on expiration of tool credentials to alert them. Notifications are also sent proactively before the expiration of tool credentials for GitHub tools created with basic authentication. This enables tool users with the sn\_devops.tool\_owner or sn\_devops.admin roles to update the tool credentials and prevent any loss of data.

A universal task is created and assigned to users with the sn\_devops.tool\_owner role who are part of any user group specified in the **Maintained by** field, and any user with the sn\_devops.admin role. They will be notified of the universal task through notification (in the bell icon), email, and an open task in the workspace home page.

Notifications are also displayed in the tool record in the form of a banner message to any user with access to the tool when the tool credentials expire. But the credentials can be updated only by users with the sn\_devops.tool\_owner or sn\_devops.admin role.

The credentials expiration check happens in the system every one hour. If your tool credentials have expired, it might take a maximum of one hour for the system to send notifications.

For GitHub tools created with basic auth, notifications are also sent proactively before the expiration of tool credentials. Apart from the universal task and banner notification, a field message is also displayed in the case of proactive expiration notifications. You can set the number of days before which expiration notifications must be sent in the **Number of days before tool credential expiry to assign a universal task and notify (if applicable)** property. By default, it is set to 3 days. To stop sending proactive notifications, select 0 as the value for this property.

If you want to stop sending notifications for expired credentials after expiry, disable the *Assign a universal task and notify to update tool credentials when expired* property. For more information, see [DevOps Change Velocity Properties](#).

The following types of notifications are sent:

#### Universal task

A universal task is created and notifications are sent to users with the sn\_devops.tool\_owner role who are part of any user group specified in the **Maintained by** field, and any user with the sn\_devops.admin role.

#### Banner message

A banner message is displayed on the tool record to all users with access to the tool record.

### Field message

A field message is displayed on the **Credentials expiration** field in the tool record for a GitHub tool created with basic auth.

When the credentials of your tool expire, the tool gets disconnected. You can select the **Update credentials** link in the notifications, and update your tool credentials. After the credentials are updated, connect to the tool again to start receiving data. For information on updating tool credentials, see [Update third-party tool credentials in DevOps Change Velocity](#).

## Create an application in DevOps Change Velocity

In DevOps Change Velocity, an application collects all the up-to-date data that connected tools send about plans, repositories, and pipelines. You must create an application to enable traceability for user stories, commits, test results, and more. Associating plans, repositories, and pipelines to an application also enables pipeline modeling, change governance, and metric reporting.

### Before you begin

[Set up system accounts in DevOps Change Velocity](#).

Role required: sn\_devops.admin or sn\_devops.app\_owner

### About this task

An application is needed to group plans, repositories, and pipelines together which enable tracking automatically and provide associations for DevOps data such as commits linked to work items. You can link the application with other ServiceNow products, including DevOps Config.

In the DevOps Change, this procedure generates a single DevOps application record (sn\_devops\_app) that is used by both DevOps and DevOps Config (if installed). When DevOps Config is installed, the DevOps application record is also linked to a CDM application record (sn\_cdm\_application). The CSDM application model (name, owner, manufacturer) functions as the link between ServiceNow DevOps products.

### Procedure

1. You can create applications using the DevOps Change Workspace, Classic UI, or the Service Catalog.
  - DevOps Change Workspace: Navigate to **Workspaces > DevOps Change Workspace > Tools and applications > Create applications**.
  - Classic UI: Navigate to **All > DevOps > Apps & Pipelines > Create App**.
  - Service Catalog: Navigate to **All > Service Catalog > Catalog Definitions > Maintatin Items > Create DevOps Application > Try It**.
2. On the Create an application form, choose whether to create a new application and new application model or to base the application on an existing application model.

Choice	Actions
New application	Enter a name for the new application.

Choice	Actions
	<ul style="list-style-type: none"> <li>◦ The system generates a new DevOps application, a new application model, and a new SDLC component (SDLC-C) in the CMDB.</li> <li>◦ The SDLC-C is assigned the same name as the new application.</li> <li>◦ If the DevOps Config application is installed, then DevOps associates the application to the CDM application.</li> </ul>
Existing application model	<p>Select the application model from the list.</p> <p>The system generates a new DevOps application from that application model.</p> <ul style="list-style-type: none"> <li>◦ The application is given the same name as the application model.</li> <li>◦ If an SDLC component exists for the application model, then DevOps associates the SDLC-C to the new application.</li> <li>◦ If an SDLC component doesn't exist for the application model, then DevOps generates a new SDLC-C with the same name as the application model.</li> <li>◦ If DevOps Config is installed, then DevOps creates a CDM application and associates the SDLC-C to the application.</li> </ul>

### 3. Select **Create** or **Submit**.

#### What to do next

You can view the list of all the applications from the Applications () menu in DevOps Change Workspace or the Apps page in the Classic UI.

For each application, you can:

- Connect to a tool.
- Update details such as **#Business app, state (Active)**, and **#Log Level**. Applications marked as **Active** are polled daily for updates to associated plans, repositories, and pipelines.
- Control access to the application by adding user groups to the **Maintained by** field. You can select any group that has at least one user with DevOps roles.

When user groups are added:

- Users with DevOps App Owner role or DevOps Administrator role can edit and associate objects with the application.
- Users having other DevOps roles can view the application.
- Users having DevOps roles, but aren't part of the groups added can't view the application.

**Note:** DevOps admins can always see and edit everything in DevOps.

- View and associate plans, repositories, pipelines, and artifact repositories. Associate the appropriate tool objects to the application to ensure that all the DevOps data is grouped and tracked.
- Delete the application record.

## Associate tool objects to applications - Workspace

After creating an application, you can associate plans, repositories, and pipelines with it. Applications group plans, repositories, and pipelines from DevOps tools, which provides traceability to user stories, commits, test results, and so on.

### Before you begin

Role required: sn\_devops.admin or sn\_devops.app\_owner

### Procedure

1. Navigate to **Workspaces > DevOps Change Workspace > Applications**.
2. Select the application.
3. From the application record page, select the tab for the object type that you want to associate.

Object type	Steps
Plans	<ol style="list-style-type: none"> <li>Select the <b>Plans</b> tab.</li> <li>Select <b>Associate plans</b>.</li> <li>From the list of plans available in DevOps, select the plans to track and associate with the application.</li> <li>Select <b>Associate plans</b>.</li> <li>If you want to import historical data for the plans, select the start and end dates and select <b>Import data</b>. You can import up to 90 days of data.</li> <li>To import data after associating the plans, select the required plans from the <b>Plans</b> tab and select <b>Import data</b>. Select the start and end dates and select <b>Import data</b>.</li> </ol>
Repositories	<ol style="list-style-type: none"> <li>Select the <b>Repositories</b> tab.</li> <li>Select <b>Associate repositories</b>.</li> <li>From the list of repositories available in DevOps, select the repositories to track and associate with the application.</li> <li>Select <b>Associate repositories</b>.</li> <li>If you want to import historical data for the repositories, select the start and end dates and select <b>Import data</b>. You can import up to 90 days of data.</li> </ol>

Object type	Steps
Pipelines	<p>f. To import data after associating the repositories, select the required repositories from the <b>Repositories</b> tab and select <b>Import data</b>. Select the start and end dates and select <b>Import data</b>.</p> <p>a. Select the <b>Pipelines</b> tab.</p> <p>b. Select <b>Associate pipelines</b>.</p> <p>c. From the list of pipelines available in DevOps, select the pipelines to track and associate with the application.</p> <p>d. Select <b>Associate pipelines</b>.</p> <p>e. If you want to import historical data for the pipelines, select the start and end dates and select <b>Import data</b>.</p> <p>You can import up to 90 days of data.</p> <p>f. To import data after associating the pipelines, select the required pipelines from the <b>Pipelines</b> tab and select <b>Import data</b>. Select the start and end dates and select <b>Import data</b>.</p> <p><b>i Note:</b> While associating a pipeline with an app, the pipeline steps are also fetched during import.</p>

#### 4. Select **Save**.

### Associate tool objects to applications - Classic

After creating an application, you can associate plans, repositories, pipelines, and artifacts with it.

#### Before you begin

Role required: sn\_devops.admin or sn\_devops.app\_owner

#### Procedure

1. Navigate to **All > DevOps > Apps & Pipelines > Apps**.
2. Select the application.
3. From the application record page, select the related list for the object type that you want to associate.

Object type	Steps
Plans	<p>a. Select the <b>Plans</b> related list.</p> <p>b. Select <b>Edit</b>.</p>

Object type	Steps
Repositories	<p>c. From the list of plans available in DevOps, select the plans to track and associate with the application.</p> <p>d. Select <b>Save</b>.</p>
Pipelines	<p>a. Select the <b>Repositories</b> related list.</p> <p>b. Select <b>Edit</b>.</p> <p>c. From the list of repositories available in DevOps, select the repositories to track and associate with the application.</p> <p>d. Select <b>Save</b>.</p> <p><b>i Note:</b> While associating a pipeline with an app, the pipeline steps are also fetched during import.</p>

## Accelerating your DevOps change process

Enable the Change Acceleration feature of DevOps for automatic change request creation in your pipeline, and use change approval policies to automate approval under certain conditions.

**i Note:** ServiceNow [Change Management](#) must be installed for change acceleration.

Enable and set up change control when you model your pipeline in DevOps:

- [Model an Azure pipeline in DevOps](#)
- [Model a GitLab basic CI pipeline in DevOps](#)
- [Model a Jenkins pipeline in DevOps](#)

You can view details for active change requests by navigating to **DevOps > Orchestrate > Pipeline Change Requests**.

### Change control process

When change control is enabled for a job in your DevOps development pipeline, a change request is automatically created and set to Assess state to request approval for the execution of the current stage or job.

Pipeline steps can be configured to create change receipts, which do not pause the pipeline. Change receipts include all pipeline data, but do not require approval to proceed in the pipeline.

Change requests can be approved automatically by configuring conditions in a change approval policy. The DevOps change approval policy (DevOps Change Policy) and workflow (Change Request - DevOps) are provided with the DevOps application.

The difference in the DevOps workflow (Change Request - DevOps) from the Normal workflow (Change Request - Normal) is that the change approval policy in the DevOps workflow is the DevOps Change Policy instead of the Normal Change Policy and, therefore, does not use Technical approvals nor Risk approvals activities that are part of the Normal workflow.

Once approved, either automatically or manually, change requests move to Implement state and the job is run. Once the job is run, the change request is moved to Post-Implement state with Close code as successful on a successful job run or Unsuccessful on error in the job run. For more information, see [Custom change request process](#).

If a change request is not approved and moved into canceled state or closed state, the associated Jenkins job is marked as failed and a console message is shown:

```
[ServiceNow DevOps] Job was not approved for execution
```

## Default change policy with DevOps Change Velocity

The DevOps Default Change Request workflow is available by default when you install DevOps Change Velocity (DCV). The three normal change flows available to you are:

- DevOps Demo Change Policy workflow
- DevOps Model Change Request workflow
- DevOps Default Change Request workflow

The DevOps Default Change Request workflow has 12 default inputs (conditions) that you can use in the workflow decision table. Some examples: Number of commits, last commit date, and integration tests failed. You can add, edit, or remove conditions and modify the flow according to your company requirements.

Based on the values of the conditions, the workflow analyses the change request and approves or rejects the change request. For example, if the value for **Code Coverage** is set to 70 (default), the workflow analyses the unit test coverage for the code if the coverage is 70% or more, the change request is automatically approved. If the code coverage is less than 70%, the workflow does not check if the other conditions are met and sets the change request to manual approval.

A default input (condition) **Risk** is available. Set the value of this condition based on the risk assessment to a value between 1 to 4, where 1 means high risk and 4 meaning low risk. Based on the value of this field, the change request would be either auto approved or would require manual approval.

The workflow helps you automate change requests, with minimal manual intervention by users and groups. The three outcomes for the DevOps Default Change Request workflow (depending on the conditions that you specify) are:

- Auto approval: If the conditions specified in the workflow are met, the change request is automatically approved.
- Auto reject: If one or more of the conditions specified in the workflow are not met, the change request is automatically rejected.
- Manual approval: If one or more conditions need manual approval by a user or group, that is specified in the workflow. Notifications are sent by the workflow to the relevant users or groups to expedite the manual approval and progress the change request.

- Note:** By default, the DevOps Default Change Request workflow is deactivated. If you must activate it, open the Workflow page, and click **Activate**. Only one of the three workflows can be active at a given time.

## Automatic approval of change requests using Change Policy workflows

You can enable automatic approval of DevOps changes using the [Change Approval Policy workflow](#) and the DevOps Change Policy workflow.

### Change Approval Policy workflow

A change request is automatically approved for low risk changes, when calculated risk and impact are below threshold values (set on the Step form for the pipeline). The state is moved to Implement.

When calculated risk and impact values are at or above threshold values, the normal change stays in the Assess state until approved.

- Note:** If the **Standard change template** field on the Step form is set, and the calculated risk and impact are below threshold values, a standard change request is created. Otherwise the change is always normal.

### DevOps Change Policy workflow

A change request is automatically approved when certain conditions in the DevOps auto approval cycle are met. The state is moved to Implement. DevOps Change Policy approval is useful for less critical changes that do not require manual approval.

Change Approval Policy is mapped to DevOps Change Policy (instead of Normal Workflow Policy), and Risk Approvals is mapped to DevOps Change Policy.

This policy is inactive by default.

### DevOps Demo Change Policy workflow

When demo data is installed, Flow Designer #DevOps# Demo# Change Automation flow# is available where# normal type change requests can be auto-approved based on the decision policies.

As a part of demo data, the decision policies available are:

- Low risk auto approval policy, where the failed test value is zero##.
- High risk manual approval policy, where the failed test value is greater than zero#.

See [DevOps Demo Change Automation flow](#) for more details.

## Custom change request templates

When you enable change control in the ServiceNow DevOps step , you can select a custom template to populate fields automatically while creating the change request. The change request **Category** field is automatically set to DevOps.

- Note:** Do not configure the **Category** and **changeType** fields from the custom template.

The type of change request corresponds to the change request table in global scope.

## DevOps Model Change Request flow

Customize the [DevOps Model Change Request flow](#) using a flow or a script.

Once the change request is approved, rejected, or canceled, the state of the step execution can be updated by calling Flow Designer action **Update the state of step execution table** even before implementing the change request. If a change request is not approved, and moved into# canceled or rejected state, the associated job is marked as # canceled or failed and a console message is shown.

Flow Designer action **Update the state of step execution table** serves as a trigger for **Change Control Callback** flow, which is used to notify the change decision to the orchestration tool. Hence, calling the Flow Designer action **Update the state of step execution table** is required.

### Automatic change request related lists

For a change request automatically created by DevOps, the **Category** field is automatically set to DevOps, and these related lists are added:

Commits

Commits associated with the change request.

Work Items

Work items associated with the change request.

Artifact Versions

List of artifact versions associated to the package linked to pipeline execution for packages created before the change request is approved.

If no package is linked to the pipeline execution, then the list is empty.

Test Summaries (replaces Test Results related list)

List of test summaries for a pipeline execution associated with an artifact, package, or task execution before the change request.

See [Test Results](#) for more details.

**Note:** Implementation details from the orchestration tool are automatically added to the **Work notes** field on the change request form. Detail added to the work notes is limited to 5 KB of the task execution log for the step.

### Custom change request process

These [DevOps change properties](#) are available to customize your change request flow.

- DevOps change request implement state
- DevOps change request post implement state
- DevOps change request cancel state
- DevOps change request approval text

To customize your change request flow, you must first create a **System Definition > Choice List**. For example, `DevOps_Implement` (value - 10).

Then, add the choice list to **System Definition > Script Include > ChangeRequestStateHandlerSNC**.

Once you have created the choice list and added it to the script include, you can update **DevOps change properties** with the new choice list values. For example, *DevOps change request implement state-10*.

## DevOps Risk Condition

You can use the DevOps risk and impact calculation based on committer risk score.

This condition is inactive by default.

## Test Results related list

Lists the tests that were run in a pipeline after a package was created. If no package was created, then the list includes the tests that were run after an artifact version was created.

Scenarios:

A package is created in the pipeline, but no artifact versions are registered.

- If the change request is created in the package creation stage:

No test results are displayed because a package is not yet linked to the pipeline execution.

- If the change request is created in a stage after the package creation stage:

Build test summaries include those associated with stages after the package creation stage, up to the change-controlled stage.

Artifact versions are registered, but no package is created.

- If the change request is created in the artifact version stage:

No test results will be displayed, because no tests are associated until the task execution is completed.

- If the change request is created in a stage after the artifact version stage:

Build test summaries include those in the artifact version stage, as well as the stages afterward, up to the change-controlled stage.

Both artifact versions and package are created in the pipeline.

- If the change request is part of the stage after artifact version and package creation stages:

Build test summaries include those associated with the package creation stage, as well as the stages afterward, up to the change-controlled stage.

- If the change request is part of the package creation stage, and artifact versions are created as part of an earlier stage;

- or the change request is created in a stage (not package creation) after the artifact version stage, but before package creation stage;
- or the change request is part of the package creation stage and artifact versions are created as part of an earlier stage:

Build test summaries include those associated with the artifact version stage, as well as stages afterward, up to the change-controlled stage.

## Pipeline executions view

You can view pipeline activity by navigating to **DevOps > Orchestrate > Pipeline Executions**.

## Create a DevOps Change request and associate existing DevOps data — Workspace

Create a DevOps change request and associate existing DevOps data through the Workspace UI.

### Before you begin

Role required: sn\_devops.admin

### About this task

For more information on using Change features, see [Create a Change](#).

**Important:** You can update the associated data after you create a request, but you cannot edit a change request if it is in one of the following states:

- Implement
- Post-Implement
- Cancelled

### Procedure

1. Navigate to **Workspaces > DevOps Change > Applications**.
2. Select an application from **Applications** list.
3. In the Recommended actions panel, select **Create manual change**.
4. In the **Category** list, select DevOps and then select **Save**.
5. Select **Add DevOps data** to add supporting data to the change request.
6. Complete the Add DevOps data pages.

- a. Specify the following values on the **Select data type and associations** page:

#### Select data type and associations

Field	Description
Data type	The type of data to associate with the change request.

Field	Description
	<ul style="list-style-type: none"> <li>▪ Artifact version</li> <li>▪ Release version</li> <li>▪ Build number</li> </ul>
Data associations	<p>The specific data to associate with the change request. You can select multiple artifact versions and build numbers.</p> <p>If you select <i>Build number</i> as the <b>Data type</b>, then you must also specify the application name and corresponding pipelines and build numbers.</p>

- b.** Select **Next** to open the Verify data associations page.
- c.** Navigate the tabs to verify that the associated data is mapped accurately. Update settings as needed.
  - Work items
  - Commits
  - Test summaries
  - Artifact versions
  - Software quality summaries
- d.** Select **Submit**.

The Change request form is updated.

- 7.** Configure other settings as needed. For example, specify the schedule.

**i Note:** For more information on configuring settings, see [Create a Change](#).

- 8.** Select **Save**.

### What to do next

To modify the data that is associated to the change request, select **Edit DevOps data**. Verify the associated data that you added or removed in the **Additions/Removal** tabs in the **Add DevOps Data** modal.

## Create a DevOps Change request and associate existing DevOps data — Classic

Create a DevOps change request and associate existing DevOps data through the Classic UI.

### Before you begin

Role required: sn\_devops.admin or sn\_devops.app\_owner

## About this task

Create a DevOps change request and associate existing DevOps data. You can modify the associated details after you have created a DevOps change and associated data with it.

**Important:** If the Change request is in one of the following states, the **Edit DevOps Data** button is inactive:

- Implement
- Post-Implement
- Cancelled

## Procedure

1. Navigate to **All > Change > Create New > Create Normal Change**.

**Note:** For more information on using the Change module, see [Create a Change](#).

2. From the **Category** list, select DevOps or Other.

3. Optional: Select the **DevOps Change** option, if you have selected Other in the **Category** field.

4. Click the **Add DevOps Data** button on the form header.

**Note:** The **Add DevOps Data** button appears only if you have selected DevOps as the category, or Other as the category and selected the **DevOps Change** option.

The **Add DevOps Data** modal displays.

5. In the **Select data type and associations** section, select the data type and the related data associations.

a. Select any of the following options from the **Data Type** list.

- Artifact Version
- Release Version
- Build Number

**Note:** You can select multiple Artifact versions and Build Numbers.

b. Select the data you want to associate with the change from the **Artifact Version**, **Release Version** or **Build Number** field.

**Important:** If you select **Build Number** as the **Data Type**, you must also specify the application name, and corresponding pipelines and build numbers.

6. Click **Next**.

7. Verify that the associated data for the following fields is mapped accurately.

- Work Items
- Commits
- Test Summaries
- Artifact Versions
- Software Quality Summaries

**8.** Click **Submit**.

The Change Request form displays.

**9.** Select the **Assignment group** to which this change request is assigned.

**10.** Configure the change window for the change request from the Schedule tab.

**1 Note:** For more information on configuring the schedule, see [Create a Change](#).

**11.** Click **Submit**.

### What to do next

To modify the associated DevOps data to the change request, click the **Edit DevOps Data** button. Verify the associated data that you added or removed in the **Additions/Removal** tabs in the **Add DevOps Data** modal.

## DevOps change request without mandating category as DevOps

Enable categorizing DevOps change request from DevOps properties. Change requests created as part of pipeline execution steps can now be created with a category other than DevOps.

### Before you begin

Role required: sn\_devops.admin

### About this task

Change requests created in the DevOps application are not created with **Category** as DevOps.

Use related templates for specific categories such as *Hardware*, *Software*, *Database* and so on, to create DevOps change requests. You can now categorize DevOps change requests from the DevOps properties.

New change requests that are created in the DevOps application after you enable the property can continue to use custom change policies and approval workflows, in addition to the base system's DevOps Demo Change Automation flow, and the DevOps Model Change Request flow. If you do not enable this property, change requests continue to be created and processed with the category attribute set to DevOps.

### Procedure

1. Navigate to **All > DevOps > Administration > Properties > Categorize DevOps changes requests on "DevOps change" field**.  
By default, the property check box is cleared.
2. Select the check box to enable categorizing any change request created from the DevOps application, as a DevOps change request.

**Important:** DevOps Insights capabilities are currently disabled when this property is selected.

### Result

Any new change request that is created from the DevOps application is considered as a DevOps change irrespective of the category attribute and processed accordingly. DevOps change request creation and processing no longer depend only on the category attribute being set to DevOps.

## DevOps change acceleration for releases

DevOps change acceleration for releases lets you view all commits and work items in the change request for the DevOps release when approving, rather than having commits spread across multiple task executions.

When you associate commits with an artifact version (CI pipeline), and define an artifact package (CD pipeline), all artifact versions generated since the last time the app was deployed to production are included in the list of commits for the change. Consolidation of these items is helpful, especially when there are multiple CI builds before the deployment.

DevOps change request attributes:

- **Category** DevOps or the **Categorize DevOps change requests on "DevOps Change" field** check box is selected in DevOps properties. For more information, see [DevOps change request without mandating category as DevOps](#)
- Commits and Work Items related lists

## Artifact setup

### 1. (Optional) Create an artifact tool record in DevOps.

**Note:** An artifact tool is not necessary unless a webhook or user-created integration subflow configuration is required to look up artifact versions.

### 2. Register artifacts in the CI pipeline.

### 3. Create a package in the CD pipeline.

**Note:** The package creation step must be before the Prod Deploy step.

## Artifact registration

Configure artifact registration in a scripted pipeline or freestyle job using the [DevOps API / artifact/registration endpoint](#). Multiple artifact versions are supported.

For Jenkins pipeline:

- Scripted and declarative pipeline (`snDevOpsArtifact` Jenkinsfile command)

For example:

```
snDevOpsArtifact(artifactsPayload: """{"artifacts": [{"name": "sa-web.jar", "version": "1.9", "semanticVersion": "1.9.0", "repositoryName": "services-1031"}], "branchName": "master"}")
```

- Freestyle job (**Register Artifact** build step)

For example:

```
{"artifacts": [{"name": "sentiment-analysis-web2", "version": "1.9", "semanticVersion": "1.9.0", "repositoryName": "maven-releases"}]}
```

## Artifact package creation

Configure artifact package creation in a scripted pipeline or freestyle job using the [DevOps API](#) /package/registration endpoint.

**i Note:** Package name must be specified.

For Jenkins pipeline:

- Freestyle job (**Create Package** build step)

For example:

```
{"artifacts": [ {"name": "sentiment-analysis-web2", "version": "1.9", "repositoryName": "maven-releases"} ]}
```

- Declarative and scripted pipeline (`snDevOpsPackage` Jenkinsfile command)

Package with more than one artifact (from different repos) in the payload, for example:

```
snDevOpsPackage(name: "sentimentpackage", artifactsPayload:  
    """{"artifacts": [{"name": "sa-web.jar", "version": "1.9",  
        "repositoryName": "services-1031"}, {"name": "sa-db.jar", "version":  
        "1.3.2", "repositoryName": "services-1032"}], "branchName":  
        "master"}""")
```

**i Note:** When an artifact version is not available during the build, build details (pipelineName or projectName, taskExecutionNumber, stageName, branchName) are used to look up the artifact version in the task execution.

Jenkins plugin step `includeBuildInfo` can be used to include build details in the API call.

## Artifact workflow and objects

The orchestration tool job publishes the new artifact (consisting of versions) to the artifact repository. Each artifact version is associated with a task execution (consisting of the related code commits). A package is created for the release (consisting of specific artifact versions flagged for deployment) and, once the deployment stage completes, the package is marked as deployed to production.

These objects are part of the artifact structure.

### Artifact tool

Used to support artifact repository managers such as JFrog Artifactory.

**i Note:** An artifact tool is not necessary unless a webhook or user-created integration subflow configuration is required to look up artifact versions.

### Artifact repository

Target for artifacts generated in a build, and also a source of artifacts required by a build.

Can be created manually, or through the RegisterArtifact API as new artifacts are published under new repositories in a tool.

## Artifact

Artifact name for which different builds (artifact versions) are generated.

Can be created manually, or through the RegisterArtifact API. Artifacts (versions) are associated with a task execution and published to an artifact repository.

## Artifact version

Specific version of the artifact. Deployable component of an application generated by a CI build. When provided, semantic version is used.

Can be created through discovery, or through the RegisterArtifact API. Artifacts (versions) are associated with a task execution and published to a tracked artifact repository.

## Semantic version

Optional attribute of an artifact version that, when provided, is used to determine commits for a change.

Semantic version format is (MAJOR.MINOR.PATCH).

## Package

Collection of artifact versions used as input to a CD pipeline, or for associating test results.

Package creation is triggered by the CreatePackage API call from the orchestration tool and contains the name, version, and repository name of all the artifact versions included in the package. A check box indicates whether the package has been deployed to production.

## DevOps change acceleration for releases

### **DevOps change request – category DevOps**

### **DevOps change request – commits and work items**

### **DevOps artifact version list**

### **DevOps artifact version – commits**

### **DevOps artifact version – packages**

### **DevOps package**

## **How commits are determined for a release in DevOps**

The DevOps artifact package and its associated artifact versions are used to determine which commits are included in a DevOps change.

All commits for artifact versions in the package that were generated after the last deployment date (up to the version currently being deployed) are included in the change request. Previous major versions are not included.

**Note:** Patch and hot fix versions are excluded.

When provided, semantic version is used to determine commits for a change. Format is (MAJOR.MINOR.PATCH). For example, semantic version 2.0.1 is read as follows:

- Major version 2
- Minor version 0
- Patch/hotfix version 1

Failed task executions between the previous and current artifact versions that did not build an artifact but have associated commits are also associated to the created artifact version.

### Example: Commits and packages

This example consists of:

- Three build pipelines (A, B, and C)
- A release pipeline (ABC) under change control, with four packages:
  1. Build pipeline A (major version 1)
  2. Build pipelines A and B (major version 1)
  3. Build pipelines B and C (major version 1)
  4. Build pipelines A, B, and C (major version 1)

#### Package 1 (A 1.1.0)

Commit	Build pipeline	Semantic version	Included in package
1	A	1.0.0	X
2	A	1.0.1	--
3	A	1.1.0	X

**Note:** Commit 2 (build A, semantic version 1.0.1) is not included in the package because the semantic version syntax indicates a patch or hot fix.

#### Package 2 (A 1.2.0, B 1.1.0)

Commit	Build pipeline	Semantic version	Included in package
4	A	1.1.1	--
5	A	1.2.0	X
6	A	1.2.0	X
7	B	1.0.0	X
8	B	1.0.0	X
9	B	1.1.0	X
10	B	1.1.0	X

**Package 2 (A 1.2.0, B 1.1.0) (continued)**

Commit	Build pipeline	Semantic version	Included in package
<b>i Note:</b> Commit 4 (build A, semantic version 1.1.1) is not included because the semantic version syntax indicates a patch or hot fix.			

**Package 3 (B 1.2.0, C 1.0.0)**

Commit	Build pipeline	Semantic version	Included in package
11	A	1.3.0	--
12	B	1.2.0	X
13	B	1.2.0	X
14	C	1.0.0	X
15	C	1.0.0	X
16	C	1.0.0	X

**i Note:** Commit 11 (build A, semantic version 1.3.0) is not included because the package does not specify build A.

**Package 4 (A 1.4.0, B 1.3.0, C 1.2.0)**

Commit	Build pipeline	Semantic version	Included in package
17	A	1.4.0	X
18	A	1.4.0	X
19	B	1.3.0	X
20	B	1.3.0	X
21	C	1.1.0	X
22	C	1.1.0	X
23	C	1.2.0	X
24	C	1.2.0	X

**i Note:** Commit 11 is also included in this package because it is part of the changes in major version 1 of build A since the last package (including major version 1 of build A), package #2, was deployed to production.

**Setting up a DevOps artifact tool record**

Set up a DevOps artifact tool connection to discover artifact repositories when webhook or user-created integration subflow configuration is required to look up artifact versions.

## Configure connection and credential alias - artifact tool

Before you set up your tool records in DevOps, your admin configures the DevOps CreateDevOpsTool connection and credential alias to allow access to the tools in your environment.

### Before you begin

Role required: sn\_devops.admin

### About this task

Your admin must configure an HTTP connection in the CreateDevOpsTool connection and credential alias provided, and add admin credentials to connect automatically when you set up your DevOps planning, coding, and orchestration tools:

- URL: `https://<instance name>.service-now.com/api/now/table`
- Credentials: admin

**Note:** The admin role is not necessary. A user with connection\_admin role can configure an HTTP connection.

You only need to configure the CreateDevOpsTool connection and credential alias once to set up all of your tools in DevOps.

The CreateDevOpsTool connection and credential alias is used with the DevOps Create a Tool subflow in [Flow Designer](#). You can view executions in Flow Designer for information regarding flow results.

### Procedure

1. Navigate to **All > Connections & Credentials > Connection & Credential Aliases** and open the **CreateDevOpsTool** record.
2. In the Connections related list, create a record and enter a **Name** for the connection.
3. On the Connection form, click the **Credential** field lookup list, and then click **New** to create an admin credential.
  - a. Click **Basic Auth Credentials** and enter a **Name**.
  - b. Enter admin username and password (required to access the tools in your DevOps environment).

**Note:** The admin role is not necessary. A user with connection\_admin role can configure an HTTP connection.
4. On the Connection form, enter `https://<instance name>.service-now.com/api/now/table` for the **Connection URL**.

### Create an artifact tool record in DevOps

Create an artifact tool record in DevOps to discover artifact repositories when webhook or user-created integration subflow configuration is required to look up artifact versions.

### Before you begin

Role required: sn\_devops.admin

## About this task

**Note:** An artifact tool is not necessary unless a webhook or user-created integration subflow configuration is required to look up artifact versions.

## Procedure

1. Navigate to **All > DevOps > Tools > Create New** and create a record.
2. Enter a **Name**, **Tool Integration**, and **Tool URL**.

**Note:** If you do not have global admin privileges for your tool (to allow automatic configuration of the webhook URL), you might need to have the tool admin user configure it for you (cut and paste the webhook URL into the tool configuration). Once the webhook is configured in the tool, **Enter Manual Configuration Mode** to connect to the tool manually, then exit.

After submitting, the artifact tool is automatically **Connected Successfully** using a connection alias, and HTTP tool connection (basic authentication credential).

## Example:

### DevOps artifact tool setup

## Configuring DevOps change request details within the pipeline

Set closure code and change request fields from within Azure and Jenkins pipelines.

**Note:** Configuring change request details from within a GitLab pipeline is not supported.

Closure information and change request attributes are contained with the `changeRequestDetails` object.

### Closure code

Set the `setCloseCode`: parameter to true/false based on the desired behavior. Default is false.

If set to true, Close Code, Closure Notes are updated as specified in the change step attributes and the change request is moved to post-implement state.

If set to false, once the job or pipeline has completed the change request is not updated and remains in the Implement state:

- Closure Information in the change request is not set (**Close code** and **Close notes** fields are left empty).
- A link to the step execution is added to the **Work notes**.

### Change request fields

Set change request field values within the pipeline for the change request template specified.

- Use the `attributes`: parameter to set field values.
- Use the [DevOps - POST /devops/orchestration/changeControl](#) endpoint of the DevOps API.

**i Note:**

- If a specified field has a dependent field that is required, you must set that attribute as well.
- If the attribute for the dependent required field is not set, change request and related step execution are canceled, and work notes are updated.

Field values within the `attributes:` parameter are key-value pairs. Meaning, the key is the field name within the template and the value is the information to populate in the field.

You can use the `changeControl` API to specify fields such as `type`, `cmdb_ci`, `template`, `assignment_group`, `business_service`, `standard_change_template` and create a change request.

**i Note:** Please ensure that you keep in mind the following points while configuring change requests from within the pipeline:

- Any attributes that you set in the `changeControl` API overrides any values set on the step.
- Change type and template fields are always taken from one source, that is you cannot use a combination attributes from the API request and the change request form.

All fields in the Change Request [change\_request] table are supported except where specified.

**Change request fields supported**

Unsupported fields	<ul style="list-style-type: none"> <li>• risk</li> <li>• impact</li> <li>• number</li> <li>• sys_id</li> <li>• risk_impact_analysis</li> </ul>
Supported fields	All remaining fields in the Change Request [change_request] table.

**i Note:** The attribute name must match the change request field name, and the value specified must be valid.**Sample JSON payload**

```
{
  "callbackURL": "http://192.168.0.4:3000/jenkins/sn-devops/pipeline_839b7605-b98d-4831-bc87-96829de7da37",
  "orchestrationTaskURL": "http://192.168.0.4:3000/jenkins/job/java_sample_tests#deploy/",
  "isMultiBranch": "false",
  "orchestrationTaskName": "java_sample_tests#deploy",
  "orchestrationTaskDetails": {
    "branch": "main"
  }
}
```

```

    "triggerType": "upstream",
    "upstreamTaskExecutionURL": "http://192.168.0.4:3000/jenkins/job/java_sample_tests/129/execution/node/35/wfapi/describe",
    "taskExecutionURL": "http://192.168.0.4:3000/jenkins/job/java_sample_tests/129/execution/node/50/wfapi/describe"
},
"changeRequestDetails": {
    "setCloseCode": false,
    "attributes": {
        "sys_created_by": "1832fbe1d701120035ae23c7ce610369",
        "sys_updated_by": "56826bf03710200044e0bfc8bcbe5dca",
        "requested_by": {
            "name": "Abel Tuter"
        },
        "watch_list": [
            {
                "name": "Abel Tuter"
            },
            {
                "name": "Aileen Mottern"
            },
            {
                "name": "Alejandra Prenatt"
            },
            "56826bf03710200044e0bfc8bcbe5dca"
        ],
        "work_notes_list": [
            "56826bf03710200044e0bfc8bcbe5dca",
            "46c6f9efa9fe198101ddf5eed9adf6e7",
            "d8f57f140b20220050192f15d6673a98"
        ],
        "assigned_to": "1832fbe1d701120035ae23c7ce610369",
        "category": "Service",
        "sys_created_on": "2021-02-09 18:58:41",
        "priority": "2",
    }
}
}

```

## Pipeline examples

### Change request details - Azure pipeline

### Job-level settings — Jenkins

### Change request details - Jenkins

## Using DevOps Model Change Request flow

Customize or recreate the DevOps Model Change Request flow based on your requirements using a flow or a script.

In the DevOps Model Change Request flow, the state of step execution is changed based on the change approval. However, you can customize or recreate this flow based on your requirements.

After the change request state is moved to approve, canceled, or rejected (either manually or by using a change policy), call the **Update state of step execution based on change approval** Flow Designer action to update the **State** field of step execution record.

You can use either a flow or a script to call the action.

### Calling the Flow Designer action using a flow

Calling the **Update state of step execution based on change approval** Flow Designer action is required to update the state of the step execution record according to the approval field in the change request record.

This action serves as a trigger for the **Change Control Callback** flow, which is used to notify the change decision to the orchestration tool.

### Calling the Flow Designer action using a script

Method to call the Flow Designer action from a script:

```
sn_fd.FlowAPI.executeAction('sn_devops.name_of_FD_action', inputs);
```

### Default Change Handler subflow

Use the Default Change Handler subflow to populate these change request fields with default values.

- Requested by
- Justification
- Implementation Plan
- Backout plan
- Test Plan
- Short Description
- Description
- Start Date
- End Date
- Risk Impact Analysis

The Default Change Handler subflow overrides the field values that were populated using a template while creating the change request record.

If desired, you can write a custom subflow in place of this flow by modifying the [`/sn\_devops.change\_request\_handler\_subflow`](#) DevOps property.

## DevOps# Demo# Change Automation flow

When demo data is installed, Flow Designer **DevOps# Demo# Change Automation** flow is available where normal type change requests can be auto-approved based on the decision policies.

As a part of demo data, the decision policies available are:

- Low risk auto-approval policy, where the failed test count is zero#.
- High risk manual approval policy, where the failed test count is greater than zero#.

As a part of the demo flow, Flow Designer actions available are:

- DevOps Set Change Window

Action used to set the change request start date.

- Get Change Policy User/Group

Action used to fetch the user/group based on the change request approval policy.

- DevOps Create Auto Approval Record

Action used by the user to create Auto-Approve/Auto-Reject approval records.

## Manage pull request pipelines from DevOps

Manage Jenkins pipeline's pull requests for GitHub and Bitbucket coding sources from ServiceNow DevOps. You can enable the change approval process on your pull request to control pull request merge approvals from ServiceNow DevOps and monitor pull request details associated with the change request.

### Before you begin

Ensure that the following prerequisites are met before approving pull requests:

- Your admin has created a pull request branch in GitHub or Bitbucket using a Multi-branch project pipeline. For more information, see [GitHub branch source](#) and [Bitbucket branch source](#).
- Your admin has configured Jenkins integration with DevOps. For more information, see [Jenkins integration with DevOps](#).
- Your admin has configured GitHub integration with DevOps. For more information, see [GitHub integration with DevOps](#).
- Your admin has configured Bitbucket integration with DevOps. For more information, see [Bitbucket integration with DevOps](#).
- Your admin has enabled the Change Acceleration feature of DevOps for automatic change request creation in your pipeline. For more information, see [Accelerating DevOps change](#).

Role required: sn\_devops.admin

## Procedure

1. Create a pull request in GitHub or Bitbucket.  
For more information, see [Creating a pull request](#) or [Create a pull request to merge your change](#).
2. In the Jenkins dashboard, navigate to **Manage Jenkins > Configure System > ServiceNow DevOps Configuration**.
3. Select the **Pull Request Pipeline Tracking Check** option.
4. Run the pull request pipeline in Jenkins for the required project.  
A build is initiated and a change request notification is sent to the approver associated with the pipeline.
5. Navigate to **DevOps > Orchestrate > Pipeline Change Requests**.
6. Select the change record associated with the pull request.
7. Approve the change request associated with the pull request by selecting **Approved** in the **State** field.  
Merge is enabled in GitHub or Bitbucket for the pull request.
8. Select the Pull Requests related list associated with the change record.
9. View the pull request details by clicking the pull request number corresponding to the **Number** field.  
All the pull request details (including the details after the request is merged) associated with the change record are displayed.

## DevOps Insights reports and Pipeline UI

To help you plan and implement updates to your DevOps processes, DevOps Insights displays a variety of configurable reports that are grouped by type of metric. Select the DevOps Insights icon () to view the reports.

### DevOps Insights

Analyze operational and business reports and to determine the overall efficiency# and growth of your# development processes.

- Monitor activities, KPIs, and metrics in a central dashboard.
- Track performance over time.
- View summary data broken out by bug, epic, feature, "other", story, or task.

### Mining and analyzing the metrics

Each report provides both a summary of an important business metric and access to more detailed information about the metric.

- By default, many reports display data for the last 30 days. Select the **Date** icon () to configure a custom date range.
- Point to a specific data point in a chart to view detailed data.
- Select a specific data point to view the key performance indicator (KPI) details for the report in a separate tab. You can then use the powerful data-management tools in the [Performance Analytics](#) application to drill deeply into the data.
- To apply any filter other than the date filter, select the filter icon and then move filter categories from the **Available** list to the **Applied** list.

- Filters appear at the top of the tab. For each report, the Applied filter icon () displays the number of filters that are currently applied to a report. Select the icon to view and configure the filters that are applied to a widget.

**i Note:** Typically, you can configure a maximum of two filters in addition to the date filter.

## Schedule of data collection

You must enable the daily data collection job to populate the dashboards with new DevOps data coming in from the connected third-party tools.

1. Navigate to **All > Performance Analytics > Data Collector > Jobs**.
2. Configure the *[DevOps] Daily Data Collection* job to run as system admin.

Run the *Load Change Request Repo Detail* scheduler script and *[DevOps] Historical Data Collection* job for historical data:

1. Navigate to **All > System Definition > Scheduled Jobs**.
  2. Execute the *Load Change Request Repo Detail* scheduler script.
- This execution can take 2-3 minutes to complete. For more information, see [Populate the Change Request Repo Detail table](#).
3. Navigate to **All > Performance Analytics > Data Collector > Jobs**.
  4. Execute the *[DevOps] Historical Data Collection* job to run as system admin.

This job can take 20-25 minutes to complete execution but this time depends on the volume of data. This is a one-time task to collect historical DevOps data, and not meant to be on a schedule.

### Tip:

Based on the time taken to complete this job, plan on running it during a period of low usage.

## DevOps Insights Summary dashboard

The Summary dashboard enables an overview of key metrics across various categories from accelerate to quality metrics along with a quick glance of the most active applications.

### Summary reports

#### Summary reports

Report	Description	
Average WIP cycle time	Average time in days that a work item is in the WIP state before completion.	Calculation: Time to complete WIP items divided by the number of completed work items

## Summary reports (continued)

Report	Description	
Average lead time	Daily average lead time for prod deploy pipeline executions.	Calculation: Sum of successful prod deploy lead times divided by the number of successful production pipeline executions.
Deployment frequency	Daily count of prod deploy pipeline executions.	Database view joined by step execution, change request, change request repository detail, step, pipeline, app, and application product detail.
Average test pass %	Daily average test pass percentage for pipeline executions.	Database view joined by test summary, test summary relations, step execution, change request repository detail, step, pipeline, app, and application product detail.
Work items completed	Work items, grouped by work item type, that were set to the complete state during the last 30 days.	Database view joined by work item, metric instance, commit work item, commit, commit execution, step execution, step, pipeline, app, and application product detail.
Activity — last 30 days	For each application, the summary of activity during the last 30 days.	Database view joined by commit, commit work item, repository, commit execution, app, test summary relations, step execution, test summary.

## DevOps Insights Flow metrics dashboard

The Flow metrics reports help you visualize how work is moving through your development process. Uncover bottlenecks by determining which kinds of work (such as stories or bugs) are taking the longest.

### Flow metrics reports

## Flow metrics reports

Report	Description	Source
Average flow time	Average time, in days, that a work item spent from the created state to step execution end time of a successful prod pipeline deployment.	Database view joined by work item, metric instance, commit work item, commit, commit execution, step execution, step, pipeline, app, and application product detail.
Average WIP cycle time	Average time in days that a work item is in a specific state.	Database view joined by Work Item, Metric Instance, Commit, Step Execution, Commit execution, and Commit work item lists.
Average WIP count	Average number of work items that are WIP during the last 30 days.	Average count of Work Items with State set to Work In Progress.
Work items completed	Number of work items in the complete state that are associated to commits through a successful production pipeline deploy in the last 30 days.  <b>i Note:</b> Filters don't apply for this chart.	Average count of Work Items associated to Step Execution type set to Prod deploy. This requires commit to work item association as only work items deployed through a pipeline are in production.
Average work item cycle time	Average time in days that a work item is in a specific state.	Database view joined by Work Item, Metric Instance, Commit, Step Execution, Commit execution, and Commit work item lists.
Throughput and distribution	Number of work items, by type, that are associated to commits through a successful production pipeline deploy.	Database view joined by Work Item, Metric Instance, Commit, Step Execution, Commit execution, and Commit work item lists.
Average planned to deploy flow time	Average duration from the time that a work item associated to a commit was created to the time that the item was successfully deployed to production through pipelines.	Database view joined by Work Item, Metric Instance, Commit, Step Execution, Commit execution, and Commit work item lists.
Work in progress	Number of work items that are currently in the work in progress state.	Database view joined by Work Item, Metric Instance, Commit, Step Execution, Commit execution, and Commit work item lists.

## DevOps Insights Change acceleration dashboard

The DevOps Insights Change acceleration tab displays change acceleration metrics that focus on your path to automation, comparing automated changes to manual changes, change policy decisions, and ROI. You can use this information to verify that automated change requests are being resolved more quickly than manually approved change requests.

### Change acceleration reports

Change requests that are part of step executions qualify as DevOps change requests.

#### Change acceleration reports

Report	Description	Source
DevOps change volume	Number of DevOps change requests created.	Database view joined by Step Execution, Change request, and change request repository.
Time to close changes	Average time in hours to close DevOps changes by application.  For each month, this is the average for (Time that change request was closed) minus (Time that change request was opened).	Database view joined by Step Execution, Change request, and change request repository.
Automated vs manual changes	Comparison between the numbers of DevOps change requests using change approval policies and DevOps change requests requiring manual approval.  <ul style="list-style-type: none"> <li>A change request is considered automated when a change policy performs the approve/reject action.</li> <li>A manual change is any change request that is assigned to a user or group for the approve/reject action and isn't acted on by a change policy.</li> </ul>	Database view joined by Step Execution, Change request, and change request repository.
Changes awaiting approval	Number of DevOps changes that are pending approval by date range.  By default, change requests in the New or Assess state	Database view joined by Step Execution, Change request, System properties, and change request repository detail lists.

## Change acceleration reports (continued)

Report	Description	Source
	<p>are considered awaiting approval.</p> <p>To specify the states that are considered awaiting approval, update the <i>Change Request Awaiting States</i> property setting. For details, see <a href="#">DevOps Insights workspace properties</a>.</p>	
Change policy decision: auto-accept	<p>The number of change policy decisions by decision type that automatically accept change requests.</p> <p>Point to a value on the chart to view the list of policy decisions that were used for auto-approval and the number of times that a decision was used for auto-approval.</p>	Database view joined by Step Execution, Change request, System properties, and change request repository detail lists.
Change policy decision: auto-reject	<p>The number of change policy decisions by decision type that automatically reject change requests.</p> <p>Point to a value on the chart to view the list of policy decisions that were used for auto-rejection and the number of times that a decision was used for auto-rejection.</p>	Database view joined by Step Execution, Change request, Policy applied, Auto approval rejection action, Change approval definition, and change request repository detail lists.
Change acceleration savings	<p>Net amount saved per month by automating DevOps changes.</p> <p>When a change is automated, a developer doesn't have to manually fill out the change request and associate each work item, code commits, test results and other evidence and artifacts to the change. After this activity is automated, hours that would have</p>	Calculation: Average hourly developer cost multiplied by developer hours saved.

## Change acceleration reports (continued)

Report	Description	Source
	<p>been spent on filling out the change, searching, tracking down and attaching items from other tools to a change, will now be saved. More work items require a relatively increasing number of hours to associate them manually. Therefore, higher numbers of work items should result in more hours saved after the change is automated. Change acceleration savings are calculated by multiplying the hours saved by the average hourly developer cost.</p> <p>To change the default value of the average hourly developer cost, update the <i>Average Hourly developer Cost</i> property setting. For details, see <a href="#">DevOps Insights workspace properties</a>.</p>	
Developer hours saved	<p>Number of developer hours saved per month by automating DevOps changes.</p> <p>To change the default value of 1 (one) hour per developer, update the <i>X hours per Developer time</i> property setting. For details, see <a href="#">DevOps Insights workspace properties</a>.</p>	<p>Calculation: Number of work items in a change request multiplied by 1 hour per developer.</p>

## DevOps Insights Accelerate metrics dashboard

The Accelerate metrics are four key DevOps metrics that measure software delivery performance. Deployment frequency and lead time measure DevOps speed, while change failure rate and mean time to recovery are used to measure stability.

### Accelerate metrics reports

## Accelerate metrics reports

Report	Description	Source
Average lead time	<p>Average of:</p> <p>([Time the code is successfully pushed to production] minus [Earliest commit time])</p> <p>Applies to pipeline steps of type <b>Prod Deploy</b> that are in the completed state.</p> <p><b>Note:</b> This widget uses average aggregation and doesn't support multi-element selection.</p>	Pipeline execution
Average deployment frequency	<p>Average number of successful production deployments.</p> <p>Applies to pipeline steps of type <b>Prod Deploy</b> that are in the completed state.</p>	Step execution
Average MTTR	<p>Average resolve time for an incident caused by a DevOps change.</p> <p><b>Note:</b> This widget uses average aggregation and doesn't support multi-element selection.</p>	Database view joined by Incident, Change Request, Step Execution, Step, Pipeline, and App lists.
Average DevOps change failure rate	<p>Daily average ratio of DevOps change requests that are associated to an incident divided by all DevOps change requests.</p> <p><b>Note:</b> This widget uses formula and doesn't support multi-element selection.</p>	Change request
Commit to deploy lead time	<p>Duration from the earliest commit time to production deployment for a successful pipeline execution.</p>	Calculation: Sum of successful prod deploy Lead times divided by the number of successful production pipeline executions.

## Accelerate metrics reports (continued)

Report	Description	Source
	<p>The value includes only steps of type <b>Prod deploy</b> that are in the completed state.</p> <p>You might investigate high lead times by identifying the slowest pipeline steps. For example, a manual change approval process could increase lead time.</p>	<p>Needs 'Commits to Task Execution' association. Both Repository and Pipeline must be tracked and associated to the same App.</p> <p>Needs Step type to be Prod Deploy.</p> <p>Needs Task Execution result to be Successful.</p>
Mean time to restore from incidents caused by DevOps changes	Daily average time to resolve an incident that was caused by a DevOps change.	<p>Database view joined by step execution, change request, change request repository detail, and incident lists.</p> <p>Calculated only for DevOps incidents by considering incident opened and closed time.</p>
Production deployment frequency	<p>Number of successful production deployments.</p> <p>The value includes only pipeline steps of type <b>Prod deploy</b> that are in the completed state.</p> <p>To change the default value of 30 days, update the <b>Date range</b> setting.</p>	<p>Database view joined by Change Request, Step Execution, change request repository detail.</p> <p>Needs Step Execution state to be Completed.</p> <p>Needs Step type to be Prod Deploy.</p>
DevOps change failure rate from incidents	<p>Percentage of DevOps changes that caused incidents divided by all DevOps changes deployed to production.</p> <p>If a change request caused multiple incidents, then only the change that caused or triggered the incident is considered. The number of incidents that the change caused isn't considered.</p>	<p>Needs DevOps Change Request (Change associated to a Step Execution).</p> <p>Needs Step Execution state to be Completed.</p> <p>Needs Step type to be Prod Deploy.</p> <p>Needs Incident caused by to be DevOps Change Request.</p>

## DevOps Insights Quality metrics dashboard

The Quality metrics dashboard enables a quick glance at data from tools such as SonarQube for code coverage, test pass percentage from your orchestration tools, vulnerabilities from security tools, and even overall bug counts.

### Quality metrics reports

#### Quality metrics reports

Report	Description	Source
Code coverage %	Percentage of code that is covered by testing.	Needs Software Quality Scan Detail with Category = Coverage (%)  Needs Software Quality Scan Summary Relations related to Task Execution
Test pass %	Test pass percentage.	Needs Test Summary related to Task Execution
Security vulnerabilities	Count of security vulnerabilities over time.	Needs Software Quality Scan Detail with Category = Vulnerabilities  Needs Software Quality Scan Summary Relations related to Task Execution.
Bug counts	Number of work items of type bug.	Needs <i>Commits to Work Items</i> association. Both Repository and Plan must be tracked and associated to the same app.  Needs <i>Commits to Task Execution</i> association. Both Repository and Pipeline must be tracked and associated to the same App  Needs Work Item to be of type Bug.

## DevOps Insights Development dashboard

Development metrics focus on commit data that provides insights into how active and agile your teams are. With this information, you can reach full traceability of work by ensuring that commits are being tagged with the appropriate work items.

## Development reports

### Development reports

Report	Description
Commit frequency	<p>Number of commits associated to pipeline executions.</p> <p>Smaller, more frequent commits are preferred over larger less frequent commits.</p>
Active committers	<p>Number of committers that submitted commits.</p> <p>Because this report uses metric aggregation, it doesn't support multi-element selection.</p>
Top committers (30-day running sum)	<p>Committers with the highest number of commits.</p>
Top reverters (30-day running sum)	<p>Committers with the highest number of reverts.</p>
Average commits per committer	<p>Average calculated as: <math>(\text{total number of commits}) / (\text{active committers})</math>. A higher value is more favorable.</p> <p>Because this report uses metric aggregation, it doesn't support multi-element selection.</p>
Average commits per pipeline execution	<p>Average commits per pipeline, calculated as <math>(\text{Total number of commits}) / (\text{Number of pipeline executions})</math>.</p> <p>A low number is preferable, which indicates a concentrated effort, versus switching from task to task without completion.</p> <p>Because this report uses metric aggregation, it doesn't support multi-element selection.</p>
Commits without work items	<p>Commits made that aren't associated to a work item, grouped by committer.</p> <p>This report is useful for investigating and resolving why a commit isn't tied to a work item, because all commits should be tied to a work item.</p>
Pipeline pass percentage	<p>Ratio of successful pipeline executions divided by total number of pipeline executions.</p>

## DevOps Insights Operational stability dashboard

Operational metrics reflect on the stability of your applications to enable you to ensure that your teams are moving fast without compromising release quality.

### Operational stability reports

#### Operations stability reports

Report	Description	Source
Incidents	Number of incidents for pipeline steps of type Prod deploy that are linked to a service in the CMDB.	Needs Step type to be Prod Deploy.  Needs Incident associated to a Service that matches the one in the prod deploy Step.
Outages	Number of outages for pipeline steps of type Prod deploy that are linked to a service in the CMDB.	Needs Step type to be Prod Deploy.  Needs Outage (cmdb_ci_outage) associated to a Service that matches the one in the prod deploy Step.
Service availability	Average service availability for pipeline steps of type Prod deploy that are linked to a service in the CMDB.	Needs Step type to be Prod Deploy.  Needs parent Service Offering associated to a Service that matches the one in the prod deploy Step.
Error budget remaining	Percentage of error budget left to spend in a month, for pipeline steps of type Prod deploy that are linked to a service in the CMDB. This data is from the Site Reliability operations application.  An error budget is the amount of Service Level Objective (SLO) that you can spend over a specified time. It can be used to manage release velocity. It's typically based on availability, latency, and so on.	Needs Step type to be Prod Deploy.  Needs parent Service Offering associated to a Service that matches the one in the prod deploy Step.  Needs SRO application. See <a href="#">Install the Site Reliability Operations application</a> .

## Operations stability reports (continued)

Report	Description	Source
	<p><b>i Note:</b> You must install the Site Reliability Operations application from the ServiceNow Store to enable the Error budget remaining report. For more information, see <a href="#">Install the Site Reliability Operations application</a>.</p>	

## DevOps Insights Standard dashboard

Use the DevOps Insights application with ServiceNow Performance Analytics to gain insight into your DevOps environment.

**i Note:** To collect daily or historical Insights data, unless these jobs were previously customized, you must select **Active** and set the **Run As** credentials to **System Administrator**:

- **[DevOps] Daily Data Collection (Daily)** (inactive by default)

**i Note:** This is a scheduled job (to be run regularly) to collect daily DevOps data. For optimal performance, set this job to run during periods of low usage.

- **[DevOps] Historical Data Collection (On Demand)** (inactive by default)

For new DevOps Insights installations, if you have already been using the DevOps app and you installed DevOps Insights at a later date, run the Historical Data Collection job to collect historical Insights data.

**i Note:** This is a one-time job to collect historical DevOps data, and not meant to be on a schedule. It might take awhile, so plan on running this job during a period of low usage.

Reports in the dashboard tabs get updated when the dashboard is refreshed.

## Change Acceleration

### Change Acceleration

Report	Source list	Description
Total Changes Submitted - Yearly	Change Request	Total DevOps changes submitted yearly.
Avg Time to Close - Last 30 days	Change Request	Average time to close DevOps changes in the last 30 days.

## Change Acceleration (continued)

Report	Source list	Description
		<p><b>i Note:</b> This widget uses average aggregation and does not support multi-element selection.</p>
Change Approval Rate - Last 30 days	Change Request	<p>DevOps average change success rate for change requests in the last 30 days:</p> $([[\text{DevOps Change Success}]] / [[\text{DevOps Change}]]) * 100$ <p><b>i Note:</b> This widget uses average aggregation and does not support multi-element selection.</p>
Non-DevOps Change Approval Rate - Last 30 days	Change Request	<p>Non- DevOps change approval rate for change requests in the last 30 days:</p> $([[\text{Non-DevOps Change Approval}]] / [[\text{Non-DevOps Change}]]) * 100$ <p><b>i Note:</b> Filter is not applicable to this widget.</p>
Change Request Volume	Change Request	<p>Volume of change requests created for DevOps in the last 7 days.</p> <p>Compare the number of change requests created after your transition to DevOps so you can see the advantage of running DevOps in your environment.</p>
Pending Changes per Pipeline	Step Execution	<p>Number of change requests that have not been closed for each pipeline.</p> <p>See the blockages in each pipeline that are keeping the change request from being completed so you can investigate the cause.</p>

## Change Acceleration (continued)

Report	Source list	Description
Average Time to Close Changes	Change Request	<p>Average time to close DevOps changes by app.</p> <p>Compare DevOps change request statistics with non-DevOps change requests to see that DevOps change requests are getting resolved faster.</p>
Changes Awaiting Approval	Change Request	<p>Number of DevOps changes awaiting approval by date range.</p> <p>Compare DevOps change request statistics with non-DevOps change requests to see that DevOps change requests are getting resolved faster.</p>
Non-DevOps Changes Awaiting Approval	Change Request	<p>Number of Non-DevOps changes awaiting approval by date range.</p> <p>Compare DevOps change request statistics with non-DevOps change requests to see that DevOps change requests are getting resolved faster.</p>

## Accelerate Metrics

The Accelerate Metrics tab shows deployment frequency, lead time, MTTR, and change failure rate info.

### Accelerate Metrics

Report	Source list	Description
Deployment Frequency - Monthly	Step Execution	<p>Number of successful production deployments in a month.</p>

## Accelerate Metrics (continued)

Report	Source list	Description
		Applies to steps of type <b>Prod Deploy</b> that are in completed state.
Average Lead Time	Pipeline Execution	<p>Average of:</p> $([\text{Time the code is successfully pushed to production}] - [\text{Earliest commit time}])$ <p>Applies to steps of type <b>Prod Deploy</b> that are in completed state.</p> <p><b>Note:</b> This widget uses average aggregation and does not support multi-element selection.</p>
Mean Time to Resolve - Last 30 days	Database view joined by Incident, Change Request, Step Execution, Step, Pipeline, and App lists.	<p>Average resolve time for an incident caused by a DevOps change in the last 30 days.</p> <p><b>Note:</b> This widget uses average aggregation and does not support multi-element selection.</p>
Change Failure Rate - Monthly	Change Request	<p>Average change failure rate in a month.</p> <p><b>Note:</b> This widget uses formula and does not support multi-element selection.</p>
Deployment Frequency	Step Execution	<p>Number of successful production deployments in the last 30 days.</p> <p>Applies to steps of type <b>Prod Deploy</b> that are in completed state.</p>
Change Failure Rate	Change Request	Average change failure rate in the last 30 days.

## Accelerate Metrics (continued)

Report	Source list	Description
Mean Time to Resolve Trend	Database view joined by Incident, Change Request, Step Execution, Step, Pipeline, and App lists.	Daily average resolve time for an incident caused by a DevOps change.
Lead Time	Pipeline Execution	([Time the code is successfully pushed to production] - [Earliest commit time])  Applies to steps of type <b>Prod Deploy</b> that are in completed state.  <b>i Note:</b> This widget uses average aggregation and does not support multi-element selection.

## Operational Stability

The Operational Stability tab shows:

- Service availability

Average service availability and daily service availability.

- Mean time to resolve (MTTR)

Mean time to resolve and daily mean time to resolve.

**i Note:** You must install the Service Portfolio Management Foundation (com.snc.service\_portfolio) plugin to see service availability widgets.

Demo data is also provided for the service availability widgets.

**i Note:** You must install the Service Portfolio Management Foundation (com.snc.service\_portfolio) plugin before installing the DevOps Insights application to see demo data.

## Operational Stability

Report	Source list	Description
Mean Time to Resolve - Last 30 days	Database view joined by Incident, Change Request, Step Execution, Step, Pipeline, and App lists.	Average resolve time for an incident caused by a DevOps change in the last 30 days.

## Operational Stability (continued)

Report	Source list	Description
		<p><b>i Note:</b> This widget uses average aggregation and does not support multi-element selection.</p>
Incidents - Monthly	Incident	<p>Number of incidents in a month (based on pipeline steps of type <b>Prod Deploy</b>) linked to business service in CMDB.</p> <p>This report provides an indication of environment stability.</p>
Average Service Availability - Last 30 days	Database view joined by Service Availability, Service Offering, Business Service, Step, Pipeline, and App lists.	<p>Average service availability in the last 30 days (based on pipeline steps of type <b>Prod Deploy</b>) linked to application service in CMDB.</p> <p>This report provides an indication of environment stability.</p>
Outages - Monthly	Outage	<p>Number of outages in a month (based on pipeline steps of type <b>Prod Deploy</b>) linked to the associated business service in CMDB.</p> <p>This report provides an indication of environment stability.</p>
Mean Time to Resolve Trend	Database view joined by Incident, Change Request, Step Execution, Step, Pipeline, and App lists.	<p>Daily average resolve time for an incident caused by a DevOps change.</p>
Service Availability Trend	Database view joined by Service Availability, Service Offering, Business Service, Step, Pipeline, and App lists.	<p>Daily average service availability (based on pipeline steps of type <b>Prod Deploy</b>) linked to application service in CMDB.</p> <p>This report provides an indication of environment stability.</p>

## Operational Stability (continued)

Report	Source list	Description
Incidents Trend	Incident	<p>Daily number of incidents (based on pipeline steps of type <b>Prod Deploy</b>) linked to business service in CMDB.</p> <p>This report provides an indication of environment stability.</p>
Outage Trend	Outage	<p>Daily number of outages (based on pipeline steps of type <b>Prod Deploy</b>) linked to business service in CMDB.</p> <p>This report provides an indication of environment stability.</p>

## Development

### Development

Report	Source list	Description
Commit Frequency	Commit	<p>Number of commits measured daily.</p> <p>Smaller more frequent commits are preferred over larger less frequent ones.</p>
Average Branches per Repository	Branches, Repository	<p>Average branches per repository on a given day.</p> <p><b>Note:</b> This widget uses formula and does not support multi-element selection.</p>
Average Commits per Pipeline Execution	Commit, Pipeline Execution	<p>Average commits per pipeline on in the last 30 days:</p> $[[\text{Total number of commits}]] / [[\text{Number of pipeline executions}]]$

## Development (continued)

Report	Source list	Description
		<p><b>i Note:</b> This widget uses average aggregation and does not support multi-element selection.</p> <p>A low number is preferable, which indicates a concentrated effort, versus switching from task to task without completion.</p>
Commits without Work Item	Commit	<p>Commits made that are not tied to a work item, grouped by committer, in the last 30 days.</p> <p>This report is useful for investigating and resolving why a commit is not tied to a work item, because all commits should be tied to a work item.</p>
Work Items	Work Item	<p>Number of work items that are complete or working in progress in the last 30 days.</p> <p><b>i Note:</b> Filter is not applicable to this widget.</p>

## Commit Insights

### Commit Insights

Report	Source list	Description
Active Committers	Commit	<p>Committees that submitted commits in the last 30 days.</p> <p>Shows how many active committers there are.</p> <p><b>i Note:</b> This widget uses count distinct aggregation and does not support multi-element selection.</p>

## Commit Insights (continued)

Report	Source list	Description
Average Commits per Committer	Commit	<p>Total number of commits in the last 30 days / Active committers.</p> <p>Shows how often committers are committing. A higher value is more favorable.</p> <p><b>Note:</b> This widget uses formula and does not support multi-element selection.</p>
Average Files Added per Commit	Commit, Commit Details	<p>Total number of files added in the last 30 days / Total number of commits in the last 30 days.</p> <p>Shows how few files are committed at a time. A lower value is more favorable.</p> <p><b>Note:</b> This widget uses formula and does not support multi-element selection.</p>
% Commits Reverted	Commit	<p>Commits reverted in the last 30 days / Total number of commits in the last 30 days.</p> <p>Shows how many commits have been reverted. A lower value is more favorable.</p> <p><b>Note:</b> This widget uses formula and does not support multi-element selection.</p>
Top Committers	Commit	<p>Committers with the highest number of commits in the last 30 days.</p> <p>Provides visibility into the users that commit the most.</p>
Top Reverters	Commit	Committers with the highest number of reverts in the last 30 days.

## Commit Insights (continued)

Report	Source list	Description
		<p>Provides visibility into the users that revert commits the most.</p>
Commits Added per App	Commit, Commit Details	<p>Number of commits added per app in the last 30 days.</p> <p>Provides visibility into the development activity for each app.</p> <p><b>Note:</b> Filter is not applicable to this widget.</p>

## Deployments

Deployment frequency lets you know how often you are delivering value based on production deployments. Typically, more frequent deployments are desired.

**Note:** Metrics are based off production deployments (**Type** field is set to Prod Deploy in the app step).

## Deployments

Report	Source list	Description
Deployment Frequency - Monthly	Step Execution	<p>Number of successful production deployments in a month.</p> <p>Applies to steps of type <b>Prod Deploy</b> that are in completed state.</p>
Failed Deployments	Step Execution	<p>Number of failed production deployments in the last 30 days.</p> <p>Applies to steps of type <b>Prod Deploy</b> that are in failed or user-canceled state.</p>
Deployment Success Rate	Step Execution	<p>Deployments success rate over the last 30 days.</p> <p>Deployment Success Rate = (Number of Successful</p>

## Deployments (continued)

Report	Source list	Description
		<p>Deployments in the last 30 days / Total Number of Deployments in the last 30 days) * 100</p> <p>Applies to steps of type <b>Prod Deploy</b> in completed state.</p> <p><b>Note:</b> This widget uses formula and does not support multi-element selection.</p>
Average Lead Time	Pipeline Execution	<p>Average of:</p> $([\text{Time the code is successfully pushed to production}] - [\text{Earliest commit time}])$ <p>Applies to steps of type <b>Prod Deploy</b> that are in completed state.</p> <p><b>Note:</b> This widget uses average aggregation and does not support multi-element selection.</p>
Successful Production Deployments	Step Execution	<p>Frequency of successful production deployments over time broken down by app.</p> <p>More frequent production deployments are preferred.</p>
Failed Production Deployments	Step Execution	<p>Frequency of failed production deployments over time broken down by app.</p>
Commit-to-Deploy Lead Time	Pipeline Execution	<p>Duration from the earliest commit time to production deployment (for a successful pipeline execution).</p> <p>Minimizing the time it takes from committing code to successfully running it in production is preferable.</p>

## Deployments (continued)

Report	Source list	Description
		When the lead time is high, you can investigate the pipeline to identify the slowest steps. For example, a manual change approval process could increase lead time.

## System Health

### System Health

Report	Source list	Description
Task Execution Success Rate	Task Executions	<p>Success rate for tasks run by the execution tools over time:</p> $[[\text{Task Execution Success}]] / [[\text{Task Execution}]] * 100$ <p><b>Note:</b> This widget does not support multi-element selection.</p>
Number of Task Executions	Task Executions	<p>Default number of tasks executions in the last 30 days.</p>
Number of API calls	Event	<p>Default number of API calls in the last 30 days.</p> <p><b>Note:</b> Filter is not applicable to this widget.</p>

## DevOps System Health dashboard

The System Health dashboard lets the DevOps administrator view the overall health of integrations, connectivity status, as well as view trends of inbound event processing data.

**Note:** The DevOps System Health dashboard is available only in DevOps Data Model 1.33 and later versions.

Use the System Health dashboard to visually track the health, event status, and connectivity status of your DevOps environment. Interactive charts enable you to view and analyze the system health and inbound event processing details.

**Note:**

- Scores are not real-time. For example, if chart displays that two events are in the waiting state, the events could have already moved to the completed state.
- Because the scores are fetched from formula indicators, you cannot drill down to the records in the KPI page.

## DevOps System Health Dashboard

### Events

Report	Type	Description
Event count	Count trend	Daily trending total count of DevOps events.  The report displays the percentage and count which is the last 7 running sum.  Total count includes events from [sn_devops_inbound_event, sn_devops_event, and the sn_devops_processed_event] tables.
Events by state	Pie chart	Total count DevOps events distributed by current state.
Events by tool	Pie chart	Total count of DevOps events, distributed by tool.
Retry count	Count trend	Daily trend of the count of total retried DevOps.  The report displays the percentage and count variance in the last 24 hours.
Processed count	Count trend	Daily trend of the total count of processed DevOps events.  The report displays the percentage and count variance in the last 24 hours.
Error count	Count trend	Daily trend of the total count of DevOps events in error state.  The report displays the percentage and count variance in the last 24 hours.

Report	Type	Description
Waiting count	Count trend	Daily trend of the total count of DevOps events in waiting state.  The report displays the percentage and count variance in the last 24 hours.
Event trend	Line trend	Total trending count of DevOps events.  The report displays the percentage and count variance in the last 24 hours.
Processing duration	Line trend	Processing duration of events (in seconds) measured daily.

## Tool Connectivity tab

The Tool connectivity status report displays the connectivity status with details of all the integrated tools. The tool connectivity table displays the connection status for each connected tool. A background job tries to connect to the tools every hour and maintains the connection history. You can view the historical details under a new related list on the tools page.

## System Health Email Notifications

A base system DevOps Report group is added by default. An email notification summarising the weekly System Health report is sent to the users in this DevOps Report user group which is specified, in the **Who will receive** related list of the **System Notifications > Email Notifications** form, as configured in the base system DevOps Health Report notification. For more information, see [Email notifications](#).

### **Note:**

- Add users with DevOps Admin (sn\_devops.admin) role to the DevOps Report user group.
- You need to remove a user from the group to unsubscribe the user from receiving notifications.

The base-system email notification contains and displays a comparative digest of key system health performance indicators as compared to their performance in the previous weekly report. The comparative data is sourced from relevant tables and calculated based on required indicators using the date range specified in weekly report. The comparative data and indicators always display with respect to the same tables and indicators vis-a-vis, the data collected against the relevant reports and indicator data for the previous week.

To configure the frequency in which the base system email notification is sent out. Navigate to **System Scheduler > Scheduled Jobs > Trigger DevOps Health Check Email Notification** and click the **Configure Job Notification** related link, and make the schedule changes.

If you want to modify the users who receive the email notification, modify the user group specified in the **Groups** field of the **Who will receive** section. Navigate to **System Notification > Email > Notification > DevOps Health Script**. For more information, see [Groups](#).

Use the **Preview Notification** option to check your notification.

For example, you can see which users will or will not receive the notification, along with what instance details and period for which the notification is sent.

## DevOps Pipeline UI

Use the Pipeline UI to visualize interactions and results across a pipeline execution. This graphical view shows pipeline step progression and other details for each pipeline.

From DevOps, get a quick view of how everything is connected to see exactly what is happening with the pipeline and when. From the ServiceNow Change Management application, you can access the Pipeline UI and quickly see the commits, the committers, and other details for the change request in one place.

The Pipeline UI displays parallel stages as modeled in Azure DevOps release pipelines. The pipeline UI displays the real-time state of the pipeline as it appears in Azure DevOps. The associated artifact details sourced from the build pipeline, Test Results, Software Quality Summary Results also display on the pipeline UI. For more information, see [Parallel stages in Azure DevOps release pipelines](#)

The Pipeline UI shows all attempts of any stage or job that has been rerun or restarted. For more information, see [Restarting failed build or release pipeline jobs and stages](#)

The Pipeline UI shows the pipeline steps that ran instead of the steps configured in DevOps.

You can access the Pipeline UI using the related link from within certain DevOps forms, and also from a DevOps change request form:

- DevOps Pipeline form
- DevOps Pipeline Execution form
- Change Request form created by DevOps

**i Note:** You must reload the view to update the status buttons in the pipeline execution History.

### Step run states (task execution status)

Green	Successful. All step executions associated to the pipeline execution passed.
Grey	Not yet run.
Yellow	Waiting (pending, building, validating). At least one step execution is waiting.
Red	Failed. At least one step execution failed.

## Step run states (task execution status) (continued)

Task execution end date is populated even when the change is rejected.

**i Note:** To enable the canceling of the change request associated with the step when the step fails, you must set the `sn_devops.cancel_change_on_pipeline_cancel` property to Yes. For more information, see [Properties installed with DevOps](#).

**i Note:** The order the cards appear in the Pipeline UI is determined by the **Order** field in each pipeline step when you modeled your pipeline in DevOps. Skipped stages are not shown.

The order the cards appear in the Pipeline UI by task execution.

UI feature	Description
Pipeline steps	<p>Timing.</p> <ul style="list-style-type: none"> <li>Start</li> <li>Last run</li> <li>Duration for each step</li> </ul> <p>When the downstream task execution starts immediately after the upstream task execution, the duration is 0 seconds.</p> <ul style="list-style-type: none"> <li>Wait times in between task executions.</li> </ul> <p>Wait time is calculated as:</p> <p>Start time of the task execution minus the end time of the upstream task execution.</p>
View change request	<p>Change request record.</p> <p>Click directly into the change request of the step that was created by DevOps to view details of the change and take action.</p> <p><b>i Note:</b></p> <ul style="list-style-type: none"> <li>The change request record for the associated with the latest task execution displays.</li> <li>Commits reverted in the same pipeline execution are not shown in the commit list.</li> </ul>

UI feature	Description
Pipeline history	<p>Pipeline Execution.</p> <p>Click a history tile to view the previous step details for a pipeline execution.</p>
View all attempts	<p>All attempts that the job has run in a step.</p> <p>Click the link in the relevant step to view all attempt details.</p>
Artifacts	<ul style="list-style-type: none"> <li>• Artifact versions             <ul style="list-style-type: none"> <li>◦ Work items</li> <li>◦ Commits</li> <li>◦ Packages</li> </ul> </li> <li>• Commits</li> <li>• Work items</li> </ul> <p><b>i Note:</b> Commits reverted in the same pipeline execution are not shown in the commit list.</p>
Quality	<ul style="list-style-type: none"> <li>• Test Results.</li> </ul> <p>View the build test results to see what tests passed or failed.</p> <p>The quality card contains test summaries:</p> <ul style="list-style-type: none"> <li>◦ Test type and test category in the format:</li> </ul> <pre>test type/test category</pre> <ul style="list-style-type: none"> <li>◦ Native ID of the step</li> <li>◦ Test pass percentage (unit and functional tests only)</li> <li>◦ Throughput (performance tests only)</li> <li>◦ Step name</li> </ul> <ul style="list-style-type: none"> <li>• Software Quality Results</li> </ul> <p>View all the software quality (SonarQube scan) results grouped by project name that were fetched as part of the selected pipeline. You can view the scan results for all categories in a pipeline execution step.</p> <p>You can customize the priority of categories from the <b>DevOps &gt; Administration &gt; Properties &gt; DevOps Properties Category &gt; Software quality categories shown by default in the Pipeline</b></p>

UI feature	Description
	<p><b>UI view</b> field in a comma separated format.</p> <ul style="list-style-type: none"> <li>◦ Click the gear icon to modify the category view in the pipeline UI.</li> <li>◦ Click the step execution record to view the corresponding Software Quality Scan Summary record.</li> <li>◦ Click the Vulnerabilities count record to view the Software Quality Scan Detail record and the corresponding sub category details.</li> </ul> <p><b>i Note:</b> When you reattempt running a stage or a pipeline job which contains a test or a software quality scan, the results get appended with the attempt number.</p>

Click directly into DevOps change requests, step executions, artifacts, artifact versions, work items, test summaries, and reattempts in flyout windows.

#### Pipeline UI — Artifact version flyout

#### Pipeline UI - Reuse change request - Step execution flyout

#### Azure DevOps- Pipeline UI with parallel stages

### Populate the Change Request Repo Detail table

Run the *Load Change Request Repo Detail* job to populate the Change Request Repo Detail table if you import demo data after installing the DevOps Insights application.

#### Before you begin

Role required: admin

#### About this task

The job executes asynchronously, and it might take a minute after script execution completes to populate data in the table.

#### Procedure

1. Navigate to **Scheduled Jobs** and search for the *Load Change Request Repo Detail* job.
2. Select **Execute Now**.

### Group DevOps applications into a product

Products that use an application model in the CSDM support hierarchies of applications. You can customize hierarchies to simplify tracking of "rolled-up" data on DevOps Insights reports. This is used for the Product filter in insights.

## Before you begin

Role required: application owner, sn\_devops.admin

## About this task

For example, you can include multiple DevOps microservice applications into a product, include multiple such products into a portfolio, and then include ("roll up") the portfolio in an organization. Another example structure might be **application > team > product > portfolio or business unit**.

- A DevOps application can belong to one or multiple products.
- A product or multiple products can belong to one or multiple other products.
- Multiple applications and products can belong to a product.

For every product, the system creates a corresponding entry in the application model and SDL component tables.

**Important:** To add an application to a product, the application must hold execution data. For a newly added application, therefore, you must wait until the data import job has run.

## Procedure

1. Create applications as described in .
2. Open the application models table: <instanceName>/cmdb\_application\_product\_model\_list.do and then follow this procedure to configure each application that will be included in a product.
  - a. In the Application models list, select the application model to open the record.
  - b. On the Application model form, in the **Model categories** field, select the appropriate category.
  - c. Save the record.
3. Follow this procedure to create a product that will act as a parent.
  - a. In the Application models list, select **New**.
  - b. On the Application model form, in the **Name** field, enter the name of the product.
  - c. In the **Model categories** field, select **Bundle**.
  - d. Optional: Enter a short description.
  - e. Save the record.
4. Open the Model category of component table (<instanceName>/cmdb\_m2m\_model\_component\_list.do) to specify the category of each application (the application is the component in this case).
5. For each application, select **New**, specify the following settings, and then submit the record.

### Model component form

Field	Description
Model category of component	The <b>Model categories</b> value that you specified for the application on the Application model form.
Component	The application.

Field	Description
Bundle	The product that will act as a parent of the application.

The application is now a member of the specified product.

## Result

The DevOps Insights tabs provides filters for the reports.

- The scheduled "Data collection" job processes your changes. When the job finishes, you can view reports for the added products. If you are running the job manually, execute the "Update Repo Details and Work Item State Detail" job before executing the Data collection job.
- The **Application** filter lists all applications.
- The **Product** filter lists all applications plus all products.
- To view the members of a product, view the list in the Model category of component table (<instanceName>/cmdb\_m2m\_model\_component\_list.do). Applications are listed in the **Component** column and products are listed in the **Bundle** column.

## DevOps Insights workspace properties

DevOps Insights properties configure how DevOps Insights workspace data is displayed in reports.

You can view the Insights properties from the DevOps Change workspace by navigating to **Workspaces > DevOps Change Workspace > System configuration > Insights properties**.

### **Note:**

If you import demo data after installing the DevOps Insights application, follow the instructions in [Populate the Change Request Repo Detail table](#) before configuring properties.

## base-system properties installed with DevOps Insights

### DevOps Insights workspace properties category

Property	Description	Default
X hours per Developer time	<p>The time taken per developer to work on a single work item in hours.</p> <p>This value is used to calculate the Developer hours saved report.</p> <p>For example, for every work item associated to a change request, a developer saved X hours in having to associate the work items, commits, test results, and other related DevOps data for their work.</p>	1

## DevOps Insights workspace properties category (continued)

Property	Description	Default
Change Request Awaiting States	<p>The states that are reported as awaiting a change request in the Changes awaiting approval report.</p> <p>Use the following values for the other states:</p> <ul style="list-style-type: none"> <li>• New: -5</li> <li>• Assess: -4</li> <li>• Authorize: -3</li> <li>• Schedule: -2</li> <li>• Implement: -1</li> <li>• Review: 0</li> <li>• Closed: 3</li> <li>• Canceled: 4</li> </ul>	-5, -4
Average Hourly Developer Cost	<p>The average hourly cost of a developer working on DevOps changes in the selected currency.</p> <p>This value is used to calculate the Change acceleration savings report, where we multiply the Developer hours saved by this developer cost to estimate an ROI.</p>	100

## DevOps Change Velocity reference

Reference information to provide additional details about DevOps Change Velocity such as tables, users, roles, properties, errors, limitations, and others.

### Components installed with DevOps Change Velocity

Several types of components are installed with DevOps Change Velocity, including tables, users, roles, and scheduled jobs.

Demo data is available for this application.

### Users installed

User	Description
devops.integration.user	DevOps Integration User

User	Description
	<b>Note:</b> You must configure the password before the DevOps Integration User can configure a tool.
devops.system	DevOps System user

## Roles installed

Role title [name]	Description	Contains roles
DevOps Administrator [sn_devops.admin]	Sets up and configures the DevOps application.	<ul style="list-style-type: none"> <li>• sn_devops.app_owner</li> <li>• connection_admin</li> <li>• action_designer</li> <li>• credential_admin</li> <li>• flow_designer</li> <li>• sn_devops.integration</li> <li>• sn_devops.tool_owner</li> </ul>
DevOps integration [sn_devops.integration]	Has inbound access to the tools in your environment to allow integration with the DevOps application.	<ul style="list-style-type: none"> <li>• flow_operator</li> <li>• cmdb_read</li> </ul>
DevOps App Owner [sn_devops.app_owner]	<p>Oversees the operation of the DevOps application and monitors performance in your DevOps environment.</p> <p>The app owner can:</p> <ul style="list-style-type: none"> <li>• Create, update, and delete applications only.</li> <li>• Associate or dissociate objects to applications (pipelines, repos, plans, artifact repository)</li> <li>• Perform change automation and modify pipeline steps.</li> <li>• Access to see all the tools but not modify them.</li> <li>• Can click discover from tool records to find new plans, repos, or pipelines from already connected tools.</li> <li>• Has the viewer role.</li> </ul>	<ul style="list-style-type: none"> <li>• sn_devops.viewer</li> <li>• cmdb_read</li> </ul>

Role title [name]	Description	Contains roles
DevOps Tool Owner [sn_devops.tool_owner]	<p>The tool owners can:</p> <ul style="list-style-type: none"> <li>• Create new tools.</li> <li>• Perform the following actions on the tool: <ul style="list-style-type: none"> <li>◦ Discover (classic UI and workspace)</li> <li>◦ Enter Manual Configuration Mode (classic UI and workspace)</li> <li>◦ Configure (classic UI and workspace)</li> <li>◦ Configure Guidance (Jenkins only, workspace)</li> <li>◦ Update Credentials (workspace)</li> <li>◦ Check Permissions (workspace)</li> <li>◦ Delete tool (classic UI and workspace)</li> </ul> </li> <li>• View System health and Troubleshooting sections of the Administration module and view Insights dashboards.</li> <li>• On the tool record page, the Related Links tabs will show Tool connections and Tool connectivity history.</li> <li>• If users have both Tool Owner and App Owner roles, they can also quest automations.</li> </ul> <p>Tool owners can update credentials only for tools in which they are a member of the access groups, or for tools without access restrictions.</p>	<ul style="list-style-type: none"> <li>• connection_admin</li> <li>• sn_devops.viewer</li> <li>• flow_designer</li> </ul>
DevOps viewer [sn_devops.viewer]	Has access to the DevOps application to use in their environment.	<ul style="list-style-type: none"> <li>• sn_devops_ws.workspace_user</li> <li>• cmdb_read</li> </ul>

## Scheduled jobs installed

Scheduled job	Description
[DevOps] Historical Data Collection	<p>Collects data from DevOps tools on demand.</p> <p><b>i Note:</b> You must run this scheduled job to see historical DevOps data after installation of the DevOps Insights application.</p>
[DevOps] Daily Data Collection	Collects data from DevOps tools daily.
DevOps auto discover tool associations	<p>Automatically discovers new pipelines, plans, and repositories for all the tools that are in connected state and for all the capabilities. This job is scheduled to run once daily.</p> <p>You can deactivate or change the frequency of the schedule:</p> <ol style="list-style-type: none"> <li>1. Navigate to <b>System Definition &gt; Scheduled Jobs</b> and select <b>DevOps auto discover tool associations</b>.</li> <li>2. To deactivate the job, deselect the <b>Active</b> check box.</li> <li>3. To change the frequency, update the <b>Repeat Interval</b>.</li> </ol>

## Tables installed

Name	Table
API Schema Definition	[sn_devops_api_schema_definition]
App	[sn_devops_app]
Artifact	[sn_devops_artifact]
Artifact Repository	[sn_devops_artifact_repository]
Artifact Staged Request	[sn_devops_artifact_staging]
Artifact Version	[sn_devops_artifact_version]
Base Planning Item	[sn_devops_base_planning_item]
Branch	[sn_devops_branch]
Build Test Result	[sn_devops_build_test_result]
Build Test Summary	[sn_devops_build_test_summary]

Name	Table
Callback	[sn_devops_callback]
Commit	[sn_devops_commit]
Committer	[sn_devops_committer]
Commit Details	[sn_devops_commit_details]
Contributor Score Change Factor	[sn_devops_contributor_score_chg_factor]
Environment	[sn_devops_environment]
Event	[sn_devops_event]
Event Processor	[sn_devops_event_processor]
Import Filter	[sn_devops_import_filter]
Import Request	[sn_devops_import_request]
Import Request Page	[sn_devops_import_request_page]
Inbound Event	[sn_devops_inbound_event]
Integration Capability	[sn_devops_integration_capability] Extends table Application File.
App to Plan	[sn_devops_m2m_app_plan]
Deployed Artifact to TaskExecution	[sn_devops_m2m_artifact_execution]
Artifact Version to Commit	[sb_devops_m2m_artifact_version_commit]
Artifact Version to Package	[sn_devops_m2m_artifact_version_package]
Branch To Commit	[sn_devops_m2m_branch_commit]
Run Commit	[sn_devops_m2m_commit_execution]

Name	Table
Work Item To Plan Version	[sn_devops_m2m_work_item_plan_version]
Orchestration Task	[sn_devops_orchestration_task]
Orchestration Task Definition	[sn_devops_orchestration_task_definition]
Package	[sn_devops_package] Extends table Configuration Item.
Participant	[sn_devops_participant]
Pipeline	[sn_devops_pipeline]
Pipeline Execution	[sn_devops_pipeline_execution]
Plan	[sn_devops_plan] Extends table Base Planning Item.
Plan Version	[sn_devops_plan_version] Extends table Base Planning Item.
Repository	[sn_devops_repository]
Step	[sn_devops_step]
Step Execution	[sn_devops_step_execution]
Tag	[sn_devops_tag]
Task Execution	[sn_devops_task_execution]
Test Execution	[sn_devops_test_execution]
Test Result	[sn_devops_test_result]
Test Type	[sn_devops_test_type]
DevOps Tool	[sn_devops_tool]

Name	Table
Tool Action	[sn_devops_tool_action]
Tool Capability Mapping	[sn_devops_tool_capability_mapping]
DevOps Tool Integration	[sn_devops_tool_integration] Extends table Application File.
Tool Type Capability	[sn_devops_tool_type_capability]
Work Item	[sn_devops_work_item] Extends table Base Planning Item.

## Assign roles and tasks using Workspace

Onboard directly from the DevOps Change Workspace by assigning roles and tasks for users and groups.

### Before you begin

Role required: sn\_devops.admin

### Procedure

1. Navigate to **Workspaces > DevOps Change Workspace**.
2. Use one of the following ways to onboard users and groups:
  - From the Home page, from the Accounts and users widget, select **Assign roles** or **Assign users**.
  - From the left navigation panel, navigate to **Administration > Onboarding** and select the onboarding task.
3. Onboard users and groups by assigning them roles and tasks.

Onboarding task	Steps
Assign user roles	<ul style="list-style-type: none"> <li>a. Select <b>User roles</b>.</li> <li>b. From the User roles page, select <b>Assign role</b>.</li> <li>c. Select the users that you want to assign a specific DevOps role to.</li> <li>d. Select the DevOps role from the list.</li> </ul> <p>For more information on the roles, see <a href="#">Roles installed</a>.</p> <ul style="list-style-type: none"> <li>e. Select <b>Done</b>.</li> </ul>
Assign group roles	<ul style="list-style-type: none"> <li>a. Select <b>Group roles</b>.</li> <li>b. From the Group roles page, select <b>Assign role</b>.</li> </ul>

Onboarding task	Steps
	<p><b>c.</b> Select the group for which you want to assign a specific DevOps role.</p> <p><b>d.</b> Select the DevOps role from the list.</p> <p>For more information on the roles, see <a href="#">Roles installed</a>.</p> <p><b>e.</b> If you want to also assign the same role to the child groups of the selected group, then select the <b>Assign selected role to child groups</b> option.</p> <p><b>f.</b> Select <b>Done</b>.</p>
Assign user tasks	<p><b>a.</b> Select <b>User tasks</b>.</p> <p><b>b.</b> From the User tasks page, select <b>Assign task</b>. If you want to assign the task to yourself, select <b>Assign to me</b>.</p> <p><b>c.</b> Select the task from the <b>Task</b> list. For example, create an application or connect a tool.</p> <p><b>d.</b> Select the user from the list. Only users whose roles allow them to perform the task are available in the list.</p> <p><b>e.</b> Enter a due date and add your comments for the task if required.</p> <p><b>f.</b> Select <b>Done</b>.</p>
Assign group tasks	<p><b>a.</b> Select <b>Group tasks</b>.</p> <p><b>b.</b> From the Group tasks page, select <b>Assign task</b>. If you want to assign the task to yourself, select <b>Assign to me</b>.</p> <p><b>c.</b> Select the task from the <b>Task</b> list. For example, create an application or connect a tool.</p> <p><b>d.</b> Select the group from the list. Only groups whose roles allow them to perform the task are available in the list.</p> <p><b>e.</b> Enter a due date and add your comments for the task if required.</p> <p><b>f.</b> Select <b>Done</b>.</p>

## DevOps Change Velocity Properties

Use these properties to configure settings in the DevOps Change Velocity application.

Role required: sn\_devops.admin.

You can view the properties from the DevOps Change workspace by navigating to **System configuration > Properties**.

Property	Description	Default
[sn_devops.max_retry_count_inbound]	Maximum number of retries for errored inbound events.	3
[sn_devops.tool_capabilities]	The tool capabilities supported, entered as comma-separated values.	code,plan,orchestration,artifact,test
[sn_devops.change_request.approval_text]	Approved change request approval text.  This property is used for change management customizations.	Approved
[sn_devops.health_duration_report]	Duration of the DevOps system health report (in days). Defaults to the last 7 days, to show system health metrics like inbound events.	7
[sn_devops.import.max.pages]	Max number of pages to process at a time for an import request.	10
[sn_devops.change_request_reuse]	The ability to check if an existing change can be reused, instead of creating a new one.	sn_devops.change_request_reusability_subflow
[sn_devops.github.url]	For GitHub, this field is used to get the API URL (for REST calls).	https://github.com
[sn_devops.change_request.cancel_state]	DevOps state change request cancel state.  This property is used for change management customizations.	4 (Canceled)
[sn_devops.import.orchestrations]	Max number of orchestrations executions while importing Jenkins freestyle jobs.	1000
[sn_devops.import.save.payload_as_attachment]	JSON payload is saved as an attachment in the Import Request page.  To save payloads as attachments on the Import Request Page record, set the <b>Value</b> field to true. Anything else is considered false.	False (disabled)
[sn_devops.non_admin_software_encryption_view_flag]	Encryption view flag for SonarQube scans configured	False (disabled)

Property	Description	Default
	on your GitHub Actions, Jenkins, or Azure DevOps pipelines as a non-admin SonarQube user.	
[sn_devops.default_test_type]	Default test type from orchestration pipelines.	JUnit
[sn_devops.import.coding_tool_branches_per_page]	Number of coding tool branches per page.	19
[sn_devops.import.planning_tool_issues_per_page]	Number of planning tool issues displayed per page.	100
[sn_devops.discovered.user.automatic_snigdevops_scope]	The <code>snigdevops</code> role entered in the value field is automatically added to users who are active DevOps users (for example, making a commit).  If a role that is not in the <code>sn_devops</code> scope is provided, it is not added.  Leave empty if no role should be automatically provided.	sn_devops.viewer
[sn_devops.sq_ui_category_preferences]	quality categories shown by default in the Pipeline UI view, entered as comma-separated values.	coverage,lines_of_code,bugs,code_smells,du...
[sn_devops.inbound_events_retry_error_exceptions]	Error exceptions for which the errored inbound events are set to Retry. Entered as comma-separated values.	TimeOutException,FlowObjectAPIException
sn_devops.devops_log_level	The DevOps log level.  Select the appropriate level from the following: <ul style="list-style-type: none"><li>• Error</li><li>• Warning</li><li>• Information</li><li>• Debug</li><li>• Trace</li></ul>	Warning
[sn_devops.custom_change_category]	Categorize DevOps change requests on <b>DevOps Change</b> field.  Select this option to categorize change requests with category field set to DevOps	False (disabled)

Property	Description	Default
	as a DevOps change. Clear to disable.	
[sn_devops.import.orchestration_number_of_import_executions_per_page]	Number of import executions per page.	50
[sn_devops.enable_import_polling]	Option to enable polling of import requests. Clear to disable polling.	False (disabled)
[sn_devops.github.api_version_path]	GitHub API version path.	/api/v3
[sn_devops.permission_check_timeout]	Permission check timeout per record (in milliseconds)	10000
	During the tool connection process, the system performs permission validations in the foreground, and restricts the tool connection page. For every permission validation, a REST API call is made to external tools to evaluate the availability of the permission. The REST API response time may vary based on your network settings. To avoid delays, you can specify how long the system should wait to get a response from an external tool before marking the permission as an error through this property.	
[sn_devops.devops_reused_model_change_request_reusability_subflow]	The DevOps change request reusability model subflow.	sn_devops.devops_reused_model_change_request_reusability_subflow
[sn_devops.inbound_events_error_retry_time_in_minutes]	The time in minutes. The retry job checks for errored inbound events from the time/value specified.	1440 (7 days)
[sn_devops.cancel_change_on_pipeline_failure]	Cancels the change request when the associated stage in the pipeline fails or is canceled.  When this property is enabled and the orchestration task mapped to a certain step in your DevOps pipeline fails or is canceled, the change request associated with that step is also canceled. A change request can be canceled only if it has not	False (disabled)

Property	Description	Default
	already been rejected or is not in the implement or review state at the time of cancellation.	
[sn_devops.cascade_delete_threshold]	delete threshold (recommended foreground limit 1000).	1000
[sn_devops.import.max.retries]	retries allowed per page while importing data from tools.	3
[sn_devops.change_request.post_implement_state]	request post implement state.  This property is used for change management customizations.	0 (Review)
[sn_devops.change_request.implement_state]	request implement state.  This property is used for change management customizations.	-1 (Implement)
[sn_devops.committer.score.default]	default committer score.	50
[sn_devops.committer.score.multiply_factor]	Multiply factor for committer score.	1
[sn_devops.table_auto_archive_duration]	Archive, in months. The duration after which the table data is to be auto-archived.	9
[sn_devops.enable_debug]	Deprecated - Enable Debug Flag.  <b>i Note:</b> This property is deprecated, use the <code>sn_devops.devops_log_level</code> (DevOps log level) property instead.  DevOps Debug Logger.	Yes
[sn_devops.import.max.processing_time]	in seconds, that should be allowed per page while importing data from tools.	300
[sn_devops.supported_webhook_types]	abilities supported by DevOps.	code,plan,orchestration,artifact,test
[sn_devops.change_request.closed_state]	closed state.  This property is used for	3

Property	Description	Default
	change management customizations.	
[sn_devops.bulk_flow_timeout]	Bulk flow timeout (in milliseconds)	60000
[sn_devops.import.coding_tool]	Coding tool projects per page.	100
[sn_devops.discover.jenkins.folder_depth]	Folder Depth  Retrieves only those orchestration tasks and pipelines which are in folders whose nesting level is less than or equal to the number specified here. Folder Depth is the level of nesting done on folders in Jenkins, which contains the orchestration task and pipelines that you want to discover. For example, if you want to discover orchestration tasks and pipelines for a folder structure that is nested 2 times in your Jenkins environment, you must enter 2 as the value for this property.	3
[sn_devops.change_request_handler_subflow]	Change Request Handler subflow.	sn_devops.default_change_handler_subflow
[sn_devops.import.coding_tool]	Coding tool repositories per page.  The number of repositories that must be displayed per page when you import repositories after connecting to a tool. The total number of repositories imported is not dependent on this property, but the number of repositories that must be displayed per page from the total number of repositories imported can be specified using this property. For example, if your tool contains 1000 repositories and you specify 100 as the property value, 1000 repositories will be imported of which 100 will be displayed per page in the Import Requests page.	100

Property	Description	Default
	When you discover repositories, the number of repositories that must be retrieved per API call in the backend is also dependent on this property value. For a large number of repositories, set a relatively smaller value in this property so that lesser number of repositories are retrieved per call and the system does not time out.	
[sn_devops.github.api_url]	GitHub API URL.	<a href="https://api.github.com">https://api.github.com</a>
[sn_devops.import.planning_to_start_date_per_tool_limit]	Management (SPM) Agile planning tool issues import max limit.	1000
[sn_devops.credential_expiration_notify_universal]	<p>Assignify universal [sn_devops.credential_expiration_notify_universal]</p> <p>and notify to update tool credentials when expired.</p> <p>Admins and tool owners will be notified (through universal task, email, banner, and field message) on expiry of tool credentials after the credentials have expired. As an admin or tool owner, you can update your tool credentials and connect the tool to prevent any further loss of data. For information on updating tool credentials, see <a href="#">Update third-party tool credentials in DevOps Change Velocity</a>.</p>	Yes (enabled)
[sn_devops.credential_expiration_notify_days_before]	<p>Notify days before tool credential expiry to assign a universal task and notify (if applicable)</p> <p>Set the number of days before tool credentials expiry to send notifications (through universal task, email, banner, and field message) to admins for GitHub tools created with basic authentication. To stop sending proactive notifications, select 0 as the value for this property. For</p>	3

Property	Description	Default
	information on updating tool credentials, see <a href="#">Update third-party tool credentials in DevOps Change Velocity</a> .	
[sn_devops.last_event_received_warning]	<p><b>Number</b> of days to display warning alerts when events were not being received.</p> <p>Set the number of days to display warning alerts in the <b>Last event received</b> field in the tool record when events were not being received. This only applies for tools which sends data to ServiceNow. The <b>Last event received</b> field in the tool record header and inside the tool record will be highlighted in yellow for warning alerts when the events were not being received.</p>	2
[sn_devops.last_event_received_critical]	<p><b>Number</b> of days to display critical alerts when events were not being received.</p> <p>Set the number of days to display critical alerts in the Last event received field in the tool record when events were not being received. This only applies for tools which sends data to ServiceNow. The <b>Last event received</b> field in the tool record header and inside the tool record will be highlighted in red for critical alerts when the events were not being received.</p>	7

## Onboard a new tool using DevOps generic playbook

DevOps Change has a pre-build playbook setup for users to configure a new tool.

Playbooks provide end users with a visual, task-oriented guide with the steps to complete a process, ensuring a consistent experience for tool onboarding. For detailed information about playbooks, see [Playbook](#).

Perform the following steps to onboard your tool using the DevOps generic playbook.

## 1. Identify tool capabilities

To configure a tool using the generic playbook, first you must identify the capabilities supported by the tool. A tool can have one or more capabilities like code, plan, and orchestration.

The Generic playbook consists of the following stages and each stage has its related activities:

### 1. Connect to a tool

- a. Connect to a tool
- b. Specify tool access
- c. Configure the tool

### 2. Capability

Capability can be of type plan, code, or orchestrate. If the tool has multiple capabilities, multiple stages exist with the capability name.

- a. Select to track
- b. Import data
- c. Associate (This activity is for the Orchestration tools to associate pipeline steps)

### 3. Summary

This is how a playbook with multiple capabilities looks like:

## 2. Configure Activity Definitions

Each Activity in a playbook is associated to an activity definition. DevOps has the following activity definitions:

- DevOps CreateTool AD
- DevOps Configure & Test AD
- DevOps Select Associated Objects AD
- DevOps Import Data AD
- DevOps Associate Services AD
- DevOps Summary AD

For more information, see [Process Automation Designer](#) and [Playbook](#).

The DevOps table `sn_devops_capability_activity_mapping` has been created to maintain associations between Activity definitions and capabilities. This table configuration is used to dynamically render the stages and activities for a tool. You can update the conditions as needed to either enable or disable a particular activity for a tool.

For example, tools like GitLab and JFrog don't support data import. So you don't require the Data Import activity for such tools. In that case, you must add the logic in the condition column to return false for these types of tools. See the following image as a reference:

### 3. Configure Activity UI for the Activity Definition

Playbook Activity UIs define the experience type and UI template rendered to users while managing Playbooks. You can configure multiple Activity UIs for an Activity Definition and render any one dynamically based on the condition evaluation.

DevOps has the following AUIs for Create Tool activity. Except this, all other activity definitions have only one Activity UI.

You can configure any one the Activity UI based on the requirement. Refer to [to navigate the Playbook Activity UI](#).

Identify the required activity UI and add your tool in the condition builder so that the UI gets effected in the playbook for the tool.

#### Result

After completing these steps, your tool can be onboarded using the DevOps generic playbook.

## Additional information for onboarding DevOps tools and apps using the Service catalog

Use the onboarding catalog items as a self-service approach to onboard your DevOps tools and apps.

### Onboarding tools and apps

**Note:** This content pertains to the Classic Environment, which refers to working in lists of records and on record forms directly, not in [Configurable Workspace interface](#). You can work in the Classic Environment with Next Experience active, or with it inactive, which is referred to as Core UI, (formerly known as UI16).

Use the ServiceNow Service Catalog to request the onboarding of tools and apps. Fill out the onboarding form details for a tool or an app and submit the request. You can also import the DevOps data for the app that you are creating for swifter and easier onboarding.

Prior to the request creation, an automated workflow approves or rejects the request.

- When the workflow is approved, a success message displays.
- When rejected, an inbound event is created capturing error logs. By ensuring the log is error free, you can create a request again.

You can onboard DevOps tools and apps using self-service catalog items.

- Navigate to **All > Service Catalog > Catalog Definitions > Maintain Items > DevOps App Onboarding**, and set the **Active** state to **True** to activate the **DevOps App Onboarding** maintain item. By default the catalog items (App Onboarding) is turned off.
- Navigate to **All > Service Catalog > Catalog Definitions > Maintain Items > DevOps Tool Onboarding**, and set the **Active** state to **True** to activate the **DevOps Tool Onboarding** maintain item. By default the catalog items (Tool Onboarding) is turned off.
- To add the DevOps catalog items in the Service Catalog categories in Service Portal, set the category of these items to **DevOps** and add the **Service Catalog** catalog. If the **DevOps** category is not available, you must create a new category called **DevOps** from the catalog item record itself and assign the **DevOps Onboarding** catalog to it.

- To add the DevOps catalog items in Employee Service Center, add a new topic in the **Assigned Topics** related list with taxonomy as **Employee** in the **IT > IT for IT** section. The Assigned Topics related list is not available in the Default view of the Catalog Item form. To add it, select **Additional icons > Configure > Related Lists** and move the **Connected Content** → **Catalog item** field to the selected section.
- Navigate to **All > Flow Designer > Flows**, and activate the **Request for onboarding approval** flow, so that the tool or app onboarding requests are approved by default. If you want to request manual approval for onboarding catalog items (i.e. from a user other than the DevOps system user), you can update the rule set in the **Ask For Approval** action of the flow. You can configure manual request approval for such items by configuring the fulfillment process for the catalog item. For more information, see [Service Catalog request fulfillment](#).
- Customize roles for approvals of the request for onboarding approval flow from Flow Designer.

## Onboarding at scale

You can also use the onboarding APIs to onboard DevOps tools & apps in bulk instead of onboarding one tool or app per request. In the request parameters for tools or apps, you can specify multiple values to onboard them in one go. For example:

```
{
  "tools": [
    {
      "name": "jira_revamp",
      "type": "Jira",
      "url": "http://jiral.sndevops.xyz",
      "username": "admin",
      "password": "DevOps1!",
      "useMidServer": false
    },
    {
      "name" : "azure_revamp",
      "type" : "Azure DevOps",
      "url" :
      "https://dev.azure.com/ADOLightweight/Testing%20ADO%20On%20empolugu",
      "username" : "devops.integration.user",
      "password" :
      "a5xvoea2osy3ld43p2biojcu6eog5y5q3xicqbgbgxwuphjbbcu6a",
      "useMidServer" : false
    },
    {
      "name" : "jenkins_revamp",
      "type" : "Jenkins",
      "url" : "http://jenkins5.sndevops.xyz/",
      "username" : "admin",
      "password" : "DevOps1!",
      "useMidServer" : false
    },
    {
      "name" : "github_revamp",
      "type" : "GitHub",
      "url" : "https://api.github.com",
      "username": "admin",
      "password": "ghp_GMWQCwbIHJ07WHz2XSR0BQGESx3TIq2ZY380",
      "useMidServer" : false
    }
  ]
}
```

```

        },
        {
            "name" : "bitbucket_revamp",
            "type" : "Bitbucket",
            "url" : "",
            "username" : "admin",
            "password" : "DevOps1!",
            "useMidServer" : false
        },
        {
            "name": "gitlab_revamp",
            "type": "GitLab",
            "url": "http://gitlab2.sndevops.xyz",
            "username": "admin",
            "password": "mYdAJQCLi6Qft4Nk3XvS",
            "useMidServer": false
        }],
        "credentials" : {
            "name" : "devops.integration.user",
            "password" : "devops"
        }
    }
}

```

- Onboard DevOps apps at scale by using the DevOps app onboarding API. The POST/devops/onboarding/app request creates an onboarding app event that is asynchronously processed by the DevOps service. For more information, see [POST/devops/onboarding/app](#).
- Onboard DevOps tools at scale by using the DevOps tool onboarding API. The POST/devops/onboarding/tool request creates an onboarding tool event that is asynchronously processed by the DevOps service. For more information, see [POST/devops/onboarding/tool](#).

### Importing historical data for DevOps tools

Use the service catalog to onboard a new app and import historical DevOps data for that app. Enable polling to import data that is mapped to associated plans, repositories and pipelines on a scheduled frequency.

### Import historical DevOps data for existing tools

You can create an app onboarding request and import historical data for a DevOps tool that you have already on-boarded, using the app onboarding catalog form. Currently, you can import historical data for the last 90 days from current date, and enable polling on a scheduled frequency for the following tools:

- Jira (Plan)
- GitHub and GitHub Enterprise (Coding)
- Jenkins (Orchestration)

#### ***i*** Note:

- Ensure that you have created, connected, and discovered the tool you are importing data for.
- Import requests for Plan tool (Jira) are processed first, followed by repository and pipeline import requests.

## Import workflow and retries

On successfully submitting the catalog request from the Self-service catalog, the request is sent for approval following the designated approver flow that you have configured. When the request is approved, an inbound event is created for the app onboarding request. The **Processing details** field of the inbound event record displays the import request ID and status. A single import request creates multiple child import request pages that display in the related list. The import request pages are created based on the following logic for the supported tools:

- Jira: a page is created for a range of 15 days.
- GitHub : a page is created for every 100 commits.
- Jenkins: pages are created per build.

. After import requests are complete processing, the associated Work items, Commits, Branches, Tags, Pipeline executions, Test summaries that you mapped are created and persisted in the system.

On successful import you can view all imported commits in the **DevOps > Develop**:

- Branches
- Commits
- Committers
- Tags
- Repositories
- Work Items

for details to confirm successful import for the specified date range.

While processing an import request, if any page errors out, an inbuilt retry mechanism tries processing the page for a set number of times. After all automatic retries, if the page is still in error state, the subsequent or remaining pages in the import request are processed. The overall state of the import request remains in error.

For example, if the Plan import request failed (after all retries), we will proceed to process Repository and Pipeline imports. You can configure the retries for the import request from **DevOps > Administration > Properties > Maximum retries per page, while importing**.

- Specify the count of retries to auto-attempt, in case the import request page fails in the **Maximum retries per page, while importing** field. If after all the automatic retries, page does not succeed, the import request will process the remaining pages. The overall status of the import request reflects as errored.
- You can manually retry attempting a failed import by clicking the **Retry import** button, on the failed import request page,

### Related topics

[Import existing Azure DevOps pipelines, repositories, and plans](#)

### Polling schedule and Configuration

Enable polling to import DevOps data on a scheduled frequency to apps that have imported historical data, and are mapped to associated plans, repositories, and pipelines.

After you have onboarded an app and imported the associated DevOps data, you can enable the base system schedule for import requests to be created for the plans, repositories,

and pipelines that are tracked and associated to an app. When the import requests complete processing, the associated data is persisted and display against the app. While the base system *DevOpsImportPolling* schedule job is active by default, you must enable polling from the DevOps properties to run the scheduled job.

To enable polling, navigate to **DevOps > Administration > Properties > Enable Import Polling** and select the check box.

Turning this property flag on, enables the base system *DevOpsImportPolling* schedule job. The scheduled job for polling considers either the last successful import or 30 days, whichever is later as the 'start date', and the current day's date as the 'end date' for the data import, for all apps that are active and have tracked pipelines. The job looks up the time of the last successful import and creates the subsequent import request accordingly. This logic ensures that the scheduled polling job imports the delta of relevant DevOps data for that app, from the last successful import till date, to a maximum of thirty days.

**Note:** Do not configure a polling frequency that is less than a day or 24 hours.

The default frequency of the job is set to run daily at midnight using the system time zone. To change the frequency of scheduled job you need the ServiceNow Now Platform Administrator (admin) role.

Navigate to **System Definition > Scheduled Jobs > DevOpsImportPolling** and modify the **Run** frequency, **Time zone**, and **Time** field values, as needed. For more information, see [Schedule Jobs](#).

**Note:**

- The scheduled job only applies to active apps. Ensure that the app you are configuring polling for is in active state and the **Track** field is enabled for the relevant pipelines.
- Consider the following when you're modifying the schedule frequency:
  - For JIRA the default time zone is based on the time zone of the JIRA server's location.
  - For Jenkins the default time zone is UTC. For more information see [Jenkins documentation on System Time time zones](#).

The schedule jobs that polls to import DevOps data honours the default values for the following DevOps properties related to imports and import requests:

- **Maximum retries per page, while importing**
- **Maximum number of pages to process at a time for an import request**
- **To save payloads as attachments on the Import Request Page record, set the value to "true". Anything else is considered false**

### Import existing Azure DevOps pipelines, repositories, and plans

After you have integrated Azure DevOps with DevOps, you can import up to 90 days of existing Azure DevOps pipeline, repository, and plan data. You can then use DevOps dashboards to view and manage Azure DevOps data.

### Before you begin

Role required: admin

## About this task

- You will request the data from the service catalog as a predefined catalog item.
- Imported test summaries, artifacts, and packages are linked to pipeline executions and not to step executions.
- SonarQube scan results are not imported.
- Azure DevOps imposes the following restrictions:
  - Maximum 20,000 work items can be imported every 15 days.
  - Maximum 200 run commits can be mapped to any pipeline execution.
  - Test results for pipeline executions longer than 7 days are not returned.

**Note:** The import process can take some time, hours for very large data sets.

## Procedure

1. Navigate to **All > Service Catalog > Catalog Definitions > My Catalogs** and then select **DevOps Onboarding**.
2. In the **Catalog Items** related list, select **DevOps App Onboarding**.
3. On the Catalog Item form, select **Try It** to request the data.  
The resulting DevOps App Onboarding form enables you to specify the catalog item to order. In this case, the "app" to order is your Azure DevOps instance.
4. Select the Select in list icon () on the **App** field and then select your instance of Azure DevOps.  
Now that you have specified the instance, you will specify date range and sources of data to import.
5. Repeat the following procedure for each pipeline, repository, and plan that you want to import:
  - a. Select the Select in list icon () on the appropriate **Onboarding** field to select the item to import.  
You can select multiple items.
  - b. Specify the range of dates for the data in the **Import From** and **Import To** fields.
6. Select **Order Now**.  
Your request appears on the Order Status page.
7. Select the request number so you or another user with the admin role can approve the request.
8. Approve the request: On the Request form, set **Approval** and **Request state** to **Approved**.  
The import process begins immediately on approval.

## Import existing GitLab pipelines and repositories

After you have integrated GitLab with DevOps, you can import up to 90 days of existing GitLab pipeline and repository data. You can then use DevOps dashboards to view and manage GitLab data.

## Before you begin

Role required: admin

## About this task

- You will request the data from the service catalog as a predefined catalog item.
- Imported test summaries are linked to pipeline executions and not to step executions.
- Only artifacts published using the artifacts keyword are imported.
- Test results are not displayed for artifacts that have expired. You can set the expiration date of an artifact by configuring the **expire\_in** property in the pipeline. For more information on artifact expiration policies, see [Artifact and job meta data expiration](#).
- SonarQube scan results are not imported.
- Only 6400 commits per branch can be imported in a single import.
- GitLab imposes the following restriction: While associating run commits to a pipeline execution, GitLab doesn't provide the starting part of the commit details in some scenarios. It provides only the part before the SHA as '0000000000000000'. In such scenarios, the latest commit will be associated as the run commit. For example, when a new branch is created or when a pipeline is run manually.

**Note:** The import process can take some time, hours for very large data sets.

## Procedure

1. Navigate to **All > Service Catalog > Catalog Definitions > My Catalogs** and then select **DevOps Onboarding**.
2. In the **Catalog Items** related list, select **DevOps App Onboarding**.
3. On the Catalog Item form, select **Try It** to request the data.  
The resulting DevOps App Onboarding form enables you to specify the catalog item to order. In this case, the "app" to order is your GitLab instance.
4. Select the Select in list icon () on the **App** field and then select your instance of GitLab.  
Now that you have specified the instance, you will specify date range and sources of data to import.
5. Repeat the following procedure for each repository that you want to import:
  - a. Select the Select in list icon () on the **Onboarding Repositories** field and then select the item to import.  
You can select multiple items.
  - b. Specify the range of dates for the data in the **Import From** and **Import To** fields.
6. **Note:** The pipelines mapped to the repositories are automatically selected when you select the repository in the **Onboarding Repositories** field. You do not have to select the pipelines separately.
7. Select **Order Now**.  
Your request appears on the Order Status page.
8. Select the request number so you or another user with the admin role can approve the request.

- Approve the request: On the Request form, set **Approval** and **Request state** to **Approved**. The import process begins immediately upon approval.

## Deleting a record in DevOps Change Velocity

Cascade record deletion is implemented to delete all dependent lower level DevOps records whenever a parent or higher level DevOps entity is deleted. Confirmation popups ensure that you understand that data will be lost when you delete a record (for example, the record of a tool connection).

For example, when a Plan record is deleted, all dependent Work Item, Plan Version, and many-to-many relation (like App to Plan, and Work Item to Plan Version) records are deleted.

DevOps cascade deletion is implemented for these tables.

- Pipeline, Pipeline Execution, Step, Orchestration Task, Task Execution
- Repository, Commit, Branch, Tag
- Plan, Work item, Plan Version
- DevOps Tool, Artifact Repository, Artifact, Test Summary / Performance test summary

## Delete action on a DevOps form

A user with the sn\_devops.admin role can delete a DevOps record, but only if it meets the defined ACL criteria.

**Note:** To see the **Delete** button on a form, you must have the sn\_devops.admin role, and the current record must meet the criteria defined in the scripted ACL.

Entity	Scripted ACL criteria
Pipeline	A Pipeline record can be deleted only if no other pipeline executions from other pipelines are dependent on the artifact versions generated by the pipeline executions of this pipeline.
Pipeline Execution	A Pipeline Execution record can be deleted only if no other pipeline executions are dependent on the artifact versions generated by this pipeline execution.
Task Execution	<p>A Task Execution record can be deleted if ALL of these conditions are met.</p> <ul style="list-style-type: none"> <li>• There are no step executions referencing it.</li> <li>• There are no downstream task executions referencing it.</li> <li>• There are no pipeline executions dependent on the artifact versions built by this task execution.</li> </ul>

Entity	Scripted ACL criteria
Step	A Step record can be deleted only if there are no orchestration tasks or step executions referencing this step.
Orchestration Task	<p>If the orchestration task has a step associated, it can be deleted only if there are no task executions referencing this orchestration task.</p> <p>If the orchestration task does not have a step associated (example Jenkins freestyle job), it can be deleted only if no other pipeline executions are dependent on the artifact versions generated by the task executions of this orchestration task.</p>
Repository	<p>A Repository record can be deleted only if none of the commits of this repository are associated to the artifact versions (Artifact Version to Commit table) or task executions (Run Commit table).</p> <p>Therefore, before cleaning up the repository, delete the dependent pipeline entities.</p>
Branch	A Branch record can be deleted only if there are no commits associated to it in the Branch To Commit table.
Tag	A Tag record cannot be deleted by a sn_devops.admin.
Commit	<p>A Commit record can be deleted if ALL of these conditions are met.</p> <ul style="list-style-type: none"> <li>• The commit is not associated to the artifact version (Artifact Version to Commit table).</li> <li>• The commit is not associated to task executions (Run Commit table).</li> <li>• The commit is not being referenced by other commits as a revert commit.</li> </ul>
Plan	A Plan record can be deleted only if none of the work items of this plan are associated or referenced by any commits.
Work Item	<p>A Work Item record can be deleted if ALL of these conditions are met.</p> <ul style="list-style-type: none"> <li>• The work item is not being referenced by another work item as parent.</li> <li>• There are no commits referencing or associated to this work item.</li> </ul>

Entity	Scripted ACL criteria
Plan Version	A Plan Version record can be deleted only if there are no work items associated to it in the Work Item To Plan Version table.
Test Summary / Performance Test Summary	A Test Summary record can be deleted only if it doesn't have an associated related record (Artifact version/Package/Task Execution) in the Test Summary Relations table.
Artifact	An Artifact Record can be deleted only if all the artifact versions belonging to it are deletable.
Artifact Repository	An Artifact Repository record can be deleted only if all the artifacts belonging to it are deletable.
Artifact Version	An Artifact Version record can be deleted only if its built by task execution field is empty.
Artifact Staged Request	An Artifact Staged Request record can be deleted when it is either an orphaned record, or the state is Processed / Error.
Package	<p>A Package record can be deleted if ALL of these conditions are met.</p> <ul style="list-style-type: none"> <li>• There are no pipeline executions referenced to it.</li> <li>• The built by task execution value on the Package record is null.</li> </ul>
DevOps Tool	A DevOps Tool record cannot be deleted by a sn_devops.admin.
Build Test Summary Build Test Result Commit Details Event Inbound Event	These entities cannot be deleted by a sn_devops.admin.

## DevOps record delete cascade

Deleting a record in a parent table cascade deletes all the child records in the hierarchy.

Parent record being deleted	Cascade deleted child records
Pipeline	<p>Step: Orchestration Task</p> <p>Pipeline Execution:</p>

Parent record being deleted	Cascade deleted child records
	<ul style="list-style-type: none"> <li>• Step Execution</li> <li>• Callback</li> <li>• Task Execution <ul style="list-style-type: none"> <li>◦ Package</li> <li>◦ Run Commit</li> <li>◦ Test Summary Relations</li> <li>◦ Build Test Summary: Build Test Result</li> <li>◦ Artifact Version <ul style="list-style-type: none"> <li>▪ Artifact Staged Request</li> <li>▪ Artifact Version To Commit</li> <li>▪ Artifact Version To Package</li> </ul> </li> </ul> </li> </ul>
Pipeline Execution	<p>Step Execution</p> <p>Callback</p> <p>Task Execution:</p> <ul style="list-style-type: none"> <li>• Package</li> <li>• Run Commit</li> <li>• Test Summary Relations</li> <li>• Build Test Summary: Build Test Result</li> <li>• Artifact Version <ul style="list-style-type: none"> <li>◦ Artifact Staged Request</li> <li>◦ Artifact Version To Commit</li> <li>◦ Artifact Version To Package</li> </ul> </li> </ul>
Step	<p>None.</p> <p>A Step record can be deleted only if there are no Orchestration Task or Step Execution records associated to it.</p>
Orchestration Task	<p>Task Execution</p> <ul style="list-style-type: none"> <li>• Package</li> <li>• Run Commit</li> <li>• Test Summary Relations</li> <li>• Build Test Summary: Build Test Result</li> <li>• Artifact Version <ul style="list-style-type: none"> <li>◦ Artifact Staged Request</li> <li>◦ Artifact Version to Commit</li> <li>◦ Artifact Version to Package</li> </ul> </li> </ul>

Parent record being deleted	Cascade deleted child records
Repository	Tag Branch: Branch to Commit Commit: <ul style="list-style-type: none"> <li>• Commit Details</li> <li>• Branch to Commit</li> <li>• Tag</li> </ul>
Branch	Branch to Commit
Tag	No dependent child records.
Commit	Commit Details Branch to Commit Tag
Plan	Work Item App to Plan Plan Version: Work Item to Plan Version
Work Item	Work Item to Plan Version
Plan Version	Work Item to Plan Version
Test Summary / Performance Test Summary	Test Summary Relations
Artifact	Artifact Version <ul style="list-style-type: none"> <li>• Artifact Staged Request</li> <li>• Artifact Version to Commit</li> <li>• Artifact Version to Package</li> </ul>
Artifact Repository	Artifact Artifact Version: <ul style="list-style-type: none"> <li>• Artifact Staged Request</li> <li>• Artifact Version to Commit</li> <li>• Artifact Version to Package</li> </ul>
DevOps Tool	Event Inbound Event Test Summary / Performance Test Summary: Test Summary Relations Plan:

Parent record being deleted	Cascade deleted child records
	<ul style="list-style-type: none"> <li>• Work Item</li> <li>• App to Plan</li> <li>• Plan Version: Work Item to Plan Version</li>   <li>Artifact Repository</li>   <li>Artifact</li>   <li>Artifact Version:           <ul style="list-style-type: none"> <li>• Artifact Staged Request</li> <li>• Artifact Version to Commit</li> <li>• Artifact Version to Package</li> </ul> </li>   <li>Repository:           <ul style="list-style-type: none"> <li>• Tag</li> <li>• Branch: Branch to Commit</li>   <li>• Commit               <ul style="list-style-type: none"> <li>◦ Commit Details</li> <li>◦ Branch to Commit</li> <li>◦ Tag</li> </ul> </li> </ul> </li>   <li>Pipeline           <ul style="list-style-type: none"> <li>• Step: Orchestration Task</li>   <li>• Pipeline Execution               <ul style="list-style-type: none"> <li>◦ Step Execution</li> <li>◦ Callback</li> <li>◦ Task Execution                   <ul style="list-style-type: none"> <li>▪ Package</li> <li>▪ Run Commit</li> <li>▪ Test Summary Relations</li> <li>▪ Build Test Summary: Build Test Result</li> <li>▪ Artifact Version                       <ul style="list-style-type: none"> <li>▪ Artifact Staged Request</li> <li>▪ Artifact Version To Commit</li> <li>▪ Artifact Version To Package</li> </ul> </li> </ul> </li> </ul> </li> </ul> </li> </ul>

## DevOps record delete cascade exceptions

These records are always deleted in the foreground.

Parent record being deleted	Cascade deleted child records
Artifact Version	<p>These records are deleted along with Artifact Version.</p> <ul style="list-style-type: none"> <li>• Artifact Staged Request</li> <li>• Artifact Version To Commit</li> <li>• Artifact Version to Package</li> </ul>
Build Test Summary	<p>These records are deleted along with Build Test Summary.</p>
Package	<p>These records are deleted along with Package.</p> <ul style="list-style-type: none"> <li>• Artifact Staged Request</li> <li>• Artifact Version to Package</li> </ul>

## Foreground deletion threshold property

Deletion of DevOps records occurs in the foreground (synchronously) by default. Meaning, other UI actions cannot be performed during synchronous deletion. Because the deletion of a parent record can result in the deletion of multiple child records, you can set a threshold value so the remaining records are deleted in the background.

**Note:** Artifact Version, Build Test Summary, and Package records are always deleted in the foreground.

To set the DevOps [Cascade delete threshold](#) property, navigate to **DevOps > Administration > Properties**. Define the total number of records that can be cascade deleted in the foreground, after which the remaining records are deleted in the background. Default is 1000.

**Note:** The records of tables in a hierarchy are deleted in a bottom-up manner. For example, it might be possible that even after triggering the delete action on a Repository record, it will still be available for read, write, and update in the system until deletion is complete.

DevOps record deletions do not trigger any business rules or workflows.

## DevOps record deletion UI

Cascade delete of a DevOps record triggers multiple confirmation approvals.

## Software Quality Results

Software Quality Results display scan details from SonarQube scans configured on your GitHub Actions, Jenkins, or Azure DevOps pipelines.

## Software Quality Results

You can configure SonarQube scan results from your GitHub Actions, Jenkins and Azure DevOps pipelines to be fetched into ServiceNow DevOps. SonarQube notifications create inbound events which are then processed by base system subflows associated with all inbound events with Software Quality capability.

After you have configured SonarQube scans on your pipelines and configured the corresponding plugins, run the pipelines to fetch software quality scan results into ServiceNow DevOps. You can view the scan results by Scan ID for each SonarQube scan that was part of your build or release pipeline execution steps.

### Software Quality Summaries

Software Quality Summaries shows you a summary of the scans that are run on the pipelines with SonarQube scans on the DevOps app on your Now Platform instance.

### Software Quality Summaries

1. Navigate to **Software Quality Results > Software Quality Summaries**.
2. Click a **Scan ID** record to view scan details.

The following Software Quality Scan Summary Details appear for the Scan ID.

#### Software Quality Scan Summary results

Field	Value
Number	The Scan record number in the CMDB.
Initiated By	The user who initiated the scan
Scan ID	The unique Scan ID
Project Name	The SonarQube project name or key.
Last Scanned	Date Time stamp of the last scan
Scan URL	The SonarQube project URL you configured in the extension task.
Scanner name	The code quality scanner tool name.
Domain	The domain name in which the scans are configured in the instance.
Tool	The name of the SonarQube tool you created.

**i Note:** The scanID, lastScannedBy, and initiatedBy details are not retrieved when you onboard Sonar using a non-admin user.

The Software Quality Scan Details related list (for each Scan ID) displays for the following categories with a corresponding value signifying the count value against the scan result category.

Software Quality categories for Overall Code metrics:

- Coverage
- Bugs

- Reliability
- Rating
- Code
- Smells
- Duplications
- Security Rating
- Lines of Code
- Vulnerabilities
- Security hotspots
- Maintainability rating

Software Quality categories for New Code metrics:

- Duplications
- New Duplications
- Vulnerabilities
- New Vulnerabilities
- Technical Debt
- New Technical Debt
- Maintainability rating
- New Maintainability rating
- Bugs
- New Bugs
- Lines to Cover
- New Lines to Cover
- Lines of Code
- New Lines of Code
- Coverage
- New Coverage
- Reliability Rating
- New Reliability Rating
- Code Smells
- New Code Smells
- Security Rating
- New Security Rating
- Security Review
- New Security Review
- Security Hotspots
- New Security Hotspots

## Software Quality subcategories

The Software Quality Scan Details related list (for each Scan ID) displays software quality subcategories that you can configure for results of the **Vulnerabilities** and **New Vulnerabilities** categories from the scan results.

## Software Quality subcategories- Vulnerabilities and New Vulnerabilities

Use seeded subcategories or create new categories to prioritize and list your scan results of category type- Vulnerabilities and New Vulnerabilities.

### 1. Navigate to **DevOps > Integrations > Software Quality Sub Categories**.

The Software Quality Sub Categories form displays with the following base-system subcategories:

- Blocker
- Critical
- Major
- Minor
- Info

### 2. Click the **New** button to create custom sub categories.

### 3. On the form, fill in the fields.

#### Software Quality Sub Category

Field	Description
Label	A unique label for the sub category.
Domain	The domain in which the DevOps app is running.
Name	A unique name for the sub category.
Category	Select a category from the seeded base-system categories from the lookup list.

### 4. Click **Submit**.

You have successfully created a custom sub category.

To view the Software Quality Scan Detail and the Software Quality Category Details related list > Sub category, follow these steps:

- View scan details as part of Task Executions. View details of all the Sonar scans that are part of the task execution mapped to a build or release pipeline execution step.

### 1. Navigate to **DevOps > Orchestrate > Task Execution** click a relevant Task Execution record.

### 2. Click the Software Quality Summary related list.

### 3. Click a relevant Scan ID record.

The Software Quality Scan Summary and Scan Details are displayed.

- View scan details as part of Change Request. View all the scans that were part of this build/release pipeline in the **Software Quality Results > Software Quality Summary** related list.

**1. Navigate to DevOps > Orchestrate > Pipeline Change Requests**

**2. Click the Software Quality Summary related list.**

**3. Click a relevant Scan ID record.**

The Software Quality Scan Summary and Scan Details are displayed.

## Check permissions and update credentials for tools — Workspace

You can perform permission checks and update credentials like passwords and access tokens for your tools from the tool details page.

### Permission checks

You can perform permission checks on connected tools at any time, to verify if the existing credentials have the necessary permissions. This check helps to determine if there are possible impacts or if you must get a token or credential with higher-level permissions for the tool.

#### Note:

SonarQube and Rally doesn't have the option to check for permissions.

1. From the DevOps Change Workspace, navigate to Tools and select the tool to open the details page.

The **Permission check result** field shows whether the tool credentials have all the necessary permissions, have partial permissions, or if permissions haven't been checked. The **Last Permission check** field tells you when the permission checks were previously run.

2. Click **More Actions**.

- For Bitbucket, select **Check password permissions**.
- For GitHub and GitHub Enterprise with OAuth credentials using the GitHub app, select **Check credential permissions** and then enter the **GitHub app slug name**. Click **Check permissions**.
- For others, select **Check credential permissions**.

You can see the status of the checks depending on your credential permissions. You need sufficient permissions on your credentials for seamless discovery and import.

3. If permissions aren't sufficient, it's recommended to update the credentials with those having higher-level permissions, or update the permissions for the objects on the external tool.

### Update credentials

You can update the tool credentials with credentials having sufficient permissions for seamless discovery and import of data from your tool.

1. From the DevOps Change Workspace, navigate to Tools and select the tool to open the details page.
2. Click **More Actions**. Depending on your tool, the options to update your credentials are displayed.

Tool	Steps
Bitbucket	<ol style="list-style-type: none"> <li>a. Click <b>Update password</b>.</li> <li>b. Enter the user name and the new password.</li> <li>c. Click <b>Check permissions</b>.</li> <li>d. The permission check results are shown in the Permission check dialog box. If you are satisfied with the permissions for your tool, then update the credentials.</li> </ol>
Azure DevOps	<ol style="list-style-type: none"> <li>a. Click <b>Update credentials</b>.</li> <li>b. Enter the new password or access token.</li> <li>c. Click <b>Check permissions</b>.</li> <li>d. The permission check results are shown in the Permission check dialog box. If you're satisfied with the permissions for your tool, then update the credentials.</li> </ol> <p><b>i Note:</b> Since the DevOps tool maps to an Azure DevOps organization, the <b>Project Administrators</b> privilege requires the owner of the PAT to be a member of the organization's <b>Project Collection Administrators</b> group.</p>
Jira, Jenkins, JFrog	<ol style="list-style-type: none"> <li>a. Click <b>Update credentials</b>.</li> <li>b. Enter the user name and new password or access token.</li> <li>c. Click <b>Check permissions</b>.</li> <li>d. The permission check results are shown in the Permission check dialog box. If you're satisfied with the permissions for your tool, then update the credentials.</li> </ol>
GitLab	<ol style="list-style-type: none"> <li>a. Click <b>Update credentials</b>.</li> <li>b. Select the <b>Credential type</b>. <ul style="list-style-type: none"> <li>▪ If the credential type is <b>Basic Auth</b>, enter the new password or access token.</li> <li>▪ If the credential type is <b>OAuth</b>, enter the new credential.</li> </ul> </li> <li>c. Click <b>Check permissions</b>.</li> </ol>

Tool	Steps
GitHub, GitHub Enterprise	<p>d. The permission check results are shown in the Permission check dialog box. If you're satisfied with the permissions for your tool, then update the credentials.</p> <p>a. Click <b>Update credentials</b>.</p> <p>b. Select the <b>Credential type</b>.</p> <ul style="list-style-type: none"> <li>▪ If the credential type is <b>Basic Auth</b>, enter the user name and new password or access token.</li> <li>▪ If the credential type is <b>OAuth</b> using the GitHub app, enter the new credential. If you don't want to check for permissions, then click <b>Update</b> directly, and the credentials are updated.</li> </ul> <p>To check the permissions, you must enter the <b>GitHub app slug name</b>.</p> <ul style="list-style-type: none"> <li>▪ If the credential type is OAuth using the OAuth app (not the GitHub app) at GitHub end, enter the new credential.</li> </ul> <p>c. Click <b>Check permissions</b>.</p> <p>d. The permission check results are shown in the Permission check dialog box. If you're satisfied with the permissions for your tool, then update the credentials.</p>
SonarQube, Rally	<p>These tools don't check for permissions. To update credentials:</p> <p>a. Click <b>Update credentials</b>.</p> <p>b. Enter the user name and new password or access token.</p> <p>c. Click <b>Update</b> to update the credentials.</p>

Permission checks are run on the new credentials. Once permissions check is completed, you can proceed with updating the credentials. If you want to abort the update, click **Cancel**.

## Inbound event table data archiving and cleanup

Use table cleaners on the inbound events table to purge event data records beyond a specified period. Configure a retention policy with table rotation on the auto-flush form to manage table size growth and archiving data beyond the specified duration.

### Overview

The data management feature allows you to enable archiving and purging of inbound event data. Database rotation and table cleanup configurations are enabled to ensure that eight weeks of event data is retained. Instance performance is preserved using Now Platform

features such as Database rotation, [table rotation](#), and [table cleanup](#). A scheduled job is auto-enabled two weeks after you upgrade.

## Table rotation on processed inbound events table

The DevOps processed inbound events [sn\_devops\_processed\_inbound\_event\_list.do] table contains copies of event data from the DevOps inbound events [sn\_devops\_inbound\_list.do] table, that is in state '*processed*' and '*ignored*'.

The processed inbound events table has the table rotation feature enabled. In all, there are eight preconfigured shards, each containing a week's event data from the DevOps inbound event table. By default, once you have 56 days (7 days (Duration) \*8 shards (Rotations) of data in the processed inbound table, table rotation is activated. For more information on applying table rotation, see [Apply table rotation](#)

**Tip:** To modify table rotation configuration, navigate to **System Definitions > Table Rotation** and open the record for DevOps processed inbound events table.

## Scheduled jobs and Table cleaners

**Important:** A scheduled job is auto-enabled two weeks after you upgrade . The scheduled job will run every seven days and is deactivated after the last update.

Scheduled jobs activate tables cleaners on the inbound events if either of the following scenarios are met:

- A week after you upgrade, and the processed inbound table has records older than seven weeks.
- Two weeks after you upgrade.

The table cleanup feature for the inbound events table [sn\_devops\_inbound\_table] has two cleaners that are part of the base system upgrade.

**1.** Table cleaners are activated and run on the inbound events [sn\_devops\_inbound\_table\_list.do] for events in the following states.

- *Processed*
- *Ignored*

based on the value specified in the **Age in seconds** field against the value in sys\_created\_on field.

**2.** Table cleaners are activated and run on the inbound events table [sn\_devops\_inbound\_table\_list] for events in the following states.

- *New*
- *Retry*
- *In-progress*

based on the value specified in the **Age in seconds** field against the value in sys\_created\_on field.

**Note:** By default, table cleaners are run on the inbound events table for these states from the time the schedules jobs are activated. The default value in the **Age in Seconds** field is 4,838,400 seconds (56\*24\*60\*60).

## Auto-updates to table cleaner frequency

Events in states *processed* and *ignored* are backed up and copied to the processed inbound events [sn\_devops\_processed\_inbound\_event\_list] table. By default, the process cleaners are auto-updated to an interval of 30 minutes or 1800 seconds, on the inbound events table, if the following conditions are met:

- Two weeks have passed since the scheduled job is active.
- The processed inbound event table has event data older than 7 weeks.

## Data archiving rules for DevOps tables

Base system table archiving rules ensure that DevOps data stored in the Configuration Management Database (CMDB) are systematically archived and purged.

### Archiving rules for DevOps data in CMDB tables

Data archiving involves managing table size growth and archiving old data. It moves data that is no longer needed every day from primary tables to a set of archive tables. For more information, see [Data archiving](#). Base-system archive rules are configured to auto-archive DevOps tables that are older than a specified period. An archive table is created for any table that has an archive rule associated with it. You can also choose to restore data from archive tables.

**Note:** While you can restore any record from the archive tables. Once archived data is restored, the same data is no longer auto-archived. For more information, see [Manage archived data](#).

- Navigate to **System Archiving > Archive Rules** and select the individual archive rule whose data you restored.
- Enable the **Auto rearchive** check box, to resume auto-archiving for that archive rule.

### Modify base system value for archive rules

You can configure the auto archive duration for all the archive rules that are applicable to DevOps tables from the **Auto archive (in months)** DevOps system property. By default, this property's value is set to 9 (months). Navigate to **DevOps > Administration > Properties > Auto archive (in months)**, to modify the value. For more information on configuring data archiving, see [Create an archive rule](#)

The following list indicates the tables that are auto-archived.

#### Auto-archived DevOps tables

Table	Table Name	Archived table name
Artifact Staged Request	sn_devops_artifact_staging	ar_sn_devops_artifact_staging
Artifact Version	sn_devops_artifact_version	ar_sn_devops_artifact_version
Branch	sn_devops_branch	ar_sn_devops_branch
Build Test Result	sn_devops_build_test_result	ar_sn_devops_build_test_result
Build Test Summary	sn_devops_build_test_summary	ar_sn_devops_build_test_summary
Callback	sn_devops_callback	ar_sn_devops_callback
Commit	sn_devops_commit	ar_sn_devops_commit

## Auto-archived DevOps tables (continued)

Table	Table Name	Archived table name
Commit Details	sn_devops_commit_details	ar_sn_devops_commit_details
Deployed Artifact to TaskExecution	sn_devops_m2m_artifact_execution	ar_sn_devops_m2m_artifact_execution
Artifact Version to Commit	sn_devops_m2m_artifact_version	ar_sn_devops_m2m_artifact_version
Branch To Commit	sn_devops_m2m_branch_commit	ar_sn_devops_m2m_branch_commit
Run Commit	sn_devops_m2m_commit_execution	ar_sn_devops_m2m_commit_execution
Commit To Work Item	sn_devops_m2m_commit_workitem	ar_sn_devops_m2m_commit_work_item
Pipeline Execution	sn_devops_pipeline_execution	ar_sn_devops_pipeline_execution
Software Quality Scan Summary	sn_devops_software_quality_scan_summary	ar_sn_devops_software_quality_scan_summary
Software Quality Scan Summary Relations	sn_devops_software_quality_scan_summary_relations	ar_sn_devops_software_quality_scan_summary_relations
Step Execution	sn_devops_step_execution	ar_sn_devops_step_execution
Tag	sn_devops_tag	ar_sn_devops_tag
Task Execution	sn_devops_task_execution	ar_sn_devops_task_execution
Task Summary	sn_devops_test_summary	ar_sn_devops_test_summary
Test Summary Relations	sn_devops_test_summary_relations	ar_sn_devops_test_summary_relations
Work Item	sn_devops_work_item	ar_sn_devops_work_item

The data archiving rules feature considers and honors parent rules before individual archive rules are run. If you have an associated parent archive rule, the child rules are run only when the parent rules are run. Similarly, when you modify individual rules, you must first disable or disassociate the parent archive rule, make the configuration changes, and re-enable the parent archive rule. Changes to the **Auto archive (in months)** DevOps property overrides any configuration changes made to the duration of child archive rules. For example, you could update the archive rule condition for any of the **Active** DevOps archive rules to a custom value, say three months instead of the default 9 months. However, if you modify the **Auto archive (in months)** DevOps property, all modifications made to the conditions of individual archive rules are overwritten by the value in the **Auto archive (in months)** system property.

**i Note:** The override function is binding for all scenarios except:

- When an archive rule does not belong to the DevOps scope.
- When an archive rule belongs to the DevOps scope but is inactive.

## Base system destroy rules for DevOps data

Base-system destroy rules are also enabled and activated on all the archived DevOps tables. By default, data is deleted from an archived table after 36 months or 1095 days have passed from the time the data is stored in the archive table. For more information, see [Create a destroy rule](#).

## DevOps log levels

DevOps log levels allow you to filter logs by level as well by a particular tool or application. Log levels in the DevOps application let you decide the extent of log detail you need for debugging.

The flexibility to set log levels allows you to select the right log level information that is appropriate for your organizational needs. You can select custom log levels based on the log levels or categories that you need and as an application or tool currently supports. In previous versions of DevOps, the default log level was set to debug, and the [sn\_devops.enable\_debug property that you could either enable or disable. You can select any of the following log levels for the DevOps application using the DevOps sn\_devops.devops\_log\_level property:

- Error
- Warning
- Information
- Debug
- Trace

***i* Note:**

- The base system log level is set to Warning. You must modify the base system value from DevOps properties.
- By default, if you have different log levels for connected apps and for the DevOps application, the log level with more information is printed. For example, if you set up the log level for an app at Error, and for the DevOps application at Trace, the log levels are printed with the most information (Trace).

Multiple flows for in the DevOps application now use the new logging framework to collect log data during the flow execution. The following flows currently support the new log levels:

- Artifact & Package registration flows
- Self Service Onboarding
- Self-Service Catalog — App Onboarding
- Change Traceability flow
- Notification flow for plan, code, and orchestration capabilities

## DevOps troubleshooting

Troubleshooting actions can help resolve common issues when setting up or running the DevOps application.

Issue	Action
Import request not progressing	<p>If an import request remains in the Requested state for too long while performing an import for a tool (such as Jenkins, Jira, or GitHub ), delete the import request and try again.</p> <p><b><i>i</i> Note:</b> Delete the existing request to retry importing the same range.</p>

Issue	Action
Tool connection fails	Remove the trailing slash ('/') in the <b>Connection URL</b> field on the HTTP Connection form.
No change request is created for a Jenkins job under change control	<p>Verify that:</p> <ul style="list-style-type: none"> <li>The tool integration in your instance is set up properly.</li> <li>The task has been synced in your instance.</li> <li>Tasks and app steps have been configured in your instance.</li> </ul> <p>Change request creation is not supported if the task is under change control:</p> <ul style="list-style-type: none"> <li>Is not part of a pipeline (is a standalone task, for example).</li> <li>Is the first in the pipeline.</li> <li>Is within the pipeline, but the user manually triggers, or does SCM checkout directly on the task under change control (thus not triggering the pipeline from the beginning).</li> </ul>
Jenkins does not block the job under change control (does not wait for change request approval)	<p>Verify that the Jenkins location is configured:</p> <p>Navigate to <b>Jenkins &gt; Manage Jenkins &gt; Configure System</b> and provide the hostname for the <b>Jenkins URL</b> field in the Jenkins Location section.</p> <p><b>Note:</b> To avoid caching issues, click <b>Save</b> even if the <b>Jenkins URL</b> field already contains a value when you first open the form.</p>
Events occurring in the payload log with state Not Connected	<p>If any of the following changes for a connection made manually (using manual configuration mode), the connection is automatically disconnected.</p> <ul style="list-style-type: none"> <li>Alias associated with the tool</li> <li>Type of tool</li> <li>New active HTTP connection for the same domain added to the alias</li> <li>Existing HTTP connection for the same domain activated</li> <li>Connection URL of the HTTP connection</li> </ul>

Issue	Action
	<ul style="list-style-type: none"> <li>Credentials of the HTTP connection</li> <li>Use MID Server setting in the HTTP connection</li> </ul> <p>Enter manual configuration mode and reconnect.</p>
Retry Inbound events that fail or error out due to REST API TimeoutException/FlowObjectAPIException	<p>Update the <i>Retry Errored Inbound Events</i> scheduled job to retry processing inbound events that are in <i>Error</i> state.</p> <ul style="list-style-type: none"> <li>Update the errors or exceptions list to specify exceptions that you want to retry event processing for.</li> <li>Modify the default <i>Maximum Retry</i> count.</li> </ul> <p>For more information, see <a href="#">Retry errored inbound events</a></p>
Pipeline execution in ServiceNow DevOps does not move forward and waits indefinitely as the SonarQube scans do not take place due to the absence of SonarQube tool.  The software quality inbound event displays the following error message in the processing details field. "Check if the respective SonarQube tool is created successfully. If not, create the SonarQube tool and retry the inbound event."	<p>For all SonarQube steps in code quality scans, the user must create SonarQube tool in the ServiceNow DevOps instance.</p> <p>For more information, see <a href="#">SonarQube integration with DevOps Change Velocity</a></p>
Pipeline UI displays broken links between stages .	Navigate to <b>Task Executions</b> and ensure that the Upstream executions column has the appropriate upstream link references.

### Troubleshooting errors in DevOps Change Velocity

Identify the root cause of errors that occur in DevOps Change Velocity, and the corresponding steps required to resolve them.

### Tool connection

This section lists the tool connection errors and their troubleshooting steps when you create a DevOps tool using a catalog, record producer, or workspace playbook.

#### Supported tools

- Azure DevOps
- Jenkins
- Jira
- Bitbucket
- GitHub

- GitHub enterprise
- GitLab
- JFrog
- SonarQube
- Rally

### Troubleshooting instructions

The errors that might occur when you select the **Submit** or **Connect** button after entering the tool details in the tool creation process, and their troubleshooting steps are listed in the following table:

Message	Troubleshooting steps
Tool cannot be created because the tool name has not been entered. Enter the tool name, and try again.	Re-enter the tool name.
Tool cannot be created because the tool integration has not been selected. Select the correct tool integration value, and try again.	Select the correct tool Integration value.
Tool cannot be created because the tool URL is invalid or incorrect. Re-enter the tool URL, and try again.	Re-enter the tool URL.
Tool cannot be created because the platform version cannot be determined. Create the <code>glide.buildtag.last sys</code> property, and try again.	<ol style="list-style-type: none"> <li>1. Navigate to <b>DevOps &gt; Administration &gt; Properties</b>.</li> <li>2. Verify if the <code>glide.buildtag.last sys</code> property exists.</li> <li>3. If not, create the <code>glide.buildtag.last sys</code> property.</li> </ol>
Tool cannot be created because the associated CreateDevOps tool connection is invalid. Create a valid connection alias, and try again.	<ol style="list-style-type: none"> <li>1. Navigate to <b>All &gt; Connections &amp; Credentials &gt; Connection &amp; Credential Aliases</b> and open the <b>CreateDevOpsTool</b> record.</li> <li>2. In the Connections related list, create a record and enter a name for the connection.</li> <li>3. On the Connection form, select the <b>Credential</b> field lookup list, and then select <b>New</b> to create an admin credential.</li> <li>4. Select <b>Basic Auth Credentials</b> and enter a name.</li> <li>5. Enter admin username and password (required to access the tools in your DevOps environment).</li> </ol>

Message	Troubleshooting steps
	<p>A user with connection_admin role can configure an HTTP connection.</p> <p><b>6.</b> On the Connection form, enter <b>https://&lt;instance name&gt;.service-now.com/</b> for the Connection URL.</p>
<p>Tool cannot be created because the connection URL of the CreateDevOpsTool Alias is incorrect. Use the following URL: <a href="https://&lt;instancename&gt;.service-now.com">https://&lt;instancename&gt;.service-now.com</a>, and try again.</p>	<p><b>1.</b> Navigate to <b>All &gt; Connections &amp; Credentials &gt; Connection &amp; Credential Aliases</b> and open the <b>CreateDevOpsTool</b> record.</p> <p><b>2.</b> On the Connection form, enter <b>https://&lt;instance name&gt;.service-now.com/</b> for the Connection URL.</p>
<p>Tool cannot be created because the &lt;toolname&gt; tool name already exists. Enter a different name, and try again.</p>	<p>Use a different name for the tool.</p>
<p>Tool cannot be created because the &lt;toolname&gt; does not have a valid MID server configuration. Configure a valid MID server, and try again.</p>	<p><b>1.</b> If you are using the workspace, check if the mid server is running and reachable.</p> <p><b>2.</b> If you are using the catalog and record producer, check if the mid server is running and reachable with application as DevOps and Capability as REST.</p>
<p>Tool cannot be created because there are no connection and credentials aliases available for DevOps Data Model scope. Create a new connection and credential alias, and try again.</p>	<p>This error occurs for OAuth authentication.</p> <p><b>1.</b> Navigate to <b>All &gt; Connections &amp; Credentials &gt; Connection &amp; Credential Aliases</b>.</p> <p><b>2.</b> Create a new alias with application as <b>DevOps Data Model</b>.</p>
<p>Tool cannot be created due to a technical issue while creating the credential record.</p>	<p><b>1.</b> Navigate to <b>All &gt; Process Automation &gt; Flow Designer &gt; Executions</b>.</p> <p><b>2.</b> Check for the latest execution of <code>sn_devops.devops_create_credentials</code> subflow to know the details of the error.</p>
<p>Tool cannot be created because credential and domain combination for an active record already exists.</p>	<p>This error occurs for OAuth authentication.</p>

Message	Troubleshooting steps
	<ol style="list-style-type: none"> <li>1. Use a different credential.</li> <li>2. Check if the existing credential is used by any active connection.</li> </ol>
<p>Tool cannot be created due to a technical issue while creating the connection record.</p>	<ol style="list-style-type: none"> <li>1. Navigate to <b>All &gt; Process Automation &gt; Flow Designer &gt; Executions</b>.</li> <li>2. Check for the latest execution of <i>sn_devops.create_connection_for_tool</i> action to know the details of the error.</li> </ol>
<p>Tool cannot be created because of the following reasons:</p> <ul style="list-style-type: none"> <li>• The validate subflow is not configured, and</li> <li>• An integration capability record with the same subflow name does not exist for the associated tool integration.</li> </ul> <p>Configure the subflow, and create the integration capability record with the Validate action and a subflow name for the tool integration, and try again.</p>	<p>This error occurs only for custom tool integrations.</p> <ol style="list-style-type: none"> <li>1. Navigate to <b>All &gt; Process Automation &gt; Flow Designer</b>.</li> <li>2. In the Flow Designer page, select the <b>Subflows</b> tab.</li> <li>3. Open the <b>DevOps Demo Validate Subflow</b> record from the list.</li> <li>4. From the Actions section, open the <b>DevOps Demo Validate Action</b> record by selecting the Open action in Action Designer () icon.</li> <li>5. Select <b>More Actions menu &gt; Copy Action</b>.</li> <li>6. Enter a new name for the action and select the application as <b>DevOps Integrations</b>, and select <b>Copy</b>.</li> <li>7. In the copied action, select <b>Inputs &gt; REST step</b>.</li> <li>8. In the Request Details section, enter resource path of your custom tool in the <b>Resource Path</b> field.</li> <li>9. Enter the API version of your custom tool in the <b>Query Parameters</b> field.</li> <li>10. Save the changes.</li> <li>11. Publish the action by selecting <b>Publish</b>.</li> <li>12. Navigate back to the <b>DevOps Demo Validate Subflow</b> record.</li> <li>13. Select <b>More Actions menu &gt; Copy Subflow</b>.</li> </ol>

Message	Troubleshooting steps
	<p><b>14.</b> Enter a new name for the subflow and select the application as <b>DevOps Integrations</b>, and select <b>Copy</b>.</p> <p><b>15.</b> In the copied subflow, delete the <b>DevOps Demo Validate Action</b> action and add the action you created in step 11.</p> <ul style="list-style-type: none"> <li><b>a.</b> In the <b>Action</b> field, select the action you created in step 11.</li> <li><b>b.</b> In the <b>aliasGR</b> field, select the Connection &amp; Credential Alias record from the <b>Data &gt; Lookup records</b> section.</li> <li><b>c.</b> In the <b>apiversion</b> field, select the api version from the <b>Data &gt; Subflow Inputs</b> section.</li> <li><b>d.</b> Save the changes.</li> </ul> <p><b>16.</b> Publish the subflow by selecting <b>Publish</b>.</p> <p><b>17.</b> Navigate to <b>DevOps &gt; Integrations &gt; Integration Capabilities</b> and create a record with the <b>Validate</b> action.</p> <p>For more information, see <a href="#">Create a DevOps tool integration</a>.</p> <p><b>18.</b> Associate the subflow you created in step 16 with the integration capability record.</p>
Tool cannot be created due to a connectivity issue. Check the <validate_subflow_name> subflow for more details, and try again.	<ol style="list-style-type: none"> <li><b>1.</b> Navigate to <b>All &gt; Process Automation &gt; Flow Designer &gt; Executions</b>.</li> <li><b>2.</b> Check for the latest execution of &lt;validate_subflow_name&gt; subflow to know the details of the error.</li> </ol>
Tool authorization credentials are invalid. Enter valid credentials, and try again.	Re-enter the correct username/password for the tool.
Tool cannot be created because there is no response received from the server. Enter a valid tool URL or Check if server is up and running, and try again.	<ol style="list-style-type: none"> <li><b>1.</b> Re-enter the tool URL.</li> <li><b>2.</b> Check the mid server.</li> </ol>

Message	Troubleshooting steps
The connection URL is incorrect, check the URL and try again.	Re-enter the tool URL.
Tool cannot be connected because the GitHub app slug name is incorrect. Enter the correct GitHub app slug name and try again.	You can find the GitHub app slug name on the settings page of your GitHub app. The GitHub app slug name is the URL-friendly name of your GitHub app. For example, if you have created a GitHub app with the name <b>Test App</b> , the corresponding URL-friendly GitHub app slug name will be <b>test-app</b> . In your GitHub url - "https://github.com/settings/apps/<test-app>", "<test-app>" is the GitHub app slug name. For more information, see <a href="#">GitHub documentation</a> .
Password value is too long and could be truncated after encryption. Please either reduce password length or increase field size.	From 18th January 2023, Jira has extended the length of API tokens for Atlassian accounts. You must increase the maximum password value to more than 255 in the discovery_credentials table to accommodate the extended character length. For more information, see the <a href="#">KB1269878</a> kb article and <a href="#">Atlassian documentation</a> .
Unexpected behaviour from remote host: Circular redirect to 'https://bitbucket.org/account/signin/?next=%2F...%2Frest%2Fapi%2F1.0%2Fusers'.	This error might occur if you try to connect to a BitBucket Cloud instance. BitBucket Cloud is not supported. You must use a BitBucket Server instance to connect to ServiceNow DevOps. Connect to your BitBucket instance via an MID server. A MID server is required if your tool instance is hosted on-prem. For more information about MID server, see <a href="#">MID Server selection</a> .

## Tool permissions

This section lists the troubleshooting steps for tool permission errors when you create a DevOps tool using workspace playbooks. Tool permission check guidelines are displayed in a pop-up when you connect to a tool from the workspace.

### Supported tools

- Azure DevOps
- Jenkins
- Jira
- Bitbucket
- GitHub and GitHub Enterprise

- GitLab
- JFrog

### Troubleshooting instructions

If any permission is missing, perform the following troubleshooting steps:

- Update the permissions for the provided credentials in the external DevOps tool. Select **Refresh** in the pop-up to perform the checks again.
- Close the pop-up, use different credentials, and select **Connect** again. The permission check pop-up appears with results of the newly entered credentials.
- Review the impact column, and if the impact does not affect your use-case, select **Continue** anyway.

If all permission checks are successful:

- Select **Continue** to progress to the next step in the playbook.
- If errors show up in the result, try again by selecting **Refresh**.
- If the error persists, update the [Tool permission check timeout per record \(in milliseconds\)](#) property and try again.

### Tool permission check timeout property

During the tool connection process, the system performs permission validations in the foreground, and restricts the tool connection page. For every permission validation, a REST API call is made to external tools to evaluate the availability of the permission. The REST API response time may vary based on your network settings. To avoid delays, you can specify how long the system should wait to get a response from an external tool before marking the permission as an error. You can specify this time using the [Tool permission check timeout per record \(in milliseconds\)](#) property (`sn_devops.permission_check_timeout`). If you find that some of the permission checks are resulting in an error, try increasing the timeout value, and select **Refresh** in the Permissions pop-up.

## Tool notification

This section lists the troubleshooting steps for errors that might occur in scenarios such as retrieving commit details, pipeline execution details, pull request details, test summaries, and so on. The reason for such errors and their troubleshooting steps are listed in the following table:

Reason	Troubleshooting steps
Authorization credentials are either invalid or do not have the minimum required permissions.	<ul style="list-style-type: none"> <li>• Verify if your third-party tool credentials have the required scopes. If you are using the workspace UI, you can navigate to the tool record, and select <b>More actions () &gt; Check credential permissions</b> to know about the required permissions.</li> <li>• Verify if you have entered the correct username/password for your third-party tool. If you are using the workspace</li> </ul>

Reason	Troubleshooting steps
	<p>UI, and you want to update your tool credentials, then navigate to the tool record, and select <b>More actions () &gt; Update credentials</b>. For more information on updating credentials, see <a href="#">Update third-party tool credentials in DevOps Change Velocity</a>.</p>
There is no response from the server. Check the server connection status, and try again.	<ul style="list-style-type: none"> <li>Verify that the DevOps tool server is reachable and responsive.</li> <li>If MID server is configured, verify if the server is up and running.</li> </ul>
Unable to process the request.	<p>Verify if the maximum number of instances associated with a webhook has exceeded. If you encountered this error for GitHub, verify if you have associated more than 20 instances to a webhook.</p>
Authorization credentials do not have the minimum required permissions.	<p>Verify if your third-party tool credentials have the required scopes. If you are using the workspace UI, you can navigate to the tool record, and select <b>More actions () &gt; Check credential permissions</b> to know about the required permissions.</p>
MID server configuration is invalid. Configure a valid MID server, and try again.	<p>Check if the mid server is running and reachable.</p>
The connection URL is invalid. Enter a valid URL, and try again.	<ol style="list-style-type: none"> <li>Navigate to <b>All &gt; Connections &amp; Credentials &gt; Connection &amp; Credential Aliases</b> and open the required connection &amp; credential record.</li> <li>From the Connections related list, open the related <b>Connection</b> form.</li> <li>On the <b>Connection</b> form, verify if the connection URL of the third-party tool is correct.</li> </ol>
The rate limit has been exceeded. Retry after {0}.	<p>Rate limit is the number of API calls an app or user can make within a given time period. Rate limiting is a technique to limit network traffic to prevent users from exhausting system resources. If the max rate limit allowed in your third-party tool has exceeded, you might encounter this error.</p>
Your credentials does not have the required scopes.	<p>Verify if your third-party tool credentials have the required scopes. If you are using the workspace UI, you can navigate to the tool record, and select <b>More actions () &gt; Check credential permissions</b> to know about the required permissions.</p>

Reason	Troubleshooting steps
	<b>credential permissions</b> to know about the required permissions.

## Retry errored inbound events

Run base system or modified scheduled jobs to retry processing errored inbound events. Specify the exceptions or errors to retry events that are in an *Error* state.

### Before you begin

Role required: sn\_devops.admin

### About this task

A scheduled job retries processing inbound events that are in *Error* state. You can choose to use the scheduled job with its base-system properties or customize the properties to suit your needs.

- Run a scheduled job on inbound events that are in error state.
- Update the list of exceptions and errors so that you can retry processing errored inbound events with those exceptions.
- Specify the count for maximum attempts the retry jobs runs on an inbound event.
- Specify the time elapsed since the last retry job. The retry job runs again on the errored inbound events after the duration you specify.

Based on your error or exception, perform any of the following steps to retry processing the inbound events.

### Procedure

1. Modify the *Retry Errored Inbound Events* schedule job frequency.

a. Navigate to **System Definition > Scheduled Jobs**.

b. Search and open the *Retry Errored Inbound Events* record.

**i Note:** By default, the scheduled job runs repeat every two minutes. You can modify the frequency (in minutes), as required.

c. Optional: Select the frequency to run the job, from the **Run** field, and configure the corresponding fields.

d. Click **Update**.

You have modified the frequency of the scheduled job.

2. Modify properties related to retry processing inbound events.

a. Navigate to **DevOps > Administration > Properties**.

b. Optional: Modify the **Maximum number of retries (for errored inbound events)** property.

**i Note:** The default value for this property is 3. You can modify the count as needed.

Processing inbound events are retried based on the count you specify.

- c. Optional: Modify the **Time elapsed (in minutes). The job will check errored inbound events** property.

**Note:** The default value for this property is 1440 minutes (24 hours). You can modify the count if required.

Based on the value in this property field, the system retries processing inbound events. For example, if the last attempt to process the inbound events was on 12 noon on 1 January, by default, the scheduled job retries processing errored events after 1440 minutes or 1 day from the timestamp of the last attempt to process the inbound event.

- d. Optional: To update the exceptions list based on which errored events are retried for processing, modify the **Errors or exceptions (comma separated) for which errored events are set to Retry** property.

**Note:** The default values of this property are *TimeOutExceptions*, *FlowObjectAPIException*. You can update the exceptions you want to retry processing in a comma-separated format.

The exceptions you specify in the **Errors or exceptions (comma separated) for which errored events are set to Retry** property are checked for the **Processing Details** field in the inbound event's record and retries processing.

## Result

The scheduled job to retry processing inbound events runs based on either the properties you have configured or the base system properties that are specified by default.

## Map spoke alias as parent alias for new DevOps tool

Create Jira, GitHub or Jenkins tool with parent alias set to respective spoke alias instead of DevOps Basic authentication. Use a script include to reset parent alias to spoke alias for existing or onboarded tool.

### Before you begin

**Important:** This feature is currently supported for Jira spoke v3.1, Jenkins spoke v2.1.1, and GitHub spoke v2.2.2, and later only.

- Ensure that you have an Integration Hub subscription.
- Ensure that you have set up the connection and credential alias using spoke for Jira, Jenkins and GitHub . For more information, see:
  - [Jira spoke](#)
  - [Jenkins spoke](#)
  - [GitHub spoke](#)
- Upgrade to DevOps Data Model and DevOps Integrations version 1.33 or later.

Role required: admin, sn\_devops.admin

### About this task

Whenever you create a Jira, Jenkins, and GitHub tool, the parent alias will be set to their respective spoke alias, if you have set up the spokes for the supported tools. In previous versions, when we created a tool, the parent alias was set to DevOps Basic Auth (sn\_devops.DevOps\_BasicAuth) for all the tools we onboarded.

If you have already created / onboarded tools with a connection and credential alias; the functionality does not change on upgrade and the parent alias remains DevOps Basic Auth (sn\_devops.DevOps\_BasicAuth) unless Jira, Jenkins, and GitHub tools are associated to spoke alias as parent alias. You can use a base system script include to ensure that the parent alias is reset to the spoke alias, for all existing and supported tools that were created earlier.

**Note:** Integration Hub uses aliases to manage connection and credential information.

Using an alias eliminates the need to configure multiple credentials and connection information profiles when using multiple environments. If the connection or credential information changes, you don't need to update any actions that use the connection. For more information, see [Connections and Credentials](#).

## Procedure

1. Navigate to **All > System Definition > Script Includes > DevOpsReparentingConnectionAliasFixScript**.
2. Copy the script from the **Script** field.
3. Navigate to **System Definition > Scripts - Background**.
4. Paste the script in the **Run script (JavaScript executed on server)** field.

**Note:** Ensure that the script is run in Global scope.

Once the script run is successfully completed, all supported tools created prior to the version 1.33 upgrade, are reset to their corresponding parent alias.

## What to do next

Verify that the parent alias is set to sn\_jira\_spoke.Jira for Jira, sn\_github\_spoke.GitHub for GitHub, and sn\_jenkins\_v2\_spoke.Jenkins\_v2 for Jenkins in the **Connection & Credentials Aliases** form.

## Delete orphan connection and credential aliases

Delete the connection and credential aliases that do not label any connection or credential record by running a background script. This helps you in freeing up space in your system.

### Before you begin

Role required: System Admin (admin)

## Procedure

1. Navigate to **All > System Definitions > System Background**.
2. Select the scope as **global**.
3. Enter the following script in the **Run script (JavaScript executed on server)** pane.

```
deleteOrphanAliases();
function deleteOrphanAliases() {
    var orphanAliases = new GlideRecord('sys_alias');
    orphanAliases.addEncodedQuery("nameSTARTSWITHDevOps-_");
    orphanAliases.query();
    while(orphanAliases.next()) {
        if (!checkIfAliasHasConnection(orphanAliases.getUniqueValue()))
            orphanAliases.deleteRecord();
    }
}
function checkIfAliasHasConnection(sysAliasId) {
```

```

var httpToolConnGR = new GlideRecord("http_connection");
httpToolConnGR.addEncodedQuery("connection_alias=" + sysAliasId);
httpToolConnGR.query();
if (httpToolConnGR.next())
    return true;
return false;
};

```

#### 4. Select **Run script**.

### DevOps Retry Policy

Enable the DevOps Custom HTTP Retry Policy for most tool communication from ServiceNow flows to automatically retry failed requests when a step encounters an intermittent issue, such as a network failure or request rate limit.

You can navigate to the action in **Flow Designer > Designer > Actions** and in the REST step, select **Enable Retry Policy**. For the DevOps Custom HTTP Retry Policy to take effect, override the default policy for the alias and select the DevOps retry policy.

To customize the retry configuration, access the DevOps Custom HTTP Retry Policy settings in **IntegrationHub > Retry Policy**.

### Manual configuration mode in DevOps

As an alternative to making a connection using the standard setup process, you can use manual configuration mode to set up a webhook manually.

For example, if you do not have admin privileges for a tool (to allow automatic configuration the webhook URL), you can send an email to the admin of the tool requesting the ServiceNow instance be added to the webhook. Once the instance is added, you can **Enter Manual Configuration Mode** and change the **Connection state** field to Connected (to connect manually).

This way you only need read-only permission to the tool. Once the connection is made, click **Exit Manual Configuration Mode**.

**Note:** The **Connection state** field can only be edited in manual configuration mode.

All planning, coding, and orchestration tool connections support manual configuration mode.

**Note:** Connection state automatically changes to disconnected if there is a change in configuration, such as URL, credentials, alias, type, or MID Server settings.

### Commits and task executions in DevOps

Run commits in DevOps are associated to a task execution.

For a commit to show up as a run commit, the commit record must exist in ServiceNow prior to the job/pipeline run.

In the event that jobs are rerun on the same commit, these conditions apply.

- Azure DevOps does not show any run commits.
- GitLab displays only the last commit as a run commit.
- Jenkins displays only the last commit as a run commit on which it was run. The difference of all commits is not shown.

Multiple commits in a single payload (commit arrays) have these limitations.

Tool	Max commits per payload
Azure DevOps	25
GitHub	1000
GitLab	20

## Associating multiple work items to a commit in DevOps

Multiple work items for a commit are supported in DevOps for Azure DevOps, Bitbucket, GitHub, and GitLab.

Work item syntax in the commit message can be customized to reflect the processes in your organization using the `DevopsCommitMessageParserSNC` script include in the **System Definition > Script Includes** module.

In order to link the commits with work items, the work item native ID is extracted from the commit message. In the base system, DevOps supports following commit message formats:

```
/**
 * Supported patterns
 * Colon pattern
 *   * Sample supported formats:
 *     * 1. STRY1,STRY2: Additional bug fixes
 *     * 2. STRY1 , STRY2 : Additional bug fixes
 *     * 3. STRY1, STRY2 : Additional bug fixes
 * Hash pattern
 *   * Sample supported formats:
 *     * 1. Fixes for #STRY1, #STRY2, #STRY3
 *     * 2. Fixes @#$#3 and #1 work item
 *     * 3. Fixes for #STRY1 #STRY2 #STRY3
 *     * 4. Fixes for AB#123
 * Jira pattern
 *   * Sample supported formats:
 *     * 1. JRA-123 fixed
 *     * 2. JRA-123 JRA-234 JRA-345 resolved
 */
```

If you want to add support for additional message formats to meet the processes in your organization, you can add a custom logic in the `DevopsCommitMessageParser` script include by navigating to the **System Definition > Script Includes** module. `DevopsCommitMessageParser` script include extends from `DevopsCommitMessageParserSNC`. The `DevopsCommitMessageParserSNC` has three regular expressions defined for identifying work item native IDs for supported message formats in the base system. See the following example to include a new custom message format that has work item native IDs in square brackets.

```
var DevopsCommitMessageParser = Class.create();
DevopsCommitMessageParser.prototype =
Object.extendsObject(DevopsCommitMessageParserSNC, {
    initialize: function() {
        DevopsCommitMessageParserSNC.prototype.initialize.call(this);
        this._customPattern = /\[(.*?)\]/g; // The regex pattern to match
        the words written inside square brackets.
```

```

    // Example commits message to match this custom pattern is :
    "[STRY1], [STRY2] Additional bug fixes"
},
getWorkitemsFromCommitMessage: function(message, branchName) {
    var workitems = [];
    // We first call the getWorkitemsFromCommitMessage method from
    the parent class to get the matching workitems ids for OOB formats
    var defaultWI =
        DevopsCommitMessageParserSNC.prototype.getWorkitemsFromCommitMessage.cal
    l(this, message, branchName);
    if (!gs.nil(defaultWI) && defaultWI.length > 0) {
        workitems = workitems.concat(defaultWI);
    }
    // Now call your custom method that returns an array of workitem
    native IDs matching custom pattern
    var customWI = this.getWIFromCustomPattern(message);
    if (!gs.nil(customWI) && customWI.length > 0) {
        workitems = workitems.concat(customWI);
    }
    // getUniqueWorkItems method from parent class removes duplicates
    from the workitems array
    workitems = this.getUniqueWorkItems(workitems);
    // return the final list
    return workitems;
},
getWIFromCustomPattern: function(message) {
    var wi = [];
    var l;
    var match;
    var matches = message.match(this._customPattern);
    if (gs.nil(matches))
        return wi;
    for (var i = 0; i < matches.length; i++) {
        l = matches[i].length;
        match = matches[i].substring(1, l - 1); // trim the brackets
        wi.push(match);
    }
    return wi;
},
type: 'DevopsCommitMessageParser'
});

```

Linking work items to a commit using the Azure DevOps user interface is also supported in DevOps.

You can view the list of associated work items in the DevOps Commit record, and in the Pipeline UI.

## DevOps Config

The ServiceNow® DevOps Config application validates and manages the configuration data of your enterprise applications across every stage of the DevOps pipeline.

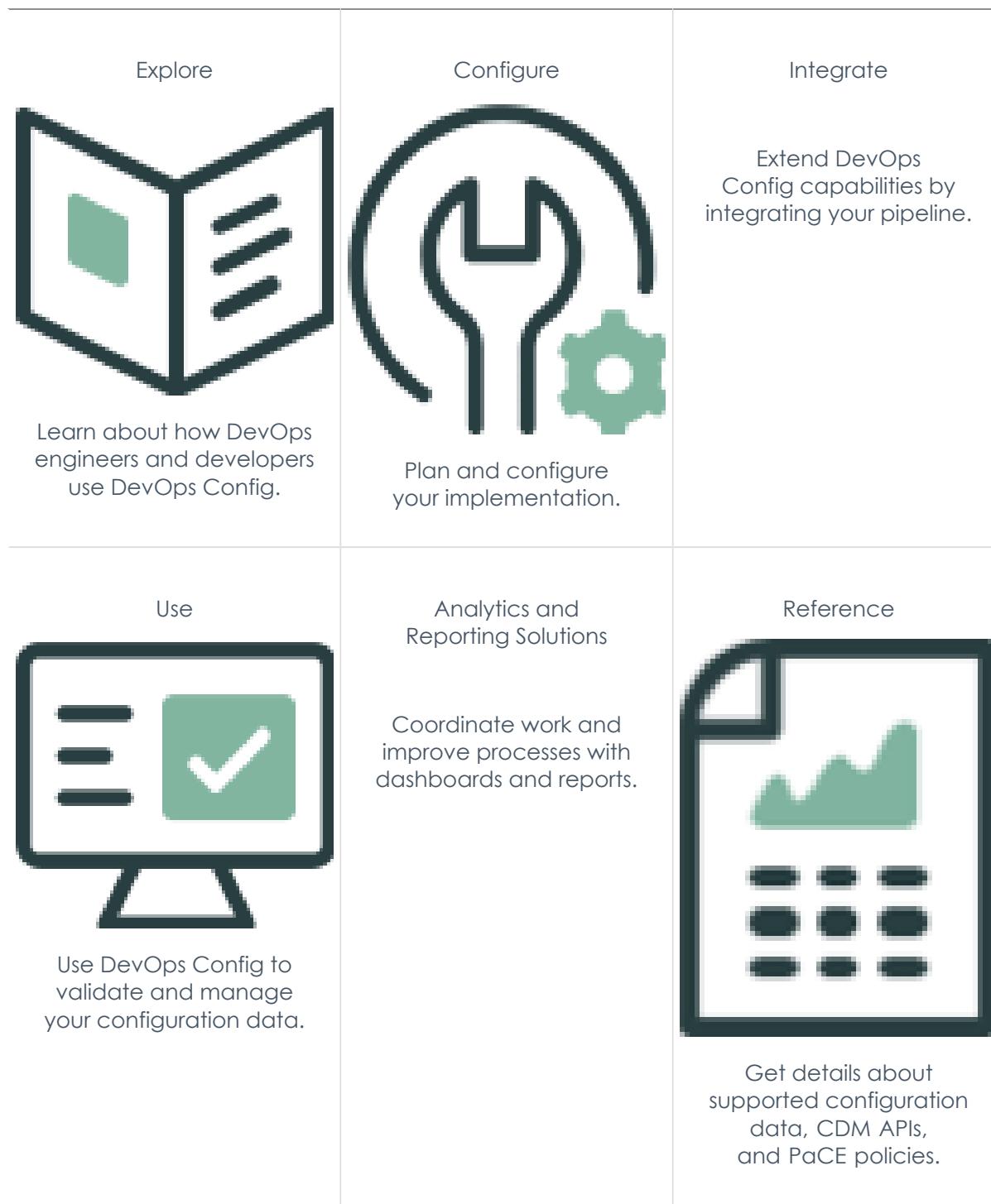
## Request apps on the Store

Visit the [ServiceNow Store](#) website to view all the available apps and for information about submitting requests to the store. For cumulative release notes information for all released apps, see the [ServiceNow Store version history release notes](#).

Watch this short video for an introduction to the DevOps Config application.

DevOps Config overview

## Get started



## Additional resources

- [DevOps Config release notes](#)
- [Ask or answer questions in the Now Community](#)

## Get help from DevOps Config resources

- [Search the Known Error Portal for known error articles](#)
- [Contact Customer Service and Support](#)

## Related ServiceNow applications and features

[DevOps Change Velocity](#) application.

## Exploring DevOps Config

Use DevOps Config to store and manage all of your config data as a single source of truth. You can also use DevOps Config to validate your config data before deployment, and resolve conflicts in deployed config data.

DevOps Config use cases for applications and Infrastructure as Code (IaC) include these main categories.

### Manage your configuration data

DevOps Config becomes the single source of truth for your configuration data, rather than the source tool. A consolidated model manages and secures config data across multiple sources with role-based access control.

Although DevOps Config prevents non-compliant changes by validating your configuration data before deployment, security of the configuration data can't be enforced if the data is kept at the source and not stored in DevOps Config.

- Workflow

DevOps Config manages all your data in one location, validates it as it's written, and exports, when needed.

- DevOps Engineer persona

Use DevOps Config and DevOps Config API to manage and validate configuration data. Thus, enabling DevOps teams to release at a faster speed, ensuring that no risky or non-compliant changes are introduced in production.

Use automated gates in a CI/CD pipeline or deployment script so that a deployment is stopped if any change to the application or infrastructure configuration is deemed risky or non-compliant.

Manage DevOps Config as more policies are added and more exporters are defined.

## Validate your configuration data

DevOps Config acts as a test tool by automatically validating your configuration data before deployment to prevent non-compliant changes, while ensuring adherence to policy frameworks.

Validation before deployment occurs by executing policies on the configuration data. The DevOps Config Policy content pack includes generic policies that check for standard issues, but can be customized based on use case.

- Workflow

When configuration data is changed or added, DevOps Config runs policies on the configuration data that's stored across multiple sources, validates it, and returns the outcome.

In the pipeline, the decision on whether to deploy is made, and the configuration data is retrieved from the source (Git, for example) to deploy.

- App Engineer (or IT infrastructure owner) persona

Use DevOps Config to validate configuration data. Thus, making sure no risks are introduced and that all changes are compliant with company policies before any changes are applied in a production environment.

Since the tool integrates with the existing toolset, there's no change to the way work is done and there are no new tools to learn. Changes made to configuration data are validated in the background, and when the outcome is reported, action can be taken.

## Resolve conflicts in deployed config data

Use root cause analysis of configuration-related outages or alerts to quickly identify and resolve unintended config data changes, also known as configuration "drift." Compare current and past versions of intended config data changes attached to change requests, and roll back to the desired state when needed.

Watch this short video to see how config data snapshots in DevOps Config can help you identify issues caused by unintended config data changes.

DevOps Config investigation with config data

For more information, see [Investigate an alert that involves a change to config data](#).

## DevOps Config and DevOps Change Velocity

DevOps Change Velocity collects data from all of your DevOps tools, providing visibility across the entire lifecycle of deployment, while DevOps Config manages and validates your DevOps configuration data.

DevOps Change Velocity is a horizontal solution orchestrating on top of all of your separate DevOps tools. DevOps Change Velocity features change acceleration and can pause your pipeline. DevOps Config serves a different purpose.

DevOps Config is a point solution that functions as one of your DevOps tools, managing and testing your configuration data prior to production.

## DevOps Config works better together with DevOps Change Velocity

Use DevOps Config with DevOps Change Velocity to manage and validate configuration data collected by DevOps Change Velocity.

DevOps Change Velocity is an umbrella product that sits on top of your entire release pipeline (planning, coding, testing, orchestration tools) as your application supply chain (value stream map).

DevOps Config manages, validates, and provides results on how your application and infrastructure are configured, which are the elements instrumented in DevOps Change Velocity.

DevOps Change Velocity and DevOps Config are synced by app object.

With the DevOps Change Velocity change control feature, a change request is created for DevOps Config configuration changes.

In DevOps Change Velocity, navigate to **DevOps > Orchestrate > Pipeline Change Requests** to view and approve DevOps change requests created by DevOps Config changes. CDM snapshots are listed in the **Config Data** related list of the DevOps change request.

## Using an app in DevOps Config

When you create an app in DevOps Config, not only is it the container for the config data of the application, but the application model you choose links DevOps Config with other ServiceNow products, including DevOps Change Velocity.

Each application model has an SDLC Component of the CMDB that is the link between ServiceNow DevOps pipeline products.

When you create an app in DevOps Config, it's synced with DevOps Change Velocity. Updates and deletions to the app made in either application are also synced.

**Note:** An SDLC-C cannot be deleted if there is a DevOps Config or DevOps Change Velocity app associated with it.

### Related entities and attributes

Mapping:

- 1:1 mapping between SDLC-C and App Model
- 1:1 mapping between DevOps Config app and SDLC-C
- 1:1 mapping between DevOps Change Velocity and SDLC-C

## DevOps Config powered by CDM and PaCE

DevOps Config uses Configuration Data Management and Policy as Code Engine platform capabilities to manage configuration data and policies.

## DevOps Config platform capabilities

### Configuration Data Management (CDM)

CDM is one of two main capabilities in the DevOps Config product. The CDM plugin manages application and/or infrastructure configuration data within the Now Platform.

CDM consists of the backend tables and logic that handles all aspects of an application configuration data model. The workspace pages that users interact with to manage their data model are also managed by this plugin.

The CDM Context Menu Component plugin is a sub-component of the CDM plugin. It provides the context (for example, three dot) menu for the node list pane in the Config Data tab for an application.

### Policy as Code Engine (PaCE)

PaCE is the second main capability in the DevOps Config product. It provides a robust policy engine that uses a policy script (written in javascript) to evaluate a snapshot of configuration data and verifies compliance.

PaCE provides of a comprehensive versioning system so users can audit their policy changes over time. It also contains a test playground to evaluate potential policy script changes before going live with user configuration data.

## DevOps Config core application

### DevOps Config

DevOps Config is the main application for the DevOps Config product.

The DevOps Config application consists of the workspace that connects users into their data model to manage their config data per application, manage policies to validate their config data, and review the health of their application configuration data.

The Insights dashboard widgets lets users gain insights on their configuration data acumen, and build better habits in managing their configuration data.

## DevOps Config content packs

### DevOps Config Exporter Content Pack (Optional)

The DevOps Exporter Content Pack is an optional plugin that contains curated exporters that the DevOps Config team created and that users can leverage out-of-the-box to export the config data from their snapshots as they see fit.

### DevOps Config Policy Content Pack (Optional)

The DevOps Config Policy Content Pack is an optional plugin that contains curated policies that the DevOps Config team created and that users can leverage out-of-the-box so they can get to validating their config data right away.

**Note:** if users install either of these optional plugins first, they will get DevOps Config and all Dependencies plugins automatically installed as well.

## DevOps Config dependencies

These plugins are installed with core DevOps Config applications. They are critical to DevOps Config and do not function properly without them.

## CMDB CI Class Models

CDM leverages the CMDB CI Class Models plugin to connect aspects of user application configuration data model to the Now Platform CMDB.

Applications are linked to an Application Model, which is part of the Build phase of the Common Service Data Model (CSDM). Similarly, deployables are linked to an Application Service, which is part of the Operations phase of the CSDM.

The link between DevOps Config and the CSDM is critical for solving problems that may arise across a user software factory.

For example, an operator is able to trace the cause for an alert for a particular CI, to a configuration change, to an application service (of which that CI is a part). This can reduce mean time to resolution, or MTTR, from hours to minutes.

For more information on how DevOps Config leverages the CSDM, see [DevOps Config product view](#).

## PA Premium

Performance Analytics (PA) Premium plugin is leveraged by the DevOps Config Insights plugin to allow for more than 180 days of reporting data for all DevOps Config Insights widgets.

## Expanded Model and Asset Classes

The Expanded Model and Asset Classes Store application adds enterprise model and asset classes that extend out-of-the-box product model and asset classes within the Configuration Management Database (CMDB) class hierarchy.

These extensions include class descriptions, identification rules, identifier entries, and dependent relationships.

## DevOps Config terms

These are some of the common DevOps Config terms.

### Application

The logical composition of an application or application stack, containing the relevant configuration data needed to provision or deploy into a fleet. This can range from a traditional, monolithic structure to a modern structure that can contain multiple micro-services.

### Collection

A set of components that make up a release composition. Collections are included into deployables, and can be used to test different versions of components.

This example shows release-1.0 of an application that is currently deployed to the Production environment.

The microservice team that owns the payment service makes a hotfix and decides to add that into a release-1.1 collection that they are now testing in their Test environment.

### Component

The smallest aggregation unit of configuration data. A microservice that is part of a larger application stack is an example of a component.

## Configuration Data Item (CDI)

A single unit of configuration data (for example, connection string, heap size, etc.) that is stored as a key-value pair.

### Data Model

A hierarchy of related configuration data used to deploy an application or infrastructure.

### Deployable

A subset of the data model, which encapsulates the set of configuration data that is used to deploy/provision an application or infrastructure for a specific target environment. It generally shares the same nomenclature (for example, Production, Staging, Test, Development, etc.).

### Snapshot

The complete data model of a deployable at the time a configuration change is committed. This includes any included components, collections, and vars, as well as deployable-specific vars and overrides.

### Vars

Variable configuration data that can be used to roll up configuration values that are used multiple times throughout different sections in the data model.

Vars created at the component level can be reused anywhere further down, like in a collection and/or deployable. However, vars defined in a particular deployable are intended to be used only in that context (for example, environment-specific creds, memory settings, etc.).

## Configuring DevOps Config

The DevOps engineer role installs and sets up DevOps Config, which is used to validate config data (committed by developers, or app engineers) before deployment.

Data validation configuration tasks	(Optional) Advanced configuration tasks to extend the use of the data model
<p>1. <a href="#">Install DevOps Config</a>.</p> <p>Install DevOps Config application, content packs, and pipeline plugins.</p> <ul style="list-style-type: none"> <li>• DevOps Config application.</li> <li>• (Optional) DevOps Config Policy content pack.</li> <li>• (Optional) DevOps Config Exporter content pack.</li> <li>• (Optional) Pipeline integration with DevOps Config. <ul style="list-style-type: none"> <li>◦ ServiceNow DevOps extension for Azure DevOps.</li> <li>◦ Jenkins plugin for ServiceNow DevOps.</li> </ul> </li> </ul>	N/A

Data validation configuration tasks	(Optional) Advanced configuration tasks to extend the use of the data model
<p><a href="#">2. Create an application in DevOps Config.</a></p> <p>Create an application to set up the entity, and link between ServiceNow products.</p>	N/A
<p><a href="#">3. Create a deployable in DevOps Config.</a></p> <p>Create a deployable to define the environments to deploy the configuration data for your application (typically Dev, Test, and Production).</p>	Extend the use of the data model by using <a href="#">components</a> and <a href="#">collections</a> for a deployable in DevOps Config.
<p><a href="#">4. Upload your configuration data.</a></p> <p>Upload your configuration data from each tool source to DevOps Config by creating and running an import script.</p>	Upload configuration data depending on use case.
<p><a href="#">5. Define policies in DevOps Config.</a></p> <p>Define policies to run for validation of config data using default DevOps Config policies.</p>	Administrate the full life cycle of PaCE policies and create your own.
<p><a href="#">6. Map policies to a deployable.</a></p> <p>Map policies to a deployable to run for validation of config data.</p>	Extend the use of the data model by mapping policies to components and collections in DevOps Config.
<p><a href="#">7. Define exporters in DevOps Config.</a></p> <p>Define exporters to export config data to your pipeline using default DevOps Config exporters.</p>	<ul style="list-style-type: none"> <li>Export config data for use by your deployment tools downstream in your CI/CD pipeline.</li> <li>Extend the use of the data model by <a href="#">creating a custom exporter</a>.</li> </ul>
<p><a href="#">8. Configure your Azure or Jenkins pipeline in DevOps Config.</a></p> <p>Configure your pipeline to interact with your data model.</p> <p>Use DevOps Config <a href="#">Azure DevOps pipeline tasks</a> and <a href="#">Jenkins pipeline actions</a> to integrate your pipeline with DevOps Config.</p>	N/A
<p><a href="#">9. Run validation in DevOps Config.</a></p> <p>Run validation and review the results.</p>	N/A

## Install DevOps Config

Install the DevOps Config application from ServiceNow Store applications. Visit the [ServiceNow Store](#) website to view all the available apps and for information about submitting requests to the store. For cumulative release notes information for all released apps, see the [ServiceNow Store version history release notes](#).

### Before you begin

Role required: admin

### About this task

#### Procedure

1. Navigate to **System Applications > All Available Applications > All**.
2. Click the **Not Installed** tab to view a list of applications available for installation.
3. Locate the DevOps Config application and click **Install**.  
Configuration Data Management and Policy as Code Engine platform capabilities are also installed.

### Install DevOps Config Policy content pack

Install DevOps Config Policy content pack for a default set of DevOps Config policies that you can use as-is, or customize for your needs.

### Before you begin

- Note:** You must install the DevOps Config application before installing the DevOps Config Policy content pack.

Role required: admin

### About this task

The DevOps Config Policy content pack contains a set of [default DevOps Config policies](#) to validate your configuration data.

#### Procedure

1. Navigate to **System Applications > All Available Applications > All**.
2. Click the **Not Installed** tab to view a list of applications available for installation.
3. Locate the DevOps Config Policy content pack and click **Install**.

### Install DevOps Config Exporter content pack

Install DevOps Config Exporter content pack for a default set of DevOps Config exporters that you can use as-is, or customize for your needs.

### Before you begin

- Note:** You must install the DevOps Config application before installing the DevOps Config Exporter content pack.

Role required: admin

### About this task

The DevOps Config Exporter content pack contains a set of [Default DevOps Config exporters](#) to export your configuration data.

## Procedure

1. Navigate to **System Applications > All Available Applications > All**.
2. Click the **Not Installed** tab to view a list of applications available for installation.
3. Locate the DevOps Config Exporter content pack and click **Install**.

## Create an application in DevOps Config

Create an app in DevOps Config to manage its configuration data, and link DevOps Config with other ServiceNow products, including DevOps Change Velocity.

### Before you begin

An admin can create, edit, delete, and view DevOps Config apps.

Role required: sn\_devops\_config.admin

### About this task

An app in DevOps Config is the container for the config data of an application. An app also ties it with the DevOps Change Velocity application, where configuration data can be viewed in the DevOps Pipeline UI.

In DevOps Change Velocity, an app links work items, code commits and CI/CD pipeline executions together in the DevOps data model, which is displayed in change request, as well as the DevOps Insights dashboard.

Apps in the DevOps Config application and in the DevOps Change Velocity application are synced. Meaning, when an app is created in DevOps Config, it is also created in DevOps Change Velocity, if installed, and vice versa.

You can edit or delete apps in DevOps Config. When you delete an app:

- In DevOps Config, the state of the application is set to **Inactive** and is no longer accessible in DevOps Config.
- In DevOps Change Velocity, it's also deleted in DevOps Config (if installed).

## Procedure

1. Navigate to **All > DevOps Config > DevOps Config Workspace**.
2. Click the Apps icon () in the left navigation to open the Apps tab.
3. On the Applications form, click **New** to create an application and a new model, or to create an application and specify an existing model (or service).

For a **New application** without an existing application model, both application and model are created.

For a new application to be tied to an **Existing application model or service**, only the application is created.

The CSDM application model (name, owner, manufacturer) functions as the link between ServiceNow DevOps products. There is an SDLC Component of the CMDB for each CSDM application model.

4. In the Overview tab, fill in the Application details and select **Save**.

Maintained by	Username of who maintains the application.
Application description	Description of the application.
Application model owner	Username of who owns the model.
Manufacturer	Manufacturer of the application.

The application does not yet contain config data, so the next step is to import your existing config data into the application.

Once you have imported config data and snapshots are generated, the list of snapshots are displayed in the **Overview** tab.

#### Example:

##### DevOps Config Applications list

##### DevOps Config App Overview

#### Create a CDM application that generates a new service in the CMDB

Generate a new service (an application service, application model, or dynamic CI group [infrastructure application type]) in the CMDB and create a CDM application as the container for the config data for the service.

#### Before you begin

Role required: CDM Admin [sn\_cdm.cdm\_admin]

#### About this task

For the application, you specify the number of deployables to generate and the SDLC environment that the deployables represent. When you upload existing config data into the application, the system maps the data as [name:value] pairs (config data items — CDIs) in the appropriate node in the data structure of the CDM application. For an overview of the process of preparing a new application to receive config data, see [Preparing an application for config data upload](#).

#### Procedure

1. Select the **Applications** icon () to open the Create new application page and then select **New application**.
2. On the **Overview** tab for the new CDM application, fill in additional application details that further identify the service in the CDM.

The CSDM application model (name, owner, manufacturer) functions as the link between DevOps features. There is an SDLC component of the CDM for each CSDM application model.

#### Application settings on the Details tab

Application name	Unique and meaningful name for the application.
------------------	---

Application model name	Name of the application model.
Maintained by	Groups with access to the application. Only members of these groups can access the application data.
Application description	Description that helps other users understand the purpose, scope, and intent of the application.
Application model owner	Name of the owner of the application model.
Manufacturer	Manufacturer of the application.

3. Select **Manage deployables** to open the **Edit deployables** tab where you can configure the deployables.
4. Select **Create deployables** and then fill in the settings on the Create Deployables dialog box to specify the quantity and environment type of deployables to create and the CMDB services to connect them to.

#### Create Deployables settings

Field	Description
Environment	<p>The kind of environment that is configured by the config data in the deployables.</p> <ul style="list-style-type: none"> <li>◦ Development</li> <li>◦ Test</li> <li>◦ Production</li> </ul>
Connection preference	<p>Option to connect the newly created deployables to the new services. Auto-connect is the only option when creating an application to an existing CMDB entry.</p> <p><b>i Note:</b> Because the DevOps pipeline applies config data in a deployable only to the services that the deployable is connected to, every deployable must be connected to a service.</p>
Quantity	<p>The number of deployables to create.</p> <p>Limit: 100.</p>

When you select **Create**, the system generates an application model in the CMDB and the CDM application with the specified number and types of deployables. The system auto-generates names of the form <AppName\_EnvironmentType\_number>.

5. On the **Edit deployables** tab, select the card for each deployable in turn and update its settings.

You connect each deployable to a service in the CMDB. The deployable holds the config data for its associated service.

The system auto-saves your updates to settings.

### Deployable details settings

Field	Description
Deployable name	<p>Default name for the deployable. To ensure that the auto-generated names are unique, the system adds a timestamp.</p> <p>Rename as needed.</p>
Deployable description	<p>Description that helps other users understand the purpose, scope, and intent of the deployable.</p>
Identifier	<p>Unique identifier that distinguishes this deployable from other deployables that share the same name and possibly many config data settings.</p> <p>For example, the production deployable that has the identifier <b>Pacific</b> might have different config settings for the language setting than the deployable that has the identifier <b>EU</b>.</p>
Connected to	<p>Service in the CMDB that the deployable is associated with. Select a service in the <b>Available services</b> list.</p> <p>Perform one of the following actions:</p> <ul style="list-style-type: none"> <li>◦ Link the deployable to a CI in the CMDB: Select the <b>cmdb_ci</b> and then select <b>Next</b>.</li> <li>◦ Create a CI to link to: Select <b>New</b>, define the CI, and then select <b>Next</b>.</li> </ul> <p><b>Note:</b> To continue, you must connect each deployable to a service.</p>

- Click **Next** to view the list of deployables on the **Edit deployables** tab.

### What to do next

The application is not yet structured to accept config data, so the next step is to add the data structure. See [Add the nodes that will contain config data to a new CDM application](#).

## Related reference

[CDM data model](#)**Create a CDM application that is based on an existing service in the CMDB**

Create an application as the container for all config data for a service (an application service, application model, or dynamic CI group [infrastructure] in the CMDB).

**Before you begin**

Role required: CDM Admin [sn\_cdm.cdm\_admin]

**About this task**

For the application, you specify the number of deployables to generate and the SDLC environment that the deployables represent. When you upload existing config data into the application, the system maps the data as [name:value] pairs (config data items — CDIs) in the appropriate node in the data structure of the CDM application. For an overview of the process of preparing a new application to receive config data, see [Preparing an application for config data upload](#).

**Procedure**

1. Select the Applications icon () to open the Create new application page and then select **Application based on existing services** to specify the CMDB service that the new application will hold config data for.
2. Fill in the settings on the Create Deployables dialog box to specify the quantity and environment type of the deployables and the CMDB services to connect them to.

**Create Deployables settings**

Field	Description
Environment	<p>The kind of code environment that is configured by the config data in the deployable.</p> <ul style="list-style-type: none"> <li>◦ Production</li> <li>◦ Development</li> <li>◦ Test</li> </ul>
Connection preference	<p>Option to connect the newly created deployables to services.</p> <ul style="list-style-type: none"> <li>◦ Auto-connect each new deployable to a service in the application model or dynamic CI group.</li> <li>◦ Manually specify the service to connect each new deployable to. You make the connections on the next page.</li> </ul>

Field	Description
	<p><b>i Note:</b> Because the DevOps pipeline applies config data in a deployable only to the services that the deployable is connected to, every deployable must be connected to a service.</p>
Quantity	<p>The number of deployables to create.</p> <p>If the number exceeds the number of services that are associated with the existing item in the CMDB, then the system generates new blank services.</p>

When you select **Create**, the system generates the application with the specified number and types of deployables and opens the **Edit deployables** tab for the application.

3. Select **Manage deployables** to open the **Edit deployables** tab where you can configure the deployables.
4. On the **Edit deployables** tab, update the settings for each deployable and then select **Next**.

You connect each deployable to a service in the CMDB. The deployable holds the config data for its associated service.

The system auto-saves your updates to settings.

#### Deployable details settings

Field	Description
Deployable name	<p>Default name for the deployable that the system generates.</p> <p>Rename as needed.</p>
Deployable description	Description that helps other users understand the purpose, scope, and intent of the deployable.
Identifier	<p>Unique identifier that distinguishes this deployable from other deployables that share the same name and possibly many config data settings.</p> <p>For example, the production deployable that has the identifier <b>Uruguay</b> might have different config settings for the currency setting than the deployable that has the identifier <b>Greenland</b>.</p>
Connected to	Service in the CMDB that is configured by the data in snapshots of the deployable. Select a service in the <b>Available services</b> list.

Field	Description
	<p><b>Note:</b> To continue, you must connect each deployable to a service.</p> <p>Perform one of the following actions:</p> <ul style="list-style-type: none"> <li>◦ Connect the deployable to a CI in the CMDB: Select the cmdb_ci and then select <b>Next</b>.</li> <li>◦ Create a CI to connect to: Select <b>New</b>, define the CI, and then select <b>Next</b>.</li> </ul>

The Review page opens.

- Click **Next** to view the list of deployables on the **Edit deployables** tab.

## What to do next

The application is not yet structured to accept config data, so the next step is to add the data structure. See [Add the nodes that will contain config data to a new CDM application](#).

Related reference

[CDM data model](#)

Related topics

[Viewing and editing config data](#)

[Define or update a component](#)

[Define or update a collection in an application](#)

[Create or update a variable CDI](#)

[How encrypted data is handled](#)

## Create and update a deployable

Create a deployable while working in a changeset. You can add multiple components and collections to a deployable.

### Before you begin

Role required: cdm\_editor or cdm\_admin

### About this task

**Important:** Save your changes whenever you are confident of the changes and before you leave the Config Data tab.

- Note:** Uploaded data items are sorted alphabetically within the structural folders to enable you to locate particular items quickly.

The system places the following limits on the count of config data items (CDIs):

- An application can contain a maximum total of 100,000 CDIs.
- Any single deployable can contain a maximum of 10,000 CDIs.

See [CDM system properties](#) for information on configuring the `sn_cdm.max_allowed_cdi_per_application` and `sn_cdm.max_allowed_cdi_per_deployable` property settings.

## Procedure

1. On the **Settings** tab for an application, select **Deployables** in the tree view and then select **New**.
2. Fill in the fields on the Enter details page of the wizard and then select **Next**.

### Enter details page

Field	Description
Name	Unique and meaningful name for the deployable.
Description	Description that helps other users understand the purpose, scope, and intent of the deployable.
Environment	<p>The kind of application or infrastructure environment that the deployable implements.</p> <ul style="list-style-type: none"> <li>◦ Development</li> <li>◦ Production</li> <li>◦ Test</li> </ul>
Restrict export	Option to ensure that only snapshots that pass validation can be exported.

The wizard moves to the Link to CMDB page of the wizard.

3. On the Link to CMDB page and perform either of the following actions:
  - Link the deployable to a CI in the CMDB: Select the `cmdb_ci` and then select **Next**.
  - Create a CI to link to: Select **New**, define the CI, and then select **Next**.
 The wizard moves to the Review page.
4. On the Review page, verify the settings for the new deployable. Select **Back** to change settings. Select **Create** to create the deployable. The system generates the deployable and displays it in the list on the **Settings** tab for the application.
5. Optional: Follow this procedure to add a collection to one or multiple deployables:

- a. For the collection node in the tree, select the menu icon (and select **Include in deployables**).
  - b. Select all deployables in which to include the collection and then select **Include**.
- Note:** Only deployables that do not already include the collection appear in the list.

The collection is included in each selected deployable.

(Optional) When you add a collection to a deployable, the Editing panel lists included collection names but not contents. The Preview panel displays the full state of the data.

6. Optional: Add CDI settings to the deployable that will override or overlay settings in components or collections.  
For more information, see [Define or update a component](#), [Define or update a collection in an application](#), or [Create or update a variable CDI](#).

7. Select **Save** to save the changes in the changeset and ready the changes in the changeset to be committed.  
The Editing panel, List view, and Preview panel refreshes to reflect the resolved state of the changeset. The system updates the changeset but does not update the application. Changes appear on the **Activity** tab. You must commit a changeset to update the config data for the application. After saving, you can move on to other activities and return later to edit the changeset.

8. Optional: Select **Commit**.

The Commit changeset dialog box opens and lists each deployable that is affected by the changes.

**Note:** After you commit a changeset, the changeset cannot be modified.

The system generates a snapshot of each deployable that is affected by the changes. The dialog box offers the following **Additional actions**:

- If no deployables are affected by the changes, select **No additional actions**. The system will update the application to persist the changes.
- Select **Validate snapshots** to perform the following operations on all generated snapshots:
  - The system executes all policies that are mapped to the associated deployable.
  - If the snapshot passes all policies, then the Validation status value is set to **Passed** and no entries are listed on the **Validation Results** tab. If validation was invoked with the **Publish Valid** publish option, then valid snapshots are auto-published.
  - If any policy encounters issues, then the Validation status value is set to **Failed**. The **Validation Results** tab displays all failures or warnings returned by policies.
  - If any policy failed to run to completion, then the Validation status value is set to **Execution error** and the **Validation Results** tab indicates that validation failed due to error.
- Select **Validate and publish snapshots** to perform validation as described in the **Validate snapshots** action. Only snapshots that pass validation are published. Only published snapshots can be exported.

Select **Commit**.

- The system updates the application to persist the changes to config data.
- The changeset state changes to **Committed**.
- The changeset cannot be modified.
- If snapshots were created, the confirmation message includes a link to the snapshots.
- If an error occurs during the commit process, the changeset state is set to Commit failed.

## What to do next

[Map policies to the deployable](#)

Related reference

[CDM data model](#)

Related topics

[Viewing and editing config data](#)

[Define or update a component](#)

[Define or update a collection in an application](#)

[Create or update a variable CDI](#)

[How encrypted data is handled](#)

## Delete a deployable

Delete a deployable to delete its config data and all associated snapshots.

### Before you begin

Role required: CDM Admin [sn\_cdm.cdm\_admin]

### About this task

- Note:** When you delete a deployable, existing snapshots cannot be exported or validated and cannot be referred to in exporters or policies.

### Procedure

1. For an open application, select the **Settings** tab.
2. Select **Deployables** to view the list of deployables for the application.
3. Select one or more deployables and select **Delete**.
  - The deployables no longer appear in any list.
  - All config data in the deployables is deleted.
  - All existing snapshots are deleted.
  - You can create a new deployable with the same name as a deleted deployable.

## Uploading your config data

You first create a new CDM application structure and then upload the existing config data into the structure.

## How it works: Performing the initial upload of your existing config data into CDM data tables

1. Open the CDM user interface and create a new application. For example, let's create the Bookstore application to support an online bookstore. An application in CDM is the full collection of config data for an application service, application model, or dynamic CI group [infrastructure] in the CMDB.
2. Now open the Bookstore application and generate a new changeset so you can begin to put the data structure in place. (A more detailed overview of this process appears in [Preparing an application for config data upload](#).)

In the changeset, define the data structure that includes a node for each set of config data that you will upload. This process is called modeling the data. You add components, collections, deployables, and CDIs as needed. For information on each of these items, see [CDM data model](#). For additional information on adding nodes to an application, see [Preparing an application for config data upload](#).

In this example, two components and two collections have been added. Now, the **PaymentSvc-1.0** component is included in the **Release-1.0** collection. As with all collections, this means that any data in the **PaymentSvc-1.0** component is now included in the **Release-1.0** collection.

When you are satisfied with the structure (you can update at any time), you open the REST Explorer to begin the process of uploading the source config data into the CDM data tables. Do not commit the changeset — it remains open so you can upload data.

3. Using the REST APIs or the config data editor, set parameter values that specify the name of the application to upload to, the path within the CDM application data structure to place the data, the format of the source data (JSON in the example), and so on.

This example shows the REST API Explorer platform utility for clarity. This is an example of how you might construct an *Upload to components* REST POST request that creates a `dbProperties.json` node in the **PaymentSvc-1.0** component and then adds the source data to the new node. (You specify the source data to upload in another field.) The APIs are described in [Application API](#), [Changesets API](#), and [Snapshot API](#).

4. When you select **Send**, the API reads the original config data, uploads it and aligns it into the CDM data structure.

**Note:** Uploaded data items are sorted alphabetically within the structural folders to enable you to locate particular items quickly.

The system places the following limits on the count of config data items (CDIs):

- An application can contain a maximum total of 100,000 CDIs.
- Any single deployable can contain a maximum of 10,000 CDIs.

See [CDM system properties](#) for information on configuring the `sn_cdm.max_allowed_cdi_per_application` and `sn_cdm.max_allowed_cdi_per_deployable` property settings.

**Important:** Each time you submit a POST request, the API performs the POST and also generates an upload script. You can specify one of several script languages. The purpose is for you to use the code in your pipeline system to automate the upload process for this application in future uploads.

5. Back to CDM: Review and update the config data as described in [Preparing an application for config data upload](#).
6. When you are satisfied that the application is a full and correct representation of the config data, you can commit the changeset. The commit action generates a snapshot of each deployable and causes the API to store the data in CDM tables.

Now that the application is fully in place, you can manage the data as needed: update config settings, apply policies to validate the data, export valid snapshots of config data, and so on.

While you export any snapshot, the system can generate API code that you can use to automate the export process. See [Generate API invocation code for an exporter](#) for details.

#### Related topics

[Preparing an application for config data upload](#)

#### **Preparing an application for config data upload**

An application in CDM is the full collection of config data for an application service, application model, or dynamic CI group [infrastructure] in the CMDB. After you upload your source config data, the application can support all potential deployables that make up each version of the development, test, and production environments of the service.

#### **Overview: Preparing an application to accept uploaded config data**

You follow this general process to prepare an application to accept the upload of config data:

1. On the **Apps** tab, you, a user with the CDM Admin [sn\_cdm.cdm\_admin] role, create an application record.

The system generates an application that includes several standard folders in a hierarchical structure. You will map your existing config data into this data structure to enable the benefits that are described in [CDM data model](#).

The application supports creation of multiple deployables. For example, you might create a deployable for each typical environment: Development, Test, and Production. You might also create multiple versions of each deployable for each environment type.

2. Working in the CDM code editor, you now create a changeset — a draft copy of the application that you can edit.
3. While working in the changeset, you create the following types of nodes in the appropriate folders. This process models the config data, that is, it prepares the application to map your source config data into the CDM data structure.

#### Components

Components are the building blocks that typically represent the config data for a logical element of an application or a part of an infrastructure service.

For example, a monolithic app, a micro-service, a physical server, or a Docker template.

A component can contain variables that can take on different values in collections and deployables. More detailed instructions appear in [Define or update a component](#).

## Collections

A collection is the set of components that together define a release — you can think of a collection as a release composition.

A collection can contain variable or override settings that are specific to the particular version. For example, the VM config data used in release-1 is different from the data used in release-2. release-1 might use the value 2Gb for the *memory* setting ("memory": "2Gb") and release-2 might specify a different value ("memory": "4Gb"). In addition, a collection might include config settings that do not appear in its components. You might think of such values as "overlays".

## Deployables

You add and configure deployables in the data structure. A deployable is a config dataset (for a development, test, or production environment) that can be deployed into your CI/CD pipeline as a service. Each deployable in an application configures a service in the CMDB. For example, you might create three deployables, one for each environment type: Development, Test, and Production.

A deployable is made up of the collection or set of collections that define the release for a particular environment. The combination of collections +environment link to an application service in the CMDB or an infrastructure service.

A deployable can contain variable or override settings that are specific to the environment. For example, the *database* variable has one value in the development environment and a different value in the production environment. An override value in the production deployable might specify a required container parameter that is not needed in the development environment.

- Now that the structure is in place, you use the REST APIs or the CDM code editing panel to upload your existing configuration data into the changeset. The process is described in [Uploading your config data](#). See [CdmApplicationsAPI](#), [CdmChangesetsAPI](#), and [CdmSnapshotAPI](#).

You can upload the following types of datasets: component variables, components, collections, and deployables.

- After the data is uploaded, you return to CDM. You update variable and override values so that the relatively small set of components and collections can provide config data for all three deployable environments. For example, the *Development* deployable can use the same components and collections as the *Test* deployable. *Development* uses the default *database* variable value. *Test*, in contrast, uses a different value that is appropriate for the test environment.
- Now, save and commit the changeset. The system performs the following actions:

- Determine whether there are conflicts with other earlier commits. If the system reports a conflict, you must resolve the conflicts and re-commit or create a new changeset and re-do your changes. For more information on conflict resolution, see [Conflicts between changeset commits](#).
- Push all changes into the data model of the application (the config data is persisted).
- Generate a snapshot of each deployable that is affected by the changes in the changeset. The system validates config data by executing specified policies against a snapshot. At the moment that the snapshot is created, the snapshot can be published and used to export the config data. Snapshots are permanent records that cannot be edited.

The source config data is now held in CDM tables. You can now manage the data as needed: map policies to each deployable so that the snapshots can be validated, validate the data in a snapshot (apply the policies), export config data, and so on.

**Note:** You can map policies to an empty deployable, but that is not a typical procedure.

#### Related topics

- [Define or update a component](#)
- [Define or update a collection in an application](#)
- [Create and update a deployable](#)
- [Changesets and version control in CDM](#)
- [Conflicts between changeset commits](#)
- [Validating and correcting configuration data](#)

#### Add the nodes that will contain config data to a new CDM application

Add the data structure to a new application that will accept the uploaded config data into appropriate nodes.

#### Before you begin

Role required: CDM Admin [sn\_cdm.cdm\_admin]

#### About this task

When the system creates a new application, the application has only the highest level of structure: **components**, **collections**, and **deployables** folders. In this procedure, you add the component, collection, and deployable nodes that will accept the uploaded config data. When you upload existing config data into the application, the system maps the data as [name:value] pairs (config data items — CDIs) in the appropriate node in the data structure of the CDM application. For additional detail on this process of modelling the data, see [CDM data model](#).

#### Procedure

1. Create the components that will be included in the collections. See [Define or update a component](#). See [Define or update a collection in an application](#).
2. Add a collection to one or multiple deployables:
  - a. For the collection node in the tree, select the menu icon () and select **Include in deployables**.
  - b. Select all deployables in which to include the collection and then select **Include**.

- Note:** Only deployables that do not already include the collection or any of its descendants appear in the list. If the collection has the same name as an existing node, then the collection is transformed into an override or overlay.

The collection is included in each selected deployable.

When you add a collection to a deployable, the **Editing** panel displays the included collection names but not their contents (overrides or overlays do appear). This strategy reduces clutter and clarifies the structure. To view the fully-resolved content of the data, view the **Preview** panel.

3. Create the components that will be included in the collections. See [Define or update a component](#).
4. Optional: Add CDIs to the deployable that will override or overlay settings in components or collections. See [Define or update a component](#), [Define or update a collection in an application](#), or [Create or update a variable CDI](#).

- Note:** Uploaded data items are sorted alphabetically within the structural folders to enable you to locate particular items quickly.

The system places the following limits on the count of config data items (CDIs):

- An application can contain a maximum total of 100,000 CDIs.
  - Any single deployable can contain a maximum of 10,000 CDIs.
- See [CDM system properties](#) for information on configuring the `sn_cdm.max_allowed_cdi_per_application` and `sn_cdm.max_allowed_cdi_per_deployment` property settings.

5. Select **Save** to save the changes in the changeset and ready the changes in the changeset to be committed.  
The Editing panel, List view, and Preview panel refreshes to reflect the resolved state of the changeset. The system updates the changeset but does not update the application. Changes appear on the **Activity** tab. You must commit a changeset to update the config data for the application. After saving, you can move on to other activities and return later to edit the changeset.
6. Optional: Select **Commit**.  
The Commit changeset dialog box opens and lists each deployable that is affected by the changes.

- Note:** After you commit a changeset, the changeset cannot be modified.

The system generates a snapshot of each deployable that is affected by the changes. The dialog box offers the following **Additional actions**:

- If no deployables are affected by the changes, select **No additional actions**. The system will update the application to persist the changes.
- Select **Validate snapshots** to perform the following operations on all generated snapshots:
  - The system executes all policies that are mapped to the associated deployable.
  - If the snapshot passes all policies, then the Validation status value is set to **Passed** and no entries are listed on the **Validation Results** tab. If validation was invoked with the **Publish Valid** publish option, then valid snapshots are auto-published.

- If any policy encounters issues, then the Validation status value is set to **Failed**. The **Validation Results** tab displays all failures or warnings returned by policies.
- If any policy failed to run to completion, then the Validation status value is set to **Execution error** and the **Validation Results** tab indicates that validation failed due to error.
- Select **Validate and publish snapshots** to perform validation as described in the **Validate snapshots** action. Only snapshots that pass validation are published. Only published snapshots can be exported.

Select **Commit**.

- The system updates the application to persist the changes to config data.
- The changeset state changes to **Committed**.
- The changeset cannot be modified.
- If snapshots were created, the confirmation message includes a link to the snapshots.
- If an error occurs during the commit process, the changeset state is set to Commit failed.

## What to do next

The application does not yet contain config data, so the next step is to import your existing configuration data into the appropriate nodes in the application. See [Uploading your config data](#).

Related reference

[CDM data model](#)

Related topics

[Viewing and editing config data](#)

[Define or update a component](#)

[Define or update a collection in an application](#)

[Create or update a variable CDI](#)

[How encrypted data is handled](#)

## Define policies in DevOps Config

Define policies in DevOps Config to validate your configuration data.

### Before you begin

Role required: sn\_devops\_config.admin

### About this task

You can use or customize default DevOps Config policies to validate that your config data content is compliant, or [write and test custom DevOps Config policies](#).

### Procedure

1. In the **Admin** view, click the **Policies** tab.
2. Select a default policy or click **New** to [create a policy](#).
3. Enter the Policy Name, Category, and a Description for the policy and click **Save**.

**Note:** You must publish a policy version to make it current, and activate it before it can be invoked.

## Related topics

[How to write and test PaCE policies](#)

### Default DevOps Config policies

The DevOps Config Policy content pack contains a set of default DevOps Config policies to validate your configuration data.

You can use or customize these default DevOps Config policies to validate that your config data content is compliant, or [administrate the full life cycle of PaCE policies](#).

**i Note:** You cannot modify default policies. However, you can make a copy of the policy and customize your copy.

### All Keys/Values Comparator (allKeysValuesComparator)

Checks that all keys-value combinations from the main deployable are the same in other deployables.

Use case: Help detect drift from a blueprint.

#### Input arguments

- Exception list (list of keynames that should be ignored during validation).
- Main deployable is the reference blueprint.
- All extra deployables are compared to main one.

#### Error handling

- Error if key is present in main deployable but not in deployable X.
- Error if key is present in deployable X and not in main one.
- Error if key is present in both deployable but not with same value Error handling.

### Authorized eMail Domains (authorizedEmailDomains)

Checks that any email addresses found in the snapshot have an authorized email domain defined.

Use case: Prevent (sensitive) data from leaving a trusted company email domain.

#### Input arguments

- AuthorizedDomains: required, list of email domains that are approved. (For example, mydomain1.com,mydomain2.fr).
- exceptionList: optional, list of keynames that are ignored by the policy (case insensitive comparison are used in the policy).

#### Special logic

- List of authorized email domains is provided as a parameter in the rule.
- eMail values are identified based on regex expression.
- A list of keynames exception to ignore is provided.

## Authorized Hosts in URLs (authorisedHostsInURLS)

Checks that any URLs found in the snapshot are authorized as defined with a regex.

Use case: Prevent exposure of (sensitive) data through unauthorized URLs.

Input arguments

- exceptionList
- authorizedHostsRegex
- maxErrorDisplay

Special logic

- A list of keynames exception to ignore is provided as arg.
- A list of host regex to validate is also provided as parameter in the rule.
- An URL value is identified based on regex.

Error handling

Check that the input is a valid JSON object.

## Authorized Key/Value (KV) Combinations (authorisedKVCombinations)

Validates that for every keyName defined in the authorised list, its value is one of the provided ones. If not, it fails.

Use case: Check that a mandatory list of keys is present in the snapshot with only authorized values.

Input arguments

A JSON list of keys with list of authorized values for each key in the form of an array of values.

Error handling

Checks that the input is a valid JSON object.

## Authorized Ports Range (authorisedPortsRange)

Checks that for every keyName defined in the keysRegexWithRange input parameter, its value is within the defined range. If not, it fails.

It can be used for any range value check, not just portNumbers. For example, to check that all parameters defining heapSize are within the appropriate range.

Use case: Check that port names are within the acceptable range for the environment in which they are used.

Input arguments

- keysRegexWithRange: Required, JSON list of keys regex and port range used to identify a port and its approved range. For example:

```
{ "^\w+_\w+$" : "5000-5003", "^\w+.\w+\.\w+$" : "10000-65535" }
```

- exceptionList: Optional, list of keynames that are ignored by the policy (case insensitive comparison will be used in the policy).

## Special logic

- A key is identified as port if it respects a specific regex, such as "`^web_port$`".
- A list of keys regex with authorized range (min and max value) is provided as parameter in the script.
- An exception list is available to ignore some keyname.

## Certificate Validator (certificateValidator)

Checks if certificates have expired or will expire.

Use case: Validate if a snapshot contains certificates.

## Consistent Nodes Comparator (consistentNodesComparator)

Checks that all key-value combinations are the same across the deployable.

Use case: Validate that all first-tier nodes contain matching children key-value combinations in a snapshot.

## Correct Host Regex (correctHostRegex)

Checks the format for host-related keys to ensure that a host name is authorized to use.

Use case: Ensure the host name used is authorized in my organization by checking the format for host-related keys.

### Input arguments

- `authorizedHostRegex`: Regular expression pattern for the authorized host name.
- `exceptionList`: A list of keys to exempt from evaluation.
- `maxErrorDisplay`: Number of failures to display; default value is 10.

### Error handling

Check if keys are found that contain a URL value that does not match the authorized host name regular expression pattern.

## Detect Unused Variables (detectUnusedVariables)

Checks whether the `vars` folder of a deployable or a collection includes variables that are not used in a snapshot.

Results into a warning when a variable is found but not used in the snapshot.

### Input arguments

None

## Different Key Names Same Values (differentKeyNamesSameValues)

Checks that values are the same for different keys.

Use case: Obtain a finer grained control of conditions by checking that the values are the same for different keys.

### Input arguments

**keyPairs:** Data array of strings that are pairs of keys whose respective values should match (for example, "keyA=keyB").

#### Error handling

Check if specified key pairs do not have the same value.

### Different Values Nodes Comparator (**differentValuesNodesComparator**)

Checks that provided keys have different values across first-level nodes in a deployable.

Use case: Validates that a snapshot contains first-tier nodes with matching children keys but keys are not in the list of key names to search.

### Docker Image Format Validator (**dockerImageFormatValidator**)

Checks whether the docker image of a service follows the regex as specified in the input. For example, provide a regex value to ensure the tagging and versioning of docker images follow best practices.

Results into a non-compliant status when the image value does not match the regex input value in *imageExpectedRegex*.

#### Input arguments

- **imageExpectedRegex**
  - Regex value for a docker image.

Example regex value to ensure that the tag of an image is an alphanumeric value:

`^.*:( ([0-9]+.?) +([ -_]{1}[A-Za-z0-9.]+) ?$`

- Default value: # None
- Usage: Mandatory
- **serviceExceptionList**
  - Comma-separated list of services that are exempted from the validation.
  - Default value: None
  - Usage: Optional
- **maxErrorDisplay**
  - Maximum number of failures to display.
  - Default value: 10
  - Usage: Optional

### Docker Image Registry Validator (**dockerImageRegistryValidator**)

Checks whether the registry of a docker image is included in the authorized list of registries.

Parses the docker image for each service, fetches the value of a registry in an image, and then checks whether the fetched registry value is included in the *authorizedRegistryList* input.

Results into a non-compliant status when the image registry is not in the *authorizedRegistryList* input.

#### Input arguments

- authorizedRegistryList
  - Comma-separated list of registries.

Example:

```
k8s.example.io,gke.example.io,docker.example.com:8444
```

- Default value: # None
- Usage: Mandatory
- maxErrorDisplay
  - Maximum number of failures to display.
  - Default value: 10
  - Usage: Optional

## Docker Network Validator (dockerNetworkValidator)

Checks whether the network for a docker service is defined in the network sections.

Results into a non-compliant status when the network for a service is not defined in the top-level network section or the network for a service configured in the `labels/com.docker.lb.network` property is not defined in the service-level section and the top-level network section.

Input arguments

- maxErrorDisplay
- Maximum number of failures to display.
  - Default value: 10
  - Usage: Optional

## Duplicate Values (duplicateValues)

Checks for duplicate values within a snapshot.

Use case: Obtain a finer grained control of conditions by checking that snapshot values are different.

Input arguments

- exceptionList: Data array of strings that indicates the values for keys to except from evaluation; default is ["true", "false", "yes", "no", "enabled", "disabled"].
- includePath: Toggle to include full path versus just node name in output; default is true.
- pathSeparator: String to indicate the path separator (for example, "/" or "."); default is forward slash (/).
- maxErrorDisplay: Number of failures to display; default value is 10.

Error handling

Check if the snapshot has no keys with duplicate values.

## Forbidden Key/Value (KV) combinations (forbiddenKVcombinations)

Checks if there are any forbidden combinations like the input parameter provided. This policy loops over all the CDI key values in the snapshot.

Use case: Ensure you don't use forbidden values for a specific list of keys so that, for example, you don't use Dev/Test values in Production snapshots. Also, ensure that root or admin is not a username for a service running in production.

## Generic List Validator (genericListValidator)

Compares the key values of a snapshot with the list of values provided in the input. The key value might be a string, list of comma-separated strings, array of strings, or object. When the key value is an object, the keys of the object are used in the comparison.

Note the following points for the policy compliance:

- When the operator is `IN`, the policy is compliant when all the strings in the key value are included in the input list.
- When the operator is `EQUALS`, the policy is compliant when all the strings in the key value match all the strings, listed in any order, in the input list.
- When the operator is `CONTAINS`, the policy is compliant when the key value contains all the strings in the input list.
- When the operator is `NOT CONTAINS`, the policy is compliant when none of the strings in the key value exist in the input list.

### Input arguments

#### keysToValidateObject

- An array of objects where each object contains a key path, an operator, and a reference list.
- Allowed operators are `IN`, `CONTAINS`, `NOT CONTAINS`, and `EQUALS`.
- For example, to enforce that a user defined in the system `/root/user` path is either `user1` or `user2`, use the following input:

```
[  
  {  
    keyPath: ["system", "root", "user"]  
  
    operator: "IN"  
  
    referenceList: [user1, user2]  
  
  }  
]
```

- Default value: `[]`
- Usage: Optional
- `maxErrorDisplay`
  - Maximum number of failures to display.
  - Default value: `10`
  - Usage: Optional
- `caseSensitive`

- Boolean value to indicate whether the list comparison should be case sensitive.
- Default value: true
- Usage: Optional

## Generic Validator (genericValidator)

Checks that a key value is a valid based on an operator and reference value.

Use case: Validate an exact value for a key based on an operator and reference value.

## Key Existence Comparator (keyExistenceComparator)

Checks that all keys from a main deployable also exist in other deployables.

Use case: Compare your environment variables and check that they are consistent.

Input arguments

- Exception list (list of keynames that should be ignored during validation).
- Main deployable should be the reference blueprint.
- All extra deployable are compared to main one.

Error handling

Error if key is present in main deployable but not in deployable X.

## Key Naming Convention (keyNamingConvention)

Checks that key names follow a given convention.

Use case: Ensure consistency in the config data for my organization by checking that key names follow a given convention.

Input arguments

- maxLength: Maximum length for a key name.
- approvedPrefixArray: Data array of strings that are approved prefixes for a key name.
- approvedSuffixArray: Data array of strings that are approved suffixes for a key name.
- exceptionList: A list of keys to exempt from evaluation.
- includePath: Toggle to include full path versus just node name in output; default is false.
- maxErrorDisplay: Number of failures to display; default value is 10.

Error handling

Check if keys are found that violate the specified mapping inputs, including:

- Exceeding the maximum length.
- Having incorrect prefix/suffix.

## Key Path Validator (keyPathValidator)

Checks a set of keys (including key path) for specific values.

Use case: Obtain finer grained control of conditions by checking a set of keys (including key path) for specific values.

Input arguments

- pathAndDesiredValues: JSON object that contains a set of keys and a mapping of condition and value the keys must have.
- pathSeparator: String to indicate the path separator (for example, "/" or ",").
- ignoreIfNotFound: Toggle to ignore case where key is not found; default is true.
- maxErrorDisplay: Number of failures to display; default value is 10.

Error handling

Check if keys are found that have an unexpected value, OR if a key is not found and ignoreIfNotFound is set to false.

## Key Value Substring Check (keyValueSubstringCheck)

Checks for keys that contain a certain substring in their name, and have a certain substring in their value.

Use case: Obtain a finer grained control of conditions by checking snapshot substring values are the same.

Input arguments

keyNameWithKeyValue: JSON object that contains a required substring set including key name substring, and key value substring pairs.

Error handling

Check if snapshot contains keys and corresponding values that match specified substring patterns.

## List Comparator (listComparator)

Compares two lists of nodes names (case insensitive).

Use case: Ensure both nodes have the same keys by comparing the list of keys between two nodes.

Input arguments

- List of nodes that should have same values: string, comma separated list of nodes.
- List of extra nodes that should be compared to main one.

Error handling

Check that the input is a valid JSON object.

## Mandatory Keys (mandatoryKeys)

Ensures that a list of provided keyNames or nodeNames are present in the snapshot.

- Note:** There is no check on the value for those keyNames; it is sufficient the keyName or nodeName exists to pass the validation check.

Use Case: Validate that the configuration data is correct for the deployable by checking for some specific keyNames.

Input argument

List of keys: String JSON notation.

Error handling

Check that the input is a valid JSON object.

## Memory Limit Validator (memoryLimitValidator)

Checks whether the memory resources assigned to a key value are within the specified limits of a memory unit. #Compares case-insensitive key values of metric units (including KB, MB, and GB) and binary units (including KiB, MiB, and GiB). Converts a key value and its input values to bytes before comparing.

Results into a non-compliant status when a key value is not within the specified limits or is not a memory unit.

Input arguments

- pathAndLimitsArray
  - Array of key path and the minimum and maximum values for memory unit limits as min and max.
  - Default value: None
  - Usage: Mandatory

Example:

```
[  
  {  
    keyPath: ["deploy", "resources", "limits", "cpu"],  
    min: "250m",  
    max: "500m"  
  },  
  {  
    keyPath: ["deploy", "resources", "limits", "cpu"],  
    min: "1GB",  
    max: "8GB"  
  }  
]
```

- maxErrorDisplay
  - Maximum number of failures to display.
  - Default value: 10
  - Usage: Optional

## Mandatory Tags List (mandatoryTagsList)

Checks that all tag/tags keys respect a list of required values.

Use case: Validate a snapshot that contains all tags in a list of tags.

## Mandatory Value (mandatoryValue)

Ensures that a value is set for the list of provided keyNames. The value check is only checking it is not null or empty.

Use Case: Ensure that certain keys contain a value because 3rd party automation tools might fail in case of a null value or empty string.

### Input arguments

- Comma separated list of keyNames.
- Case sensitive search (true/false).
- For each keyname, stop at first key/value found (true/false).

### Special logic

Warning is raised in case keyname is not found.

## Mandatory Value By Type (mandatoryValueByType)

Ensures that a list of provided keyNames have a value assigned if present in the snapshot. The policy also ensures that the value has the required value type. String, boolean and integer are supported for the value data types.

Use Case: Ensure that certain keys contain a proper value with of the right value type because 3rd party automation tools might fail in case it expects for instance an integer but a string is provided instead.

### Input arguments

List of keyNames by valueType: nested JSON notation.

### Error handling

- Check that the input is a valid JSON object.
- Check that the provided valueTypes are supported by the logic. If not, gracefully fail and still validate for the other valueTypes.

## No Clear Sensitive Data (noClearSensitiveData)

Finds all CDIs of which its name contains one of the keyWords that are provided. If a CDI keyName contains such a keyWord, the policy checks that its value is encrypted. If not encrypted, the policy triggers a failure.

Use Case: DevOps Config is used by organizations to properly manage and govern sensitive data (passwords, API keys, tokens, etc) through its permissions based access controls. This policy ensures that all config data items that should be encrypted are actually encrypted.

### Input arguments

- KeyWords: required, list of keywords (part of keynames) used to identify a sensitive value (case insensitive comparison is used in the policy); default is "pass,pwd,secret".
- exceptionList: optional, list of keynames that will be ignored by the policy (case insensitive comparison will be used in the policy).

### Special logic

- A key is identified as credential if it contains “pass”, “pwd” or “secret” in its keyname. This list should be parametrized in the rule.
- An exception list is available to ignore some keyname (like “password\_description”).

## No Duplicate Keys (noDuplicateKeys)

Validates that all provided keyNames only exist once within the snapshot. A failure is created in case a keyName appears 2 or more times.

Use Case: deployment automation tools and applications typically handle multiple occurrences of settings badly and can lead to downtime of services.

For instance, it makes no sense to define the database connection string multiple times in a single config data source – it is typically a sign of copy/paste or human error.

It also makes troubleshooting and management of configuration data more cumbersome as the DevOps engineer might think they changed the value but it actually existed as a duplicate in a different place. This policy will prevent such an issue.

Input arguments

List of keys to check.

Special logic

An exception list is available to ignore some keyname.

## No Empty Values (noEmptyValues)

Checks that all keys have a value not null or empty.

Use case: Ensure that values are set for all keys.

## No Forbidden Key/Value Combinations (noForbiddenValues)

Validate that a specific list of values are not used for CDI or variable values, except when their keyName is in the list for excluded keyNames.

Use Case: Ensure that certain settings which are allowed in non-production environments (for example, `root` for certain userNames, or `http://...` in a URL, or `debug` for `infoLevel`) do not appear in deployable snapshots for a production environment.

So rather than having to specify a list of forbidden keyValue combinations, this policy is more generic as it checks if certain values are in use or not without that you need to specify for which keys.

Input arguments

Forbidden keyvalues : string JSON notation.

Special logic

Count the nbr of forbidden keyValue combinations that were found.

Error handling

Check that the input is a valid JSON object.

## No FTP, No HTTP, No LDAP (noFTP,noHTTP,noLDAP)

Checks that there are no FTP/HTTP/LDAP URLs.

Use case: Verify that all URLs use FTPS/HTTPS/LDAPS protocol.

#### Input arguments

- CaseSensitive, optional, true/false, if search for value and exception keys should be case sensitive or not (default is false).
- exceptionList: optional, list of keynames that will be ignored by the policy.
- SearchValue, required, string to search in the values; default is "ftp:/", "http:/", "ldap:/".

#### Special logic

Ignore password values (not decrypted).

## No Whitespace (noWhiteSpaceAllowed)

Finds all CDIs whitespace character in its value. This includes leading, trailing or within the CDI value. It is best practice to not use white space in configuration data values.

The policy provides an optional list of keyNames (comma separated string) to exclude from the policy check (for instance, a description field).

Use Case: Prevent white space from causing issues for tools or applications when settings are applied.

For example, configuration data is often managed in source files located on shared drives and in repositories, and managed by various people. Because of this, it's easy to make an error and add a space on the left or right.

## Node Keys Comparator (nodeKeysComparator)

Compares the keys between two nodes to ensure both nodes have the same keys.

Use case: Ensure both nodes have the same keys by comparing the keys between two nodes.

#### Input arguments

- fromNode: Baseline node path.
- toNode: Comparison node path.
- includePath: Toggle to show node path hierarchy in output; default value is true.
- maxErrorDisplay: Number of failures to display; default value is 10.

#### Error handling

Check if keys found in the base node are not found in the comparison node, and vice-versa.

## Nodes Value Comparator (nodesValueComparator)

Compares the values for keys between two nodes.

Use case: Ensure consistency between nodes by comparing the values for keys between two nodes.

#### Input arguments

- fromNode: Baseline node path.
- toNode: Comparison node path.
- sameValueKeys: Data array of keys that should have the same values between baseline and comparison nodes.
- diffValueKeys: Data array of keys that should have different values between baseline and comparison nodes.
- maxErrorDisplay: Number of failures to display; default value is 10.
- allowMissingKeys: Toggle allowing for keys to be missing in nodes; default is false.

#### Error handling

Check if keys are found in both the baseline and comparison node paths that do not align to the respective same/different values lists in which they are specified.

### **Same Values Comparator (sameValuesComparator)**

Checks that all keys which exist in the primary deployable also exist at the same path in all the other deployables. An exception list for keyNames can optionally be defined.

Use Case: Compare environments which should have the same configurations. You can assign this policy for instance to your prd-APAC deployable and check that all keys also exist in the US and EU production instance (prd\_US, prd\_EU, prd\_APAC). Or compare your pre-production staging with all the production environments, just to ensure no configuration data drift is occurring.

#### Input arguments

- List of keys that should have same values: string, comma separated list of keynames.
- List of extra deployables that should be compared to the main one.

#### Error handling

Check that the input is a valid JSON object.

### **Same Key Value (sameKeyValue)**

Checks whether the values of a key if included in multiple components of a snapshot are same across all components.

Results into a non-compliant status when a key has different values in multiple components.

#### Input arguments

- keyList
  - Comma-separated list of key names
  - Default value:#None
  - Usage: Mandatory
- maxErrorDisplay

- Maximum number of failures to display.
- default value: 10
- Usage: Optional

## Unique Key Value (uniqueKeyValue)

Checks whether the key value is different across all components when a key exists in multiple components of a snapshot.

Results into a non-compliant status when a key has the same value in multiple components.

Input arguments

- keyList
  - Comma-separated list of key names
  - Default value:#None
  - Usage: Mandatory
- maxErrorDisplay
  - Maximum number of failures to display.
  - Default value: 10
  - Usage: Optional

## Unique Values across Deployables (uniqueKVcrossDeployables)

Checks that the corresponding value is different across all mapped deployables. A failure is triggered in case the keyName is not found in the primary deployable. A warning is triggered in case the keyName is not found in one of the additional deployables.

This policy accepts a comma-separated list of keyNames.

Use Case: Ensure that key settings such as database connection strings, usernames, API keys, for example, are different between each of the environments. Often configuration data is copied and an error is quickly made due to which a service in production now at once runs with the settings of the test environment (or vice versa).

Input arguments

- List of keys that should have different values: string JSON notation.
- List of extra deployables that should be compared to main one.

Error handling

Check that the input is a valid JSON object.

## Unresolved Variables (unresolvedVariables)

Checks if there are any unresolved variables. This policy loops over all CDI values in the snapshot.

Use Case: Ensure that all variables used in the configuration data settings are replaced with a value. Unresolved variables will typically break the deployment operation and this policy prevents this from happening.

## Map policies to a deployable

Map policies to a deployable to define the validation processes that the config data must pass. Whenever changes that affect a deployable are committed, the system generates a new snapshot of the deployable and then executes all mapped policies against the snapshot. You can also manually execute a policy at any time (for example, to test policy operation).

### Before you begin

Role required: cdm\_policy\_editor or cdm\_editor or cdm\_admin

### Procedure

1. On the **Settings** tab for an application, select **Policy Mapping** and then select **Add**.

The Map Policies dialog box displays the list of available policies and the following information for each policy.

#### Policies list on the Map Policies dialog box

Policy name	The name of the policy to map to the deployable.
State	Only mapped policies that are <b>Active</b> will execute during snapshot validation.
Category	Identifier that enables policy grouping. For example, Production Policies.
Tags	Tags that you can associate with the policy to enable filtering.

2. Select deployable against which the policies should execute.

3. Select the policies in the **Policies** list.

You can map any number of policies.

4. Select **Map Policies**.

On the **Mapped Policies** page, the deployable is updated with the list of its mapped policies. The tab lists all deployables that have mapped policies. Policies are grouped by the deployable to which they are mapped.

**Note:** By default, deleted deployables are filtered out. See [Delete a deployable](#).

#### Columns on the Mapped Policies page

Field	Description
Policy	Name of the policy. Policies appear grouped under the deployable to which they are mapped.
Category	Identifier that enables policy grouping. For example, Production Policies.

Field	Description
Input	Policy settings that are configurable.
Input status	Validation state of the mapping inputs in the policy.  <b>i Note:</b> <ul style="list-style-type: none"> <li>◦ This information reflects the validity of the input settings (that is, the readiness of the policy to perform validation).</li> <li>◦ All mapping inputs are valid upon insertion. Inputs become invalid if the policy version is archived or deleted.</li> <li>◦ This value does not indicate whether the snapshot passed validation (that is, whether the snapshot is compliant).</li> </ul>
Mapping state	Only mapped policies that are <b>Active</b> will execute during snapshot validation.
Tags	Tags that are associated with the policy.

## Define exporters in DevOps Config

Define exporters in DevOps Config to export config data from all, or part, of the data model as input for further deployment or provisioning activities.

### Before you begin

Role required: sn\_devops\_config.admin

### Procedure

1. In the **Admin** view, click the **Exporters** tab.
2. Select a default exporter or click **New** to [create a custom exporter](#).
3. Enter the Exporter Name and Description and click **Confirm**.

### Default DevOps Config exporters

The DevOps Config Exporter content pack contains a set of default DevOps Config exporters of data that can be used as input for further deployment and provisioning activities.

DevOps Config exporters allow other tools to consume the data from deployable snapshots.

**i Note:** You cannot modify default exporters. However, you can make a copy of the exporter and customize your copy.

These exporters are contained in the DevOps Config Exporter content pack.

- returnAllData-now
- returnAllData\_noVars-now
- returnDatafornodeName-now
- returnDataForNodeNames-now
- returnDataForPath-now
- returnNodeListForLevel-now
- returnNodeListForPath-now
- returnValueForKeyAtnodeName-now
- returnValueForKeyPath-now
- returnValueForUniqueKeyName-now

## Return all data

Returns the full content of the snapshot without any filtering or modifications, including the var system folder.

**Note:** The exporter fails if the application/deployable is not in Active state (deleted).

Exporter name

**returnAllData-now**

Arguments

- appName - Application name
- deployableName - Deployable name
- requestedFormat - Requested format (json/yaml/xml/ini/raw)

Special logic

None.

Error handling

None.

## Return all data except vars

Returns all of the configuration data for the deployable, except deployable name and variables.

Response does not include:

- vars folder at the deployable level
- vars folder at each of the included collections
- Deployable name at the root level of the response

**Note:** This exporter does not work for deleted applications/deployables.

Exporter name

**returnAllData\_noVars-now**

Arguments

Arguments (can be provided on the command line, or entered interactively in run mode).

- appName - Application name
- deployableName - Deployable name
- requestedFormat - Requested format (json/yaml/xml/ini/raw)

Special logic

None.

Error handling

None.

## Return data for a node name

Returns the subset of the snapshot data for a given node name, which is provided as an argument. The argument value must be passed as string text.

Exporter name

**returnDataforNodeName-now**

Arguments

- appName - Application name
- deployableName - Deployable name
- requestedFormat - Requested format (json/yaml/xml/ini/raw)
- nodeName - Node name (string, in quotes)

Special logic

If nodeName is empty, all data is returned.

Error handling

- If the nodeName is not unique, multiple instances of nodeName found.
- If the nodeName is not found, node not found: <nodeName>.

## Return data for list of nodes

Returns the full data from the snapshot for a list of nodes. Same as Return data for a node name but returns a nested JSON with configuration data for a list of given node names (including any child nodes).

Exporter name

**returnDataForNodeNames-now**

Arguments

- appName - Application name
- deployableName - Deployable name
- requestedFormat - Requested format (json/yaml/xml/ini/raw)
- nodeNames - Node names (string, in quotes, comma-separated)

Special logic

If nodeNamesList is empty, returns all config data.

#### Error handling

None.

#### Response details

```
{"node1": {"contentKey": "contentValue"}, "node2": { "error": "nodeName not found"}},
```

#### Error handling

- In case the nodeName is not unique the exporter returns an error response stating “multiple instances of nodeName found” for that specific nodeName. Other nodeNames contain the data
- If a nodeName is not found it should contain error message for that node

## Return data for path

Returns all of the configuration data for a given node path in the snapshot.

#### Exporter name

**returnDataForPath-now**

#### Arguments

- appName - Application name
- deployableName - Deployable name
- requestedFormat - Requested format (json/yaml/xml/ini/raw)
- nodePath - Node path (string, in quotes)

#### Special logic

If nodePath is empty, return the whole content (similar to all config data).

#### Error handling

If nodePath is not found, the last node name that was not found is stated path not found: <nodeName>.

## Return node list for level

Returns a list of names of nodes that are children of root node at specified level (depth) in the snapshot. For example, level 1 is a direct child of root node, level 2 is a grandchild, etc.

#### Exporter name

**returnNodeListForLevel-now**

#### Arguments

- appName - Application name
- deployableName - Deployable name
- requestedFormat - Requested format (json/yaml/xml/ini/raw)
- ExcludeVarsNode [true | false] - Exclude the vars node from the result (true or false, default is true)
- nodeLevel - Level of the node (integer, default is 0)

### Special logic

If no level is specified, then exporter returns the value for level 0 (for example, the deployable root node name).

### Error handling

None.

### Response details

["node1", "node2", "node3"]

## Return node list for path

Returns the list of nodes for a given node path in the snapshot (not taking into account sub-nodes).

### Exporter name

**returnNodeListForPath-now**

### Arguments

- appName - Application name
- deployableName - Deployable name
- requestedFormat - Requested format (json/yaml/xml/ini/raw)
- ExcludeVarsNode [true | false] - Exclude the vars node from the result (true or false, default is true)
- nodePath - Path to follow with list of nodes separated by pathSeparator (string, in quotes)
- pathSeparator - Character to separate list of nodePaths (string, default is ',')

### Special logic

None.

### Error handling

None.

### Response details

["node1", "node2", "node3"]

## Return value for key within a node

Returns the value of a specific key that is part of a node in the snapshot. The key can either be directly defined to the node, or lower in the data model to one of the children of the node.

The difference between this exporter and `export value for unique keyName` is that the key name only needs to be unique within the subtree of the node.

Key/node combination is expected to be unique in the snapshot. If the key/node combination is found more than once, there is an error.

### Exporter name

**returnValueForKeyAtNodeName-now**

### Arguments

- appName - Application name
- deployableName - Deployable name
- requestedFormat - Requested format (json/yaml/xml/ini/raw)
- keyName - Key name (string, in quotes)
- nodeName - Node name (string, in quotes)

Special logic

None.

Error Handling

- If keyPath is not provided, no keyPath argument provided.
- If keyName nodeName combination is not found an empty response is returned.

## Return value for keyPath

Returns the value of a specific key in a specific path.

Exporter name

**returnValueForKeyPath-now**

Arguments

- appName - Application name
- deployableName - Deployable name
- requestedFormat - Requested format (json/yaml/xml/ini/raw)
- keyPath - List of node names with key name at the end separated by pathSeparator (string, in quotes)
- pathSeparator - Character to separate list of keyPaths (string, default is ',')

Special logic

None.

Error handling

- If the keyPath is not provided, no keyPath argument provided.
- If the keyPath is not found, states the last node name not found path not found: <path>/<nodeName>.
- If the keyPath is found and is a node (not a key), keyPath provided is a node and not a key.

## Return value for unique keyName

Returns the value of a specific key based on its name in the snapshot. Unlike `export value for key` within a node, the key is expected to be unique across the snapshot data model. Multiple keys are supported.

**i Note:** Formats xml and ini are not supported for this exporter.

Exporter name

**returnValueForUniqueKeyName-now**

## Arguments

- appName - Application name
- deployableName - Deployable name
- requestedFormat - Requested format (json/yaml/raw)
- keyName - Key name (data array)

## Special logic

If the key is present multiple times in the snapshot, the exporter returns the first value found (returns error).

## Error handling

- If the keyName is not provided, no keyName argument provided.
- If the key is not found, key not found: <keyName>.

**Configure your Azure or Jenkins pipeline in DevOps Config**

Configure your Azure DevOps or Jenkins pipeline with DevOps Config to interact with your data model.

These orchestration tool integrations are included with the DevOps Config base system.

- Azure DevOps
- Jenkins

**Required plugins**

## Azure DevOps

Use the ServiceNow DevOps extension for Azure DevOps on [Visual Studio Marketplace](#) to integrate your Azure pipeline with the ServiceNow DevOps Config application.

## Jenkins

Visit the Ancillary Software section on the [ServiceNow Store](#) website to download the Jenkins plugin for ServiceNow DevOps to integrate your Jenkins pipeline with the ServiceNow DevOps Config application.

**Run validation in DevOps Config**

Once you have installed and configured DevOps Config, validate the configuration and review the results.

**Before you begin**

Role required: sn\_devops\_config.admin

**About this task**

Having the policy validation action at the deployable level and the publish action at the snapshot level helps protect against non-compliant changes in your production environment.

You can also validate further by mapping additional policies to a deployable and validating against your latest snapshot.

## Procedure

1. Open your application and in the **Snapshots** tab, and select the snapshot created from the changeset created.
2. Click **Validate** and confirm.
3. Review compliance results.  
Any future snapshots generated are automatically validated against the configured policies.

## CDM system properties

CDM system properties

### CDM system properties

#### CDM system properties

Property	Description
sn_cdm.variable_regex	<p>The regex used to capture variables referenced in a CDI value. With this value, the syntax to use a variable is:</p> <pre>@@name@@</pre> <p>Default: @@{[^@].*?}@@</p>
sn_cdm.cdm_queue_request_expiry_time_ms	<p>The maximum time for a process to be available for processing before it is marked as expired.</p> <p>Default: 900000 ms (15 min)</p>
sn_cdm.validation_timeout_ms	<p>The maximum time limit for a validation of a snapshot.</p> <p>Default: 900000 ms (15 min)</p>
sn_cdm.default_node_identifier_keys	<p>The keys used to identify node objects within an array node.</p> <p>Default: (none)</p>
sn_cdm.execute_exporter_async_for_standard_request	<p>If set to <code>false</code>, exporter requests skip the request queue and their output is returned immediately. Changing this setting may decrease system performance if there are many export requests.</p> <p>Default: true</p>

**CDM system properties (continued)**

Property	Description
sn_cdm.exception_enabled	<p>If set to false, the control exception functionality is disabled regardless of other policy settings.</p> <p>Default: true</p>
sn_cdm.node_name_allowed_character_regex	<p>The regex used to validate the name of a node. Default allowed characters are:</p> <div style="border: 1px solid black; padding: 5px; width: fit-content;">           0-9, A-Z, a-z, _, -, ., %, \$,            whitespace, :, #         </div> <p>Default: [\w-.%:\$#]</p>
sn_cdm.app_name_allowed_character_regex	<p>The regex used to validate the name of an application. Default allowed characters are:</p> <div style="border: 1px solid black; padding: 5px; width: fit-content;">           0-9, A-Z, a-z, _, -, ., whitespace,             , :, [], (), %, \$, #         </div> <p>Default: [\w-.%\$#() : [\]]]</p>
sn_cdm.snapshot_name_allowed_character_regex	<p>The regex used to validate the name of a snapshot. Default allowed characters are:</p> <div style="border: 1px solid black; padding: 5px; width: fit-content;">           0-9, A-Z, a-z, _, -, ., %         </div> <p>Default: [\w-.%]</p>
sn_cdm.exporter_name_allowed_character_regex	<p>The regex used to validate the name of an exporter. Default allowed characters are:</p> <div style="border: 1px solid black; padding: 5px; width: fit-content;">           0-9, A-Z, a-z, _, -, ., %         </div> <p>Default: [\w-.%]</p>
sn_cdm.exporter_argument_name_allowed_character_regex	<p>The regex used to validate the name of an exporter argument. Default allowed characters are:</p> <div style="border: 1px solid black; padding: 5px; width: fit-content;">           0-9, A-Z, a-z, _         </div> <p>Default: [\w]</p>
sn_cdm.cdm_maximum_allowed_open_changesets	<p>The maximum allowed number of open changesets.</p> <p>Default: 10</p>

## CDM system properties (continued)

Property	Description
sn_cdm.preventative_duplicate_node_name_check	<p>If set to true, validating for node name duplication will check name usage across all other open changesets as well (as opposed to limiting the check within the changeset).</p> <p>Default: false</p>
sn_cdm.maximum_allowed_upload_file_size	<p>The maximum payload size for upload of config data via the REST API.</p> <p>Default: 2097152 bytes (2MB)</p>
sn_cdm.maximum_allowed_cdi_per_application	<p>The maximum number of CDIs allowed per application. If the number is exceeded, further upload of config data via the REST APIs is blocked.</p> <p>Increasing this limit may significantly decrease performance and break UI functionality. If you change the value, you must recalculate CDI usage data by running the following command:</p> <pre data-bbox="811 1050 1378 1155">new     sn_cdm.CdmApplicationManager().updateCdiCountOfApplications();</pre> <p>Default: 100000</p>
sn_cdm.maximum_allowed_cdi_per_deployable	<p>The maximum number of CDIs allowed per deployable. If the number is exceeded, further upload of config data via the REST APIs is blocked.</p> <p>Default: 10000</p>
sn_cdm.maximum_nodes_in_memory	<p>The maximum number of nodes that can be supported when using the <code>queryTree</code> method of <code>CdmQuery</code>.</p> <p>Default: 10000</p>
sn_cdm.reserved_node_names	<p>Names that cannot be used for nodes.</p> <p>Default: vars, collections, deployables, components</p>

## Integrating your Azure or Jenkins pipeline in DevOps Config

Integrate your pipeline with DevOps Config by using Azure DevOps pipeline tasks, and Jenkins pipeline actions to interact with your data model.

Validate application and infrastructure details against your organization policies, and commit config data the same way software code is committed to a code repository.

Configuration data is pulled in your CI/CD pipeline to deploy not just a code for an application, but all the related configuration with that code as well.

**i Note:** Pipeline integration requires plugin installation. See [Configure your Azure or Jenkins pipeline in DevOps Config](#) for installation requirements.

### Azure DevOps pipeline tasks

Use these tasks in your Azure DevOps pipeline to interact with the DevOps Config data model.

These tasks are provided to create a specific pipeline definition to achieve your goal.

- ServiceNow-DevOps-Config-Agent-Upload-Config  
Upload config data to a deployable in the data model via Agent Job.
- ServiceNow-DevOps-Config-Agent-Get-Snapshot  
Get the snapshots for an application.
- ServiceNow-DevOps-Config-Agent-Get-Snapshot-Name  
Extract the name of a snapshot.
- ServiceNow-DevOps-Config-Agent-Publish-Snapshot  
Publish a snapshot of your configuration data.
- ServiceNow-DevOps-Config-Agent-Export-Snapshot  
Export a subset of your configuration data.
- ServiceNow-DevOps-Config-Agent-Register-Pipeline  
Register a changeset and/or snapshot to a pipeline execution.
- ServiceNow-DevOps-Config-Agent-Validate-Snapshot  
Validate configuration data against organization policies.
- ServiceNow-DevOps-Server-Change-Acceleration  
Create a change request as part of the pipeline.

### ServiceNow-DevOps-Config-Agent-Upload-Config

Task to upload a configuration file to a given location within an application data model.

This task is meant to be used in an iterative nature for all config files the user chooses to upload to their application data model during the pipeline run.

**i Note:** Pre-uploaded files must be in a compatible format.

### Input variables

connectedServiceName	Specifies the DevOps pipeline endpoint connection.
applicationName	Specifies the application to where config data is uploaded.
deployableName	Specifies the deployable for the application (required if target is deployable).
uploadTarget	Specifies the data model target to where config data is uploaded (for example, component, collection, deployable).
collectionName	(Optional) Specifies the collection to where config data will be uploaded (required if target is collection).
namePath	Specifies the data model path to where config data is uploaded.  <b>i Note:</b> When uploading to a vars folder, you must start the name path with "vars/" to specify the variable folder path.
configFilePath	Specifies the source folder from which config data is uploaded to the component or deployable path in the data model.  Empty is the root of the repo. Use variables if files are not in the repo (for example, \$(agent.builddirectory)).
convertPath	(Optional) Specifies whether to preserve the directory structure of the configuration files (with respect to the workspace), and convert the directory to paths within the data model. Default is false.
dataFormat	Specifies the data format of the config_file (for example, JSON, YAML, XML, etc.).
changesetNumber	(Optional) Specifies the (open) changeset to which this upload activity is associated. If not provided, a new changeset is created.

**Note:** Only used for multiple upload scenarios.

autocommit	Specifies whether to commit configuration data after upload (true/false). Default is false.
autoValidate	Specifies whether to validate configuration data during commit (true/false). Default is true.

#### Output variable

changesetNumber	Changeset record created/committed during the upload.
	Provide a name to the task so it can be used later in the pipeline (per example, componentUpload).

#### Example - Upload config

```
-task: ServiceNow-DevOps-Config-Agent-Upload-Config
  name: componentUpload
  inputs:
    connectedServiceName: 'MyServiceNowInstance'
    applicationName: 'PaymentDemo'
    uploadTarget: 'component'
    namePath: 'wep-api-v1.0'
    configFilePath: 'k8s/helm/values.yml'
    dataFormat: 'yaml'
    autoCommit: true
    autoValidate: true
```

#### Example - Multiple uploads (component)

You can call the upload task more than once to upload configuration data in different file formats from different locations, while still keeping the uploads part of one changeset.

- In the first upload, name the task so the changesetNumber output variable can be reused in subsequent uploads.

YAML file upload:

```
-task: ServiceNow-DevOps-Config-Agent-Upload-Config
  name: componentUpload
  inputs:
    connectedServiceName: 'MyServiceNowInstance'
    applicationName: 'PaymentDemo'
    uploadTarget: 'component'
    namePath: 'wep-api-v1.0'
    configFilePath: 'k8s/helm/values.yml'
    dataFormat: 'yaml'
```

```
autoCommit: false
autoValidate: false
```

- In subsequent uploads, reference the changesetNumber output variable from the first upload as an input variable.

JSON file upload:

```
-task: ServiceNow-DevOps-Config-Agent-Upload-Config
  inputs:
    connectedServiceName: 'MyServiceNowInstance'
    applicationName: 'PaymentDemo'
    uploadTarget: 'component'
    namePath: 'wep-api-v1.0'
    configFilePath: 'featureToggles/set1.json'
    dataFormat: 'json'
    autoCommit: true
    autoValidate: true
    changesetNumber: '$(componentUpload.changesetNumber)'
```

#### Example - Multiple uploads (collection and vars)

You can call the upload task more than once to upload configuration data in different file formats from different locations, while still keeping the uploads part of one changeset.

- In the first upload, make sure to name the task so the changesetNumber output variable can be reused in subsequent uploads.

XML file upload:

```
-task: ServiceNow-DevOps-Config-Agent-Upload-Config
  name: componentUpload
  inputs:
    connectedServiceName: 'MyServiceNowInstance'
    applicationName: 'PaymentDemo'
    uploadTarget: 'collection'
    collectionName: 'release-1.0'
    namePath: 'v1-common-configs'
    configFilePath: 'infra/v1/config.xml'
    dataFormat: 'xml'
    autoCommit: false
    autoValidate: false
```

- In subsequent uploads, reference the changesetNumber output variable from the first upload as an input.

JSON file upload:

```
-task: ServiceNow-DevOps-Config-Agent-Upload-Config
  inputs:
    connectedServiceName: 'MyServiceNowInstance'
    applicationName: 'PaymentDemo'
    uploadTarget: 'deployable'
    deployableName: 'Production-EMEA'
    namePath: 'vars/dbSettings'
```

```

configFilePath: 'infra/prodc/dbSettings.json'
dataFormat: 'json'
autoCommit: true
autoValidate: true
changesetNumber: '$(componentUpload.changesetNumber)'

```

**Note:** To upload to a variable folder, uploadTarget must be set to deployable, and the correct values must be set for deployableName and changesetNumber.

## ServiceNow-DevOps-Config-Agent-Get-Snapshot

This task is intended to be used in different scenarios:

- Retrieve a specific snapshot.

Following the CD flow, a specific snapshot is retrieved so it can be published and then exported to be consumed downstream (for example, to provision out infrastructure or application).

- Retrieve latest validated snapshot.

The latest validated snapshot is retrieved for the given application-deployable combination.

- Retrieve all snapshots for any impacted deployables.

When config files are uploaded to an application data model, the system will create snapshots for any deployables determined to be impacted by the upload. Following along the CI flow, assuming the last Upload call had validation enabled, the next step would be to iterate through the list of snapshots and ensure they all passed validation.

- Show policy validation results in a pipeline execution.

View policy validation results as test results on the ADO build tests results page, including compliant with exception, when getting a snapshot.

### Input variables

connectedServiceName	Specifies the DevOps pipeline endpoint connection.
applicationName	Specifies the application to validate.
deployableName	(Optional) Specifies the deployable (per the specified application) on which to get the latest snapshot data.
changesetNumber	(Optional) Specifies the changeset ID for the applicable set of config changes.
isValidated	(Optional) Specifies to only retrieve the latest validated snapshots.

### Output variable

<b>snapshotObject</b>	<p>A JSON object containing the requested snapshots.</p> <p>Provide a name to the task so it can be used later in the pipeline (per example, <code>getSnapshot</code>).</p>
-----------------------	---

#### Example - Specific snapshot

```
-task: ServiceNow-DevOps-Config-Agent-Get-Snapshot
  name: getSnapshot
  inputs:
    connectedServiceName: 'MyServiceNowInstance'
    applicationName: 'PaymentDemo'
    deployableName: 'Production'
    changesetNumber: 'Chset-16'
```

#### Example - Latest validated snapshot

```
-task: ServiceNow-DevOps-Config-Agent-Get-Snapshot
  name: getSnapshot
  inputs:
    connectedServiceName: 'MyServiceNowInstance'
    applicationName: 'PaymentDemo'
    deployableName: 'Production'
    isValidated: 'true'
```

#### Example - All changeset snapshots

```
-task: ServiceNow-DevOps-Config-Agent-Get-Snapshot
  name: getSnapshot
  inputs:
    connectedServiceName: 'MyServiceNowInstance'
    applicationName: 'PaymentDemo'
    changesetNumber: 'Chset-16'
```

#### Example - Show policy validation results

Assign a variable to the path of the file that contains the snapshot validation results generated during the `ServiceNow-DevOps-Config-Agent-Get-Snapshot` task.

To load the snapshot validation results in your pipeline execution, you need to leverage the ADO native [Publish Test Results v2 task](#), using the variable as an input.

```
stages:
- stage: Two
  jobs:
    - job: A
variables:
  - name: validationResultsPath
```

```

    value:
    1/TEST_DATA_${Build.DefinitionName}_${Build.BuildNumber}/*.xml
    steps:
      - task: PublishTestResults@2
        inputs:
          testResultsFormat: 'JUnit'
          testResultsFiles: '${validationResultsPath}'
          searchFolder: '${System.WorkFolder}'

```

## ServiceNow-DevOps-Config-Agent-Get-Snapshot-Name

This task is used as a follow-up from the ServiceNow-DevOps-Config-Agent-Get-Snapshot-Name task to get the snapshot name from a particular snapshot. From here, the snapshot name can be used as an input to a downstream task, like publishing the snapshot.

### Input variables

deployableName	Specifies the deployable to get the snapshot object that was returned from the ServiceNow-DevOps-Config-Agent-Get-Snapshot task.
script	Specifies the script to extract the snapshot name from the snapshot object.

### Output variable

snapshotName	<p>The snapshot name for the deployable.</p> <p>Provide a name to the task so it can be used later in the pipeline (per example, <code>getSnapshotName</code>).</p>
--------------	---

### Example - Get snapshot name

Use this script to extract the snapshot name retrieved from the ServiceNow-DevOps-Config-Agent-Get-Snapshot task.

```

-task: ServiceNow-DevOps-Config-Agent-Get-Snapshot-Name
inputs:
  deployableName: 'PRD'
  script: |
function run() {
  let name;
  let deployableName = process.argv[2];
  let jsonObj = ${getSnapshot.snapshotObjects};
  let size = jsonObj.result.length;
  for(let i=0; i<size; i++) {
    obj = jsonObj.result[i];
    if(obj["deployable_id.name"].toLowerCase() ==
deployableName) {
      name = obj.name;
    }
}

```

```

        console.log(name);
    }
}
run();

```

Example - Get snapshot name used with get snapshot

Call the ServiceNow-DevOps-Config-Agent-Get-Snapshot-Name task right after the ServiceNow-DevOps-Config-Agent-Get-Snapshot task to return the snapshot name.

Get snapshots (returns a JSON object containing impacted snapshots):

```

-stage: Two
jobs:
  -job: B
  steps:
    -task: ServiceNow-DevOps-Config-Agent-Get-Snapshot
      name: getSnapshot
      inputs:
        connectedServiceName: 'MyServiceNowInstance'
        applicationName: 'PaymentDemo'
        deployableName: 'Production-2'
        changesetNumber: 'Chset-16'

```

Get snapshot name (returns snapshot name for a specific deployable):

```

-stage: Two
jobs:
  - job: B
  steps:
    - task: ServiceNow-DevOps-Config-Agent-Get-Snapshot
      name: getSnapshotName
      inputs:
        deployableName: 'Production-2'
        script:
          function run() {
            let name;
            let deployableName = process.argv[2];
            let jsonObj = $(getSnapshot.snapshotObjects);
            let size = jsonObj.result.length;
            for(let i=0; i<size ;i++) {
              obj = jsonObj.result[i];
              if(obj["deployable_id.name"].toLowerCase() ==
deployableName) {
                name = obj.name;
                console.log(name); // This standard output of
inline script is given as the task output
              }
            }
          }
        run();

```

Use returned snapshot name in the pipeline (for example, publish):

```

-stage: Three
  jobs:
    -job: C
      variables:
        varSnapshotName:
          $[stageDependencies.Two.B.outputs['getSnapshotName.snapshotName']]
      steps:
        -task: ServiceNow-DevOps-Config-Agent-Publish-Snapshot
          inputs:
            connectedServiceName: 'MyServiceNowInstance'
            applicationName: 'PaymentDemo'
            deployableName: 'Production-2'
            snapshotName: '${varSnapshotName}'

```

## ServiceNow-DevOps-Config-Agent-Publish-Snapshot

This task publishes a snapshot for the given application and deployable. From here, the snapshot can be consumed through the Export process.

### Input variables

connectedServiceName	Specifies the ServiceNow endpoint connection.
applicationName	Specifies the application to publish.
deployableName	Specifies the deployable for the application to publish config data.
snapshotName	Specifies the name of the snapshot to publish.

### Output variable

Not applicable (returns true if successful, otherwise false).

### Example

```

-task: ServiceNow-DevOps-Config-Agent-Publish-Snapshot
  inputs:
    connectedServiceName: 'MyServiceNowInstance'
    applicationName: 'PaymentDemo'
    deployableName: 'Production-2'
    snapshotName: 'Production-v23.dpl'

```

## ServiceNow-DevOps-Config-Agent-Export-Snapshot

This task exports a snapshot for the given application and deployable. Specify the exporter, relevant exporter arguments, the export format (for example, YAML, JSON, etc.), and output location for the exported config data. From here, the config data can be used directly as an input for a deployment or provisioning tool downstream in the pipeline.

### Input variables

connectedServiceName	Specifies the ServiceNow endpoint connection.
----------------------	---

applicationName	Specifies the application from which to publish.
deployableName	Specifies the deployable for the application from which to export config data.
snapshotName	Specifies the name of the snapshot from which to export config data.
exporterName	Specifies the exporter to apply to the snapshot (for example, UniqueCDIs).
args	(Optional) Specifies arguments to be used along with the exporter.
exportFormat	Specifies the format to export the snapshot data (for example, INI, YAML, PROPS).
saveFile	<p>Checks if file is to be saved to an Azure repository (true/false). Default is false.</p> <p><b>i Note:</b> Appropriate permission/OAuth token access to script is required.</p> <p>Otherwise the export file is created in the pipeline workspace directory.</p>

#### Output variable

Not applicable (returns true if successful, otherwise false).

#### Example

```
-task: ServiceNow-DevOps-Config-Agent-Export-Snapshot
inputs:
  connectedServiceName: 'MyServiceNowInstance'
  applicationName: 'PaymentDemo'
  deployableName: 'Production-2'
  exporterName: 'returnAllData-nowPreview'
  dataFormat: 'yaml'
  args: ''
  snapshotName: 'Production-v23.dpl'
  saveFile: true
  fileName:
    'ExporterOutput/ExportData_${(build.definitionName)}_${(build.buildNumber)}.yaml'
```

## ServiceNow-DevOps-Config-Agent-Register-Pipeline

This task ties a changeset and a snapshot to the pipeline so that it can be tracked during the pipeline execution. In DevOps Change Velocity, this is shown in the Pipeline UI.

#### Input variables

connectedServiceName	Specifies the DevOps pipeline endpoint connection.
applicationName	Specifies the application name.
changesetNumber	(Optional) Specifies the ID of the changeset to associate to the pipeline execution.
snapshotName	(Optional) Specifies the name of the snapshot to associate to the pipeline execution.

#### Output variable

Not applicable (returns true if successful, otherwise false).

#### Example

```
-task: ServiceNow-DevOps-Config-Agent-Publish-Snapshot
  inputs:
    connectedServiceName: 'MyServiceNowInstance'
    changesetNumber: 'Changeset-143'
```

## ServiceNow-DevOps-Config-Agent-Validate-Snapshot

Validate config data against your organization policies.

#### Input variables

connectedServiceName	Specifies the ServiceNow endpoint connection.
applicationName	Specifies the application to validate.
deployableName	Specifies the deployable (per the specified application) to validate.
snapshotName	(Optional) Specifies the name of the snapshot to validate.
showResults	(Optional) Specifies to show validation results in the console log.

#### Output variable

Not applicable (returns true if successful, otherwise false).

#### Example

```
-task: ServiceNow-DevOps-Config-Agent-Validate-Snapshot
  inputs:
    connectedServiceName: 'MyServiceNowInstance'
    applicationName: 'PaymentDemo'
    deployableName: 'Production-2'
    snapshotName: ''
    showResults: false
```

## ServiceNow-DevOps-Server-Change-Acceleration

This task is required for agentless (server) jobs to automatically create a change request in ServiceNow Change Management as part of the Azure DevOps pipeline.

In DevOps Config, to associate multiple snapshots of the same changeset to a change request, use snapshot name and application name to track specific configuration data for a given application service.

See [Accelerating your DevOps change process](#) for more information regarding the DevOps Change Acceleration feature.

Input variables (DevOps Config-related)

connectedServiceName	Specifies the DevOps pipeline endpoint connection.
applicationName	Application associated with the snapshot being attached to the change request.
snapshotName	Name of the snapshot to attach to the change request.

Example

```
-stage: ChangeRequest
  jobs:
    -job: 'changerequestjob'
      pool: server
      steps:
        -task: ServiceNow-DevOps-Server-Change-Acceleration
          inputs:
            connectedServiceName: 'MyServiceNowInstance'
            applicationName: 'PaymentDemo'
            snapshotName: 'Production-v23.dp1'
```

## YAML pipeline example

```
trigger:
  branches:
    include:
    - none
stages:
- stage: One
  displayName: Upload Configuration Data
  pool:
    vmImage: ubuntu-latest
  jobs:
    - job: A
      displayName: Upload
      steps:
        - task: ServiceNow-DevOps-Config-Agent-Upload-Config@1
          name: componentUpload
          inputs:
            connectedServiceName: 'MyServiceNowInstance'
            applicationName: 'PaymentDemo'
```

```

uploadTarget: 'component'
configFile: 'k8s/helm/values.yml'
namePath: 'processor-api-v1.0'
dataFormat: 'yaml'
autoValidate: true
autoCommit: true
convertPath: true
- stage: Two
  displayName: Get Latest Snapshot
  pool:
    vmImage: ubuntu-latest
  jobs:
    - job: B
      displayName: Get Snapshot
      variables:
        - name: varChangestNumber
          value:
            $[stageDependencies.One.A.outputs['componentUpload.changestNumber']]
      - name: varConfigValidationResults
        value:
          1/TEST_DATA_${Build.DefinitionName}_${Build.BuildNumber}_*.xml
      steps:
        - task: ServiceNow-DevOps-Config-Agent-Get-Snapshot@1
          name: getSnapshot
          inputs:
            connectedServiceName: 'MyServiceNowInstance'
            applicationName: 'PaymentDemo'
            deployableName: 'Production-EMEA'
            changeSetNumber: '$(varChangestNumber)'
        - task: ServiceNow-DevOps-Config-Agent-Get-Snapshot-Name@1
          name: getSnapshotName
          inputs:
            deployableName: 'Production-EMEA'
            script: "function run() {\n  let name;\n  let\n  deployableName = process.argv[2];\n  let jsonObj =\n  $(getSnapshot.snapshotObjects);\n  let size = jsonObj.result.length;\n  for(let i=0; i<size ;i++) {\n    obj = jsonObj.result[i];\n    if(obj['deployable_id.name'].toLowerCase() == deployableName) {\n      name = obj.name;\n      console.log(name); // This standard output of\n      inline script is given as the task output\n    }\n  }\n}\nrun();\n"
        - task: PublishTestResults@2
          inputs:
            testResultsFormat: 'JUnit'
            testResultsFiles: '$(varConfigValidationResults)'
            searchFolder: '$(System.WorkFolder)'
        - task: ServiceNow-DevOps-Config-Agent-Register-Pipeline@1
          inputs:
            connectedServiceName: 'MyServiceNowInstance'
            snapshotName: '$(getSnapshotName.snapshotName)'
            applicationName: 'PaymentDemo'
- stage: Three
  displayName: Publish Snapshot
  pool:
    vmImage: ubuntu-latest
  jobs:
    - job: C
      displayName: Publish

```

```

variables:
- name: varSnapshotName
  value:
$[stageDependencies.Two.B.outputs['getSnapshotName.snapshotName']]
steps:
- task: ServiceNow-DevOps-Config-Agent-Publish-Snapshot@1
  inputs:
    connectedServiceName: 'MyServiceNowInstance'
    applicationName: 'PaymentDemo'
    deployableName: 'Production-EMEA'
    snapshotName: '$(varSnapshotName)'
- stage: ChangeRequest
dependsOn:
- Two
- Three
jobs:
- job: 'changerequestjob'
  timeoutInMinutes: 2
pool:
  name: server
variables:
- name: varSnapshotName
  value:
$[stageDependencies.Two.B.outputs['getSnapshotName.snapshotName']]
steps:
- task: ServiceNow-DevOps-Server-Change-Acceleration@1
  inputs:
    connectedServiceName: 'MyServiceNowInstance'
    applicationName: 'PaymentDemo'
    snapshotName: '$(varSnapshotName)'
- stage: Four
displayName: Export Snapshot
dependsOn:
- Two
- Three
- ChangeRequest
pool:
  vmImage: ubuntu-latest
variables:
- name: varSnapshotName
  value:
$[stageDependencies.Two.B.outputs['getSnapshotName.snapshotName']]
jobs:
- job: D
  displayName: Export
  steps:
- task: ServiceNow-DevOps-Config-Agent-Export-Snapshot@1
  inputs:
    connectedServiceName: 'MyServiceNowInstance'
    applicationName: 'PaymentDemo'
    deployableName: 'Production-EMEA'
    exporterName: 'returnAllData-now'
    dataFormat: 'yaml'
    snapshotName: '$(varSnapshotName)'
    saveFile: true
    fileName:
      'ExportData_$(build.definitionName)_$(build.buildNumber).yaml'

```

```

- task: CmdLine@2
  inputs:
    script: |
      echo Write your commands here
      echo Hello world
      tree $(Pipeline.Workspace)

```

## Jenkins pipeline actions

Use these actions in your Jenkins pipeline to interact with the DevOps Config data model.

Jenkins scripted and declarative pipelines are supported.

These actions are provided to create a specific pipeline definition to achieve your goal. Add a command to your Jenkins file to perform these actions.

- **snDevOpsConfigUpload**

Upload config data to DevOps Config via Agent Job.

- **snDevOpsConfigGetSnapshots**

Retrieve snapshots for a specific deployable, or all impacted deployables.

- **snDevOpsConfigPublish**

Publish a snapshot for the given application and deployable.

- **snDevOpsConfigExport**

Export a snapshot for a given application and deployable.

- **snDevOpsConfigRegisterPipeline**

Tie a changeset and/or snapshot to a pipeline execution.

- **snDevOpsConfigValidate**

Validate config data against your organization policies.

- **snDevOpsChange**

Create a change request with associated snapshot attached.

## **snDevOpsConfigUpload**

This action uploads a configuration file to a given location within an application data model.

It is meant to be used in an iterative nature for all config files to upload to the application data model during the pipeline run.

Supports:

- Upload to:
  - A component, collection, or deployable.
  - The variable (vars) folder of a component, collection, or deployable.
- Regex pattern for config file input.
- Ability to be called multiple times in the same pipeline.

## Input variables

configFile	Specifies the configuration data file to upload to the component or deployable path in the data model.
applicationName	Specifies the application to where config data will be uploaded.
target	Specifies the data model target to where config data will be uploaded (for example, component, collection, deployable).
collectionName	(Optional) Name of the collection to upload to (required if target is collection).
deployableName	Name of the deployable to upload to (required if target is deployable).
namePath	Specifies the name path in the data model to where config data will be uploaded.  <b>i Note:</b> When uploading to a vars folder, you must start the name path with "vars/" to specify the variable folder path.
dataFormat	Specifies the data format of the config file (for example, JSON, YAML, XML, etc.)
convertPath	(Optional) Specifies whether to preserve the directory structure of configuration files (with respect to the workspace) and convert the directory to paths within the data model.
changesetNumber	(Optional) Specifies the (open) changeset to which this upload activity is associated. If not provided, a new changeset is created.  <b>i Note:</b> Only used for multiple upload scenarios.
autoCommit	Specifies whether to commit configuration data after upload (true/false). Default is false.
autoValidate	Specifies whether to validate configuration data during commit (true/false). Default is false.

## Output variable

changesetNumber	<p>(Optional) Specifies the (open) changeset to which this upload activity is associated.</p> <p>If a changeset number is not provided, a new changeset is created.</p>
-----------------	---

## Example

- Input:

Here is an example of the `snDevOpsConfigUpload` action. For the sake of illustration, we'll assign the response to a variable, `changeSetId`, which could be echoed out to our console log for debugging scenarios.

```
changeSetId = snDevOpsConfigUpload(
    applicationName: "PaymentDemo",
    target: 'component',
    namePath: "web-api-v1.0",
    configFile: "k8s/helm/values.yml",
    dataFormat: "json",
    autoCommit: 'true',
    autoValidate: 'true'
)

echo "Changeset: $changeSetId created"
```

- Output:

In addition to the data being uploaded to our data model in DevOps Config, the output would look something like this (using the Blue Ocean plugin to visualize the console output).

## Example - Multiple uploads (component)

You can call the upload action more than once to upload configuration data in different file formats from different locations, while still keeping the uploads part of one changeset.

- In the first upload, name the action so the `changesetNumber` output variable can be reused in subsequent uploads.

YAML file upload:

```
$changeset = snDevOpsConfigUpload(
    applicationName: 'PaymentDemo',
    target: 'component',
    namePath: 'wep-api-v1.0',
    configFile: 'k8s/helm/values.yml',
    dataFormat: 'yaml',
    autoCommit: 'false',
```

```

        autoValidate: 'false'
    )

```

- In subsequent uploads, reference the changesetNumber output variable from the first upload as an input variable.

3 JSON files upload:

```

snDevOpsConfigUpload(
    applicationName: 'PaymentDemo',
    target: 'component',
    namePath: 'wep-api-v1.0',
    configFile: 'infra/*.json',
    dataFormat: 'json',
    autoCommit: 'false',
    autoValidate: 'false',
    changesetNumber: "${changeset}"
)

```

- In the final call, in addition to referencing the changesetNumber output variable from the first upload as an input variable, set autoCommit and autoValidate to true.

INI file upload:

```

snDevOpsConfigUpload(
    applicationName: 'PaymentDemo',
    target: 'component',
    namePath: 'wep-api-v1.0',
    configFile: 'featureToggles/set1.ini',
    dataFormat: 'ini',
    autoCommit: 'true',
    autoValidate: 'true',
    changesetNumber: "${changeset}"
)

```

#### Example - Multiple uploads (collection and vars)

You can call the upload action more than once to upload configuration data in different file formats from different locations, while still keeping the uploads part of one changeset.

- In the first upload, create a variable (for example, \$changeset), and assign the return value of the step to it so it can be reused in subsequent uploads.

XML file upload:

```

$changeset = snDevOpsConfigUpload(
    applicationName: 'PaymentDemo',
    target: 'collection',
    collectionName: 'release-v1.0',
    namePath: 'v1-common-configs',
    configFile: 'infra/v1/config.xml',
    dataFormat: 'xml',
    autoCommit: 'false',

```

```

        autoValidate: 'false'
)

```

- In subsequent uploads, use the variable as an input.

JSON file upload:

```

snDevOpsConfigUpload(
    applicationName: 'PaymentDemo',
    target: 'deployable',
    deployableName: 'Production',
    namePath: 'vars/dbSettings',
    configFile: 'infra/prod/dbConfig.json',
    dataFormat: 'json',
    autoCommit: 'true',
    autoValidate: 'true',
    changesetNumber: "${changeset}"
)

```

**i Note:** To upload to a variable folder, uploadTarget must be set to deployable, and the correct values must be set for deployableName and changesetNumber.

## snDevOpsConfigGetSnapshots

This action is intended to be used in different scenarios:

- Retrieve all snapshots for any impacted deployables.

When config files are uploaded to an application data model, the system will create snapshots for any deployables determined to be impacted by the upload. Following along the CI flow, assuming the last Upload call had validation enabled, the next step would be to iterate through the list of snapshots and ensure they all passed validation.

- Retrieve a specific snapshot.

Following the CD flow, a specific snapshot is retrieved so it can be published and then exported to be consumed downstream (for example, to provision out infrastructure or application).

- Show policy validation results in a pipeline execution.

View policy validation results as test results on the Jenkins build tests results page, including compliant with exception, when getting a snapshot.

Input definitions

applicationName	Specifies the application.
deployableName	(Optional) Specifies the deployable for the application on which to get the latest snapshot data.
changesetNumber	(Optional) Specifies the changeset ID for the set of config changes the user is interested in.

isValidated	(Optional) Specifies to only retrieve the latest validated snapshots, with or without exceptions.
-------------	---

## Output

- If successful, a snapshot or set of snapshots.
- If failure, an API/backend failure message is shown.

## Example

- Specific snapshot (specified):

```
$snapshots = snDevOpsConfigGetSnapshots(
    applicationName: 'PaymentDemo',
    deployableName: 'Production',
    changesetNumber: 'Chset-16'
)
```

- Latest validated snapshot (returns the latest snapshot for application and deployable combination):

```
$snapshots = snDevOpsConfigGetSnapshots(
    applicationName: 'PaymentDemo',
    deployableName: 'Production',
    isValidated: 'true'
)
```

- All changeset snapshots (returns all snapshots for application and deployable combination):

```
$snapshots = snDevOpsConfigGetSnapshots(
    applicationName: 'PaymentDemo',
    changesetNumber: 'Chset-16',
)
```

- Show policy validation results in a pipeline execution.

1. Assign a variable to the path of the file that contains the snapshot validation results generated during the `snDevOpsConfigGetSnapshots` action.
2. Call the [JUnit action](#) ↗ to load the snapshot validation results into the pipeline execution test section.

```
stage('Validate') {
    steps {
        script {
            changeSetResults = snDevOpsConfigGetSnapshots( ... )
            if (!changeSetResults) {
                echo "No snapshots were created"
            } else {
                def changeSetResultsObject = readJSON text:
                changeSetResults
            }
        }
    }
}
```

```

        changeSetResultsObject.each {
            snapshotName = it.name
            snapshotObject = it
        }
        // STEP 1
        validationResultsPath =
        "${snapshotName}_${currentBuild.projectName}_${currentBuild.number}.xml"
    }
}
}

post {
    always {
        // STEP 2
        junit testResults: "${validationResultsPath}",
        skipPublishingChecks: true
    }
}

```

## snDevOpsConfigPublish

This action publishes a snapshot for the given application and deployable. From here, the snapshot can be consumed through the Export process.

### Input definitions

applicationName	Specifies the application from which to publish config data.
deployableName	Specifies the deployable for the application from which to publish config data.
snapshotName	Specifies the name of the snapshot to publish.

### Output

- If successful, true.
- Otherwise false.

### Example

```

snDevOpsConfigPublish(
    applicationName: 'PaymentDemo',
    deployableName: 'Production',
    snapshotName: 'Production-v23.dpl',
)

```

## snDevOpsConfigExport

This action exports a snapshot for the given application and deployable.

The user should specify the exporter, relevant exporter arguments, the export format (for example, YAML, JSON, etc.), and output location for the exported config data.

From here, the config data can be used directly as an input for a deployment or provisioning tool downstream in the pipeline.

#### Input arguments

applicationName	Specifies the application from which to export data.
deployableName	Specifies the config deployable for the application from which to export data.
snapshotName	(Optional) Specifies snapshot from which to export data.  If a snapshot is not specified, the latest snapshot for the deployable is used.
exporterName	Specifies the exporter to apply to the snapshot (for example, UniqueCDIs).
exporterArgs	(Optional) Specifies arguments to be used along with the exporter.
exportFormat	Specifies the format to export the snapshot data (For example, INI, YAML, PROPS).
fileName	Specifies the file to export data to (assumed to be in the workspace).  If a filename is not specified, a concatenation of application name and deployable name (plus file extension) is used by default.

#### Output

- If successful, true.
- Otherwise false.

#### Example

```
snDevOpsConfigExport(
    applicationName: 'PaymentDemo',
    deployableName: 'Production',
    snapshotName: 'Production-v23.dpl',
    exporterFormat: 'yaml',
    exporterName: 'returnAllData-now',
    exporterArgs: '',
    fileName: 'exported_file-Production-20220302.yml'
)
```

## snDevOpsConfigRegisterPipeline

This action ties a changeset and/or snapshot to the pipeline so that it can be tracked during the pipeline execution. In DevOps Change Velocity, this is shown in the Pipeline UI.

See [Accelerating your DevOps change process](#) for more information regarding the DevOps Change Acceleration feature.

### Input arguments

applicationName	Specifies the name of the application.
changesetNumber	(Optional) Specifies the changeset to associate with the pipeline execution.  <b>i Note:</b> Specify either changesetNumber or snapshotName, but not both.
snapshotName	(Optional) Specifies the name of the snapshot to associate to the pipeline execution.  <b>i Note:</b> Specify either changesetNumber or snapshotName, but not both.

### Output

- If successful, true.
- Otherwise false.

### Example

- Input:

Here is an example of the `snDevOpsConfigRegisterPipeline` action. For the sake of illustration, we'll assign the response to a variable, `changeSetRegResult`, which could be echoed out to our console log for debugging scenarios.

```
changeSetRegResult = snDevOpsConfigRegisterPipeline(
    applicationName: "PaymentDemo",
    changesetNumber: "Chset-122"
)
echo "Pipeline registration result: ${changeSetRegResult}"
```

- Output:

In addition to the data being uploaded to our data model in DevOps Config, the output would look something like this (using the Blue Ocean plugin to visualize the console output).

## snDevOpsConfigValidate

Validate config data against your organization policies.

### Input arguments

applicationName	Application to validate.
deployableName	Deployable for the application to validate.
snapshotName	(Optional) Name of the snapshot to validate.
markFailed	(Optional) Fail the pipeline in the event that the validation attempt failed (due to a backend issue).
showResults	(Optional) Show validation results in the Jenkins job console log.

### Output

- If successful, no output.
- If failure, an API/backend failure message is shown.

### Example

- Specific snapshot (specified):

```
snDevOpsConfigValidate(
    applicationName: 'PaymentDemo',
    deployableName: 'Production',
    snapshotName: 'Production-v23.dpl',
)
```

- Latest snapshot (retrieves and validates the latest snapshot for application and deployable combination):

```
$changeset = snDevOpsConfigValidate(
    applicationName: 'PaymentDemo',
    deployableName: 'Production'
)
```

## snDevOpsChange

Create a change request and attach a snapshot for reference.

See [Accelerating your DevOps change process](#) for more information regarding the DevOps Change Acceleration feature.

#### Input arguments

applicationName	Specifies the name of the application.
snapshotName	Specifies the name of the snapshot to associate with the change request.

#### Example

```
snDevOpsChange(
    applicationName: 'PaymentDemo',
    snapshotName: 'Production-v23.dpl'
)
```

### Jenkins pipeline example

```
pipeline {
    environment {
        buildArtifactsPath = "build_artifacts/${currentBuild.number}"
        validationResultsPath = ""
    }

    agent any

    stages {
        // Initialize pipeline
        stage('Initialize') {
            steps {
                script {
                    // DevOps Config application related information
                    appName = 'PaymentDemo'
                    deployableName = 'Production'
                    componentName = "web-api-v1.0"
                    collectionName = "release-1.0"
                    // Configuration file information
                    exportFormat = 'yaml'
                    configFilePath = "k8s/helm/values.yml"
                    // Exporter related information
                    exporterName = 'returnAllData-nowPreview'
                    exporterArgs = ''
                    // Jenkins variables declared to be used in pipeline
                    exportFileName =
                    "${buildArtifactsPath}/export_file-${appName}-${deployableName}-${currentBuild.number}.${exportFormat}"
                    changeSetId = ""
                    snapshotName = ""
                    snapshotObject = ""
                    isSnapshotValidationRequired = false
                    isSnapshotPublishingRequired = false
                }
            }
        }
    }
}
```

```

// Validate configuration data changes
stage('Validate') {
    parallel {
        stage('Config') {
            stages('Config Steps') {
                // Upload configuration data to DevOps Config
                stage('Upload') {
                    steps {
                        script {
                            changeSetId = snDevOpsConfigUpload(
                                applicationName: "${appName}",
                                target: 'component',
                                namePath: "${componentName}",
                                configFilePath: "${configFilePath}",
                                autoCommit: 'true',
                                autoValidate: 'true',
                                dataFormat: "${exportFormat}"
                            )
                        }
                    }
                }
            }
        }
    }
}

echo "Changeset: $changeSetId
created"

if(changeSetId != null) {
    // Track config data changes in
    // DevOps Change pipeline
    changeSetRegResult =
    snDevOpsConfigRegisterPipeline(
        applicationContext:
        " ${appName}",
        changesetNumber:
        "${changeSetId}"
    )
    echo "Pipeline registration
result: ${changeSetRegResult}"
} else {
    error "Changeset was not created"
}
}

// Auto-validation was set during upload; get
// status of snapshot
stage('Validate') {
    steps {
        script {
            changeSetResults =
            snDevOpsConfigGetSnapshots(
                applicationContext:" ${appName}",
                deployableName:" ${deployableName}",
                changesetNumber:" ${changeSetId}",
                showResults: false,
                markFailed: false
            )
            if (!changeSetResults) {
}

```

```

                echo "No snapshots were created"
            } else {
                echo "Changest set result :
${changeSetResults}"

def changeSetResultsObject =
readJSON text: changeSetResults

        changeSetResultsObject.each {
            snapshotName = it.name
            snapshotObject = it
        }
        snapshotValidationStatus =
snapshotObject.validation
        snapshotPublishedStatus =
snapshotObject.published
    }
}

script {
    // Set path to snapshot validation
results file
    validationResultsPath =
"${snapshotName}_${currentBuild.projectName}_${currentBuild.number}.xml"

    if(snapshotObject.validation ==
"passed" || snapshotObject.validation == "passed_with_exception") {
        echo "Latest snapshot passed
validation"
    } else {
        error "Latest snapshot failed
validation"
    }
}
}

// Publish snapshot now that it passed validation
stage('Publish') {
    when {
        expression { (snapshotValidationStatus
== "passed" || snapshotValidationStatus == "passed_with_exception") &&
snapshotPublishedStatus == false }
    }
    steps {
        script {
            publishSnapshotResults =
snDevOpsConfigPublish(applicationName:"${appName}",deployableName:"${dep
loyableName}",snapshotName: "${snapshotName}")
        }
    }
}

// Export published snapshot to be used by
downstream deployment tools
stage('Export') {
    steps {

```

```

        script {
            // create build artifacts dir to
            store export file
            sh "mkdir -p ${buildArtifactsPath}"

            exportResponse =
            snDevOpsConfigExport(
                applicationName: "${appName}",
                snapshotName:
                "${snapshotObject.name}",
                deployableName:
                "${deployableName}",
                exporterFormat:
                "${exportFormat}",
                fileName: "${exportFileName}",
                exporterName: "${exporterName}",
                exporterArgs: "${exporterArgs}"
            )
        }
    }
}
}

// Create change request and attach snapshot for reference
stage('Change Management') {
    steps {
        script {
            // Trigger change request
            snDevOpsChange(
                applicationName: "${appName}",
                snapshotName: "${snapshotName}"
            )
        }
    }
}
}

// NOTE: attach snapshot validation results to run (if the snapshot
fails validation)
post {
    always {
        // attach policy validation results
        junit testResults: "${validationResultsPath}",
        skipPublishingChecks: true
    }
}
}
}

```

## Using DevOps Config

The developer, or app engineer, role uses DevOps Config, once it's installed and set up by the DevOps engineer role, to validate and correct config data (that they commit) before it gets deployed.

Consumption process:

1. A configuration change is committed as part of the role of the developer, or app engineer.
2. The build process is kicked off in the pipeline.

When a configuration change is committed in the source code repository, it typically kicks off the build process in the pipeline.

3. Upload configuration data using Azure DevOps pipeline tasks or Jenkins pipeline actions.

DevOps Config pipeline tasks and actions are used to interact with your data model to upload config data for validation.

4. Get snapshot status of the uploaded configuration file.

A snapshot (of configuration data) is created when the configuration change is committed.

5. Check validity of the snapshot using DevOps Config policies.

Once config data is uploaded and the snapshot is created, it's validated in DevOps Config against a set of policies predefined for the specific deployment environment.

6. Publish snapshot after validation.

Once validated by DevOps Config, the config data is then published and a DevOps change request is created by DevOps Change Velocity (change control).

Snapshot information is shown in the DevOps change request **Config changes** tab for accelerated root cause analysis.

7. Approve the DevOps change request.

Once the DevOps change request is approved, the config data is used downstream in your CI/CD pipeline.

8. Deploy to your environment.

Once the change is deployed to your environment, the change request is closed.

9. Use DevOps Config exporters to export your config data to be used by your deployment tools.

## Viewing and editing config data

You update the config data for an application by creating or opening a changeset to the **Config Data** tab. The tab enables you to update the structure and CDIs of config data.

## Tabs on the Application page

When you open an application, the header area for each tab displays administrative details for the application: application name, name of the user that created the application, and timestamp of creation.

- **Overview:** Basic settings of the application and a **Manage deployables** button that enables you to edit config data.
- **Snapshots:** List of snapshots for each deployable in the application. Each entry includes validation and publication status. Policies that execute against a snapshot return validation results. Published snapshots are available for release to the CI/CD pipeline.

- **Config Data:** Read-only view of the config data in a tree structure. Select a node in the tree to view the config data for that node in the application service or infrastructure service. (You can edit an open changeset or create a new changeset. The tab then updates to show an Editing panel and a Preview panel where you can create, edit, and save the changeset. You cannot change an existing snapshot, but you can start with an snapshot changeset and save the changes in a new changeset.)

**i Important:** Save your changes whenever you are confident of the changes and before you leave the Config Data tab.

- **Settings:** A list of policies that are mapped to the deployables. Select an item to open it.
- **Activity:** List of changesets. On this tab, you can open a changeset to continue editing or publish a changeset as a new version of the application config data.

## The Config Data tab

After you open a changeset by selecting **Edit Config Data**, you can edit the config data on the Config Data tab. In the config data tree (A in the screenshot), select the node to edit. By default, the editing panel (C) displays the script view of the key-value pairs (the config data items or CDIs) in the selected node.

### A: Config data tree

The config data tree displays the structured config data. Select a node to edit the associated data.

### B: View selector

To switch from the Script view (shown as panels C and D) to an editable list of CDIs and their values, slide the selector to **List view**.

### C: Editing panel

The Editing panel displays the contents of the selected node. You can edit the config data directly in the Editing panel.

**i Important:** To ensure that you edit only the data that you intend to edit, the panel displays only data that is directly in the selected node. The Editing panel lists all included collection and component names but not their contents. This strategy reduces clutter and clarifies the structure of the deployable. To view the fully-resolved config data, view the Preview panel (D).

### D: Preview panel

The Preview panel displays the persisted state of the data in structured form. If you make changes in the Editing panel and save the changes, the data in the Preview panel is updated to include the changes.

- To resolve variables and view the fully-resolved data in the preview panel: In the **Actions** menu (), select **Apply variables**.
- Encrypted data appears as \*\*\*\*\*. Users with the CDM Secrets [sn\_cdm.cdm\_secrets] role can view all encrypted values in the preview panel. In the **Actions** menu (), select **View encrypted data**.

### E: Actions

- **Refresh View:** Update data in the view.
  - **Save Changes:** Save (persist) the current changes but do not commit the data. The Editing panel, List view, and Preview panel refreshes to reflect the resolved state of the changeset. The system updates the changeset but does not update the application. Changes appear on the **Activity** tab. You must commit a changeset to update the config data for the application. After saving, you can move on to other activities and return later to edit the changeset. The button appears only if you have made changes.
  - **Delete Changeset:** Delete the record of the changeset.
  - **Commit Changeset:** The system generates a snapshot of each deployable that is affected by the changes.
- i Note:** Because changes in two changesets that are open at the same time can conflict, the system blocks such commits. See [Conflicts between changeset commits](#).

#### F: Header for the changeset

The header displays general read-only information about the changeset. You can view this information and additional information on the **Details** tab, as described in the next section.

### The changeset header (F) and the Details tab

#### Information on the changeset header and on the Details tab

Field	Description
Changeset number	Unique sequential changeset number that the system auto-assigns.
Description	Description that helps other users understand the purpose, scope, and intent of the changeset.
Application	Application on which the changeset is based.
State	<ul style="list-style-type: none"> <li>Committed: This draft of the changeset has been committed.</li> <li>Open: The changeset is being updated and is not committed.</li> <li>Blocked: Other commits conflict with this commit. The changeset cannot be committed. See <a href="#">Conflicts between changeset commits</a>.</li> <li>Commit in progress: A draft of the changeset is currently being committed.</li> </ul>
Created / Created by	User that created the changeset and timestamp of creation.
Updated / Updated by	User that updated the changeset and timestamp of update.

## Information on the changeset header and on the Details tab (continued)

Field	Description
Committed / Committed by	User that committed the changeset and timestamp of the commit.

### Define or update a component

Define or update a component while working in a changeset.

#### Before you begin

Role required: cdm\_editor or cdm\_admin

#### About this task

**Important:** Save your changes whenever you are confident of the changes and before you leave the Config Data tab.

#### Procedure

1. Follow this procedure to add a component to an application:
  - a. While working in a changeset, select the menu icon () for the **components** node and select **Create component**.
  - b. Enter a unique and meaningful name and then select **Create**.
2. To perform other actions, select the menu icon () for a component and select the action, as follows.

Selection	Action
Create component	Add a component as described above.
Create CDI	<p>Add an individual config data item (CDI) and value to the collection. See the instructions in the next step</p> <p>The CDI can have an explicit value, or it can be a variable, override, or overlay.</p> <ul style="list-style-type: none"> <li>◦ Variable: The value for this CDI will set the value for an identically-named CDI in an included component.</li> <li>◦ Override: The value for this CDI will override the value for an identically-named CDI in an included component.</li> <li>◦ Overlay: A CDI that does not appear in any included component.</li> </ul>
Include in collections	<p>Include the selected component in specified collections.</p> <p>In the Include in collections dialog box, select the collections that should include the component and then select <b>Include</b>.</p>

Selection	Action
	<b>i Note:</b> Only collections that do not already include the component appear in the list.
Rename	Rename the selected component.
Delete	Delete the selected component.

3. Add a CDI: Select the menu icon () for the item and select **Create CDI**.

### Create CDI

Name	Unique and meaningful name for the CDI.
Value	<p>Value that the CDI has in the current context or define a variable.</p> <ul style="list-style-type: none"> <li>◦ You can specify a direct <code>key : value</code> pair.</li> <li>◦ You can define a variable and specify its default value. Variable definitions have the following form:</li> </ul> <div style="border: 1px solid #ccc; padding: 5px; margin-top: 5px;">@@&lt;variableName&gt;@@ : &lt;value&gt;</div> <ul style="list-style-type: none"> <li>◦ You can specify a value for a variable using the following form:</li> </ul> <div style="border: 1px solid #ccc; padding: 5px; margin-top: 5px;">"&lt;variableName&gt;" : "&lt;value&gt;"</div>
Encrypted	<p>Option to specify that the value of the CDI should be encrypted. This option appears only for users with the CDM Secrets [sn_cdm.cdm_secrets] role.</p> <p>After you create the CDI, the value appears in all views as *****. To view the value on any tab that displays the CDI, users with the CDM Secrets [sn_cdm.cdm_secrets] role can select the <b>View encrypted data</b> menu option.</p>

#### Related reference

[CDM data model](#)

#### Related topics

[Create or update a variable CDI](#)

[Viewing and editing config data](#)

[How encrypted data is handled](#)

## Define or update a collection in an application

Define or update a collection while working in a changeset of a application.

### Before you begin

Role required: cdm\_editor or cdm\_admin

### About this task

**i Important:** Save your changes whenever you are confident of the changes and before you leave the Config Data tab.

### Procedure

1. To add a collection to the application: While working in a changeset, select the menu icon () for the **collection** node and select **Create collection**.
2. To edit individual data items, you can either work directly in the code or select the menu icon () for an item and select an action, as follows.

Selection	Action
Create collection	Add an individual component to the collection.
Create CDI	<p>Add an individual config data item (CDI) and value to the collection. See the instructions in the next step.</p> <p>The CDI can have an explicit value or it can be a variable, override, or overlay.</p> <ul style="list-style-type: none"> <li>◦ Variable: The value for this CDI will set the value for an identically-named CDI in an included component.</li> <li>◦ Override: The value for this CDI will override the value for an identically named CDI in an included component.</li> <li>◦ Overlay: A CDI that does not appear in any included component.</li> </ul>
Include in deployables	<p>Include the selected collection in specified deployables.</p> <p>In the Include in deployables dialog box, select the deployables that should include the collection and then select <b>Include</b>.</p> <p><b>i Note:</b> Only deployables that do not currently include the collection appear in the list.</p>
Rename	Rename the selected collection.
Delete	Delete the selected collection.
Details	View the administrative information associated with the collection.

3. Add a CDI: Select the menu icon () for the item and select **Create CDI**.

### Create CDI

Name	Unique and meaningful name for the CDI.
Value	<p>Value that the CDI has in the current context or define a variable.</p> <ul style="list-style-type: none"> <li>◦ You can specify a direct key : value pair.</li> <li>◦ You can define a variable and specify its default value. Variable definitions have the following form:</li> </ul> <div style="border: 1px solid #ccc; padding: 5px; margin-top: 5px;">@@&lt;variableName&gt;@@: &lt;value&gt;</div> <ul style="list-style-type: none"> <li>◦ You can specify a value for a variable using the following form:</li> </ul> <div style="border: 1px solid #ccc; padding: 5px; margin-top: 5px;">"&lt;variableName&gt;": "&lt;value&gt;"</div>
Encrypted	<p>Option to specify that the value of the CDI should be encrypted. This option appears only for users with the CDM Secrets [sn_cdm.cdm_secrets] role.</p> <p>After you create the CDI, the value appears in all views as *****. To view the value on any tab that displays the CDI, users with the CDM Secrets [sn_cdm.cdm_secrets] role can select the <b>View encrypted data</b> menu option.</p>

4. Optional: Add CDI settings that will override or overlay settings in collections.  
For more information, see [Define or update a component](#).

#### Related reference

[CDM data model](#)

#### Related topics

[Viewing and editing config data](#)

[How encrypted data is handled](#)

[Create or update a variable CDI](#)

#### Create or update a variable CDI

Define variables and set their values in components, collections, and deployables.

#### Before you begin

Role required: cdm\_editor or cdm\_admin

#### About this task

**Important:** Save your changes whenever you are confident of the changes and before you leave the Config Data tab.

- When you create an application, the system auto-generates an empty **vars** folder in each of the three top-level folders: **component**, **collection**, and **deployable**.
- When you add a collection or deployable, the system adds an empty **vars** folder to the item that you created.
- A **vars** folder can contain only variable CDIs or folders. Any subfolder can contain only variables or folders.
- The value of a variable in a collection overrides the value of the variable in a component.
- The value of a variable in a deployable overrides the value of the variable in a collection or component.

## Procedure

- Use either of the following methods to edit or add an instance of a variable while working in a changeset:

- Edit directly in the editing panel. Use the following format to set the value of a variable:

```
"<variableName>": "<value>"
```

In this example, the *dbName* variable and several others reside in the *database* folder. In addition, the *backup* folder (a child of the *database* folder) holds the *dbServer* variable.

- Create the variable in a dialog box. Follow the procedure in the next step.

- Select the menu icon () for the appropriate **vars** folder, select **Create variable**, and then specify its settings.

### Create variable

Name	Unique and meaningful name for the variable.
Value	<p>Value that the variable has in the current context.</p> <p><b>Tip:</b> It is often useful to define a default value for a variable in a component or collection. This is a powerful strategy because you can create a broad variety of deployables from a small set of components and collections. Deployables that inherit a component or collection can use overrides, overlays, and variable settings to meet the needs of the environment type. For example, the Development deployable can use the same components and collections as the Test deployable. Development uses the default <i>database</i> variable value. Test, in contrast, uses a different value that is appropriate for the test environment.</p>

<p>Encrypted</p>	<p>Option to specify that the value of the variable should be encrypted. The option appears only for users with the CDM Secrets [sn_cdm.cdm_secrets] role.</p> <p>After you create the variable, the value appears in all views as *****. To view the value on any tab that displays the variable, users with the CDM Secrets [sn_cdm.cdm_secrets] role can select the <b>View encrypted data</b> menu option.</p>
------------------	--

## Compare config data of two CDM applications

Use the Config Data Analyzer tool to find similarities and differences between the config data of two applications. You can compare either different applications or different changesets of a particular application.

### Before you begin

Role required:cdm\_viewer, cdm\_editor, or cdm\_admin

### Procedure

1. While viewing an application, select **Compare config data**.

The **Compare config data** tab opens.

The current application (the application that you started with, *Demo\_App1* in the example) is called the target. The application to compare to the target is called the reference application. The **Target application** field is set as the latest committed data for the target and is read-only.

2. Select the **Latest data model** option in the **Compare type** field.

3. Select a reference application (*Demo\_App2* in the example).

The **Reference application** selection list includes the latest committed data for all applications except the target application.

4. Optional: Select the folder icon () to specify a folder path for comparison.

5. Optional: Select **Apply variables** to display fully resolved variable values in the comparison results.

6. Optional: If you've the cdm\_secrets role, you can select **View encrypted data** to view encrypted values in clear text in the comparison results.

**Note:** The check box is inactive for users who do not have the cdm\_secrets role. When the values in the target and reference differ, the comparison results indicate that the values differ, but the data itself still appears as \*\*\*\*\*.

7. Select **Compare**.

The Comparison results section displays the differences between the reference changeset and the target changeset.

**Important:** If you commit changes to a reference application while the **Compare config data** tab is open: When you return to the **Compare config data** tab and select **Compare**, the comparison is performed on the most recently committed changeset, and the updated results are displayed in the results section.

The letters in the following illustration identify the tools that you can use to analyze the data.

#### A. Navigation panel

The navigation panel displays the node structure of the applications.

- Nodes that include changed CDIs (either directly or in descendant nodes) are annotated with **Different**.
- Nodes that include newly added CDIs (either directly or in descendant nodes) are annotated with the identity of the application changeset that includes the new data, either **Reference** or **Target**.
- Use the **Search** field to search for text in the navigation panel.
- Select a node to view its contents in the Differences panel.

#### B: Differences panel

- By default, all config data is displayed. Select **Diff only** to view only data that differs between the two changesets.
- Use the **Search** icon () to search for text in the Component differences panel.
- The panel offers two tabs: **Data model** and **CDIs and variables**. In either tab, expand and close groupings with the expansion icon () .
- If a selection includes more than 50 CDIs, then CDIs are organized into pages of 50.

The **Data model** tab displays changes to the structure of the application at the folder level, such as, adding, deleting, or renaming a folder.

#### Data model tab - List columns

Column	Description
Differences	The path of the structure for which differences are described.
Description	Text that describes any differences between the reference and target changesets.
Folder	The folder in the structure.
Source level (Reference/Target)	<ul style="list-style-type: none"> <li>◦ Direct: The folder is explicitly specified or deleted in the node.</li> <li>◦ Included: The folder is included from a child folder.</li> </ul>

The **CDIs and variables** tab displays changes to individual CDIs for the node that is selected in the navigation panel.

- By default, the root node is selected and the data panel includes all CDIs for both changesets. Select a node in the navigation panel to display data for only that node and its descendants.
- When **Diff only** is selected, the number of CDIs that differ appears after the node path.
- Node paths are displayed in gray. Use the expansion icon () to view CDIs in a folder.

#### CDIs and variables tab - List columns

Column	Description
Differences	<p>Indicator to show where the differences are.</p> <ul style="list-style-type: none"> <li>◦ Target: The CDI appears only in the target changeset.</li> <li>◦ Reference: The CDI appears only in the reference changeset.</li> <li>◦ Different: For the specified CDI, the values in the target and reference changesets differ.</li> </ul>
Description	Statement of how the changesets differ. For example, the text might indicate that a CDI appears only in the reference changeset or that the values for variable differ between the changesets. See <a href="#">Types of differences between CDM applications</a> for the full list of difference types and their causes.
Name	Name of the CDI.
Reference (R)	Value of the CDI in the reference changeset.
Target (T)	Value of the CDI in the target changeset.
Source level (R/T)	<p>The node level in the code at which the CDI is defined.</p> <ul style="list-style-type: none"> <li>◦ Direct: The CDI is defined in the selected node.</li> <li>◦ Include: The value of the CDI is obtained from a node that is included in the selected node.</li> <li>◦ Override: The value of the CDI in an included node is overridden by a value that is specified in the including node.</li> </ul> <p>Values appear in the following formats:</p>

Column	Description
	<ul style="list-style-type: none"> <li>◦ &lt;reference level / target level&gt; when a value appears in both target and reference.</li> <li>◦ &lt;reference level&gt; when a value appears in only the reference changeset.</li> <li>◦ &lt;target level&gt; when a value appears in only the target changeset.</li> </ul> <p>For example, when a CDI is defined in the reference changeset with the value true and the CDI does not appear in the target changeset, then the last three columns in <b>CDIs and variables</b> tab will have the following values:</p> <ul style="list-style-type: none"> <li>◦ <b>Reference (R):</b> true</li> <li>◦ <b>Target (T):</b> [Empty]</li> <li>◦ <b>Source level (R/T):</b> Direct</li> </ul>

### Types of differences between CDM applications

Descriptions and examples of differences between CDM applications when using the Config Data Analysis feature.

#### Types of differences

On the **App compare** tab, the **CDIs and variables** tab displays changes to individual CDIs for the node that is selected in the navigation panel. The **Description** column displays one of the following values that describes how the changesets differ.

Values are different

Both the reference and target changesets include a particular CDI, but the values differ between the two changesets. See the **Reference** and **Target** columns for the values.

Encryption settings are different

The **View encrypted data** setting is selected for one changeset and not selected for the other changeset.

**Note:** For users that do not have the CDM Secrets [sn\_cdm.cdm\_secrets] role, encrypted data always appears as \*\*\*\*\*.

Source levels are different

One of the following cases:

- The value is direct in one changeset and is overridden in the other changeset.
- The value is direct in one changeset and is included in the other changeset.

Source levels and values are different

In addition to the cases described in "Source levels are different", the value differs between changesets.

Target only

A CDI or the value for a CDI appears only in the target changeset.

Reference only

A CDI or the value for a CDI appears only in the reference changeset.

## Examples of differences

## Changesets and version control in CDM

A changeset is a draft copy of an application that you can update and save as often as needed. When you are satisfied with your changes, you can commit the changeset to apply the changes to the application. Committing

## About changesets

To edit config data, you create a changeset and make changes in the changeset. You can perform the following actions:

- Create a new changeset: The changeset includes the full config dataset of the application.
- Save a changeset: The Editing panel, List view, and Preview panel refreshes to reflect the resolved state of the changeset. The system updates the changeset but does not update the application. Changes appear on the **Activity** tab. You must commit a changeset to update the config data for the application. After saving, you can move on to other activities and return later to edit the changeset.
- Update an existing changeset: Edit a changeset that had been saved but not committed.
- Commit a saved changeset: The system generates a snapshot of each deployable that is affected by the changes.

## About conflicts with other changesets of the application that you are working on

Sometimes, UserA and UserB are working at the same time on two different changesets of the same application. If UserA commits a changeset that sets *variableX* to A and later, UserB tries to commit a changeset with *variableX* set to B, a conflict results.

An open changeset with conflicts is blocked—it cannot be committed. The system notifies you of conflicts with a warning message on the page. In addition, the **State** value in the header changes from **Open** to **Blocked**.

See [Conflicts between changeset commits](#) for descriptions of the types of conflicts that the system identifies.

## Open a changeset to update config data

Update config data in an existing changeset or in a new changeset. You can save your changes for later updates or save and commit the changes. Committing the changes generates a snapshot for each deployable that is affected by the changes.

## Before you begin

Role required: cdm\_editor or cdm\_admin

## About this task

- When you commit the changes that you have made to an application (commit the changeset), the system generates a snapshot of each deployable that has been updated. A snapshot is an exact record of the config data for a deployable. Snapshots are read-only — you cannot modify a snapshot. You can, however, start a new changeset that is based on a particular snapshot.
- Sometimes, another user has committed changes to a data set (thus creating a new snapshot) while you are working on a changeset of the same data. In this case, the data that you are working on is stale (the data is in the state it was in before the other user committed changes). This can result in data conflicts. Learn more about how you use changesets and how CDM manages version control at [Changesets and version control in CDM](#).

## Procedure

- Use one of the following methods to open a changeset for editing:
  - On the **Activity** tab for an application, select a changeset number.
  - On the **Overview** or **Config Data** tab for an application, select **Edit Config Data**.
- Important:** Save your changes whenever you are confident of the changes and before you leave the Config Data tab.
- The changeset opens on the **Config Data** tab.
- In the config data tree, select the item to edit in either the script view or the list view. By default, the editing panel displays the script view of the key:value pairs (the config data items — CDIs) in the selected node. See [Viewing and editing config data](#) for details on operations in the **Config Data** tab.
- Add components, collections or deployables, see the instructions in [Define or update a component](#), [Define or update a collection in an application](#), and [Create and update a deployable](#).
- Select **Save** to save the changes in the changeset and ready the changes in the changeset to be committed.

The Editing panel, List view, and Preview panel refreshes to reflect the resolved state of the changeset. The system updates the changeset but does not update the application. Changes appear on the **Activity** tab. You must commit a changeset to update the config data for the application. After saving, you can move on to other activities and return later to edit the changeset.

- Optional: Select **Commit**.

The Commit changeset dialog box opens and lists each deployable that is affected by the changes.

**Note:** After you commit a changeset, the changeset cannot be modified.

The system generates a snapshot of each deployable that is affected by the changes. The dialog box offers the following **Additional actions**:

- If no deployables are affected by the changes, select **No additional actions**. The system will update the application to persist the changes.
- ▪ Select **Validate snapshots** to perform the following operations on all generated snapshots:
  - The system executes all policies that are mapped to the associated deployable.
  - If the snapshot passes all policies, then the Validation status value is set to **Passed** and no entries are listed on the **Validation Results** tab. If validation was invoked with the **Publish Valid** publish option, then valid snapshots are auto-published.
  - If any policy encounters issues, then the Validation status value is set to **Failed**. The **Validation Results** tab displays all failures or warnings returned by policies.
  - If any policy failed to run to completion, then the Validation status value is set to **Execution error** and the **Validation Results** tab indicates that validation failed due to error.
- Select **Validate and publish snapshots** to perform validation as described in the **Validate snapshots** action. Only snapshots that pass validation are published. Only published snapshots can be exported.

Select **Commit**.

- The system updates the application to persist the changes to config data.
- The changeset state changes to **Committed**.
- The changeset cannot be modified.
- If snapshots were created, the confirmation message includes a link to the snapshots.
- If an error occurs during the commit process, the changeset state is set to Commit failed.

## Related topics

[Changesets and version control in CDM](#)

[Define or update a component](#)

[Define or update a collection in an application](#)

[Publish or unpublish a snapshot](#)

[Validate a snapshot manually](#)

[View the results of snapshot validation](#)

[View snapshots](#)

## Conflicts between changeset commits

Service delivery can include multiple teams working at the same time on config data with potentially hundreds of configuration changes every day. Because changes can be in conflict with earlier changes by a different user, CDM manages commits and snapshots to block commits that conflict. You are notified of changeset conflicts to help you to resolve them.

## When a conflict happens

Every time you attempt to commit a changeset, the system determines whether there are conflicts with other earlier commits. If the system reports a conflict, you can choose to attempt to keep some of the changes or discard all conflicted changes and start from a new changeset. For this reason, to ease the task of recreating your work, you might copy-paste larger changes to a text editor before closing a conflicted changeset.

## How to avoid conflicts

Follow these suggestions to avoid conflicts:

- Try to keep a changeset open for a brief period. If you need to do research, close the changeset and start a new changeset after you have the information.
- Coordinate your code editing tasks with coworkers. This enables you to avoid updating the same configuration item at the same time.

## Types of conflicts

In each of the following examples of an identified conflict, the "item" being described is the configuration data item (CDI) in your changeset. Another user committed changes that your changes conflict with.

Stale data in your working changeset

- The value of the item was changed in another changeset.
- The item is no longer included in a collection or deployable in another changeset.
- Data corruption caused by an incorrect change in the data table: The newly added item in your open changeset was modified in the data table to incorrectly refer to a previous version. The item in your open changeset was superseded by a change in the data table. The updated or deleted item in your open changeset was incorrectly modified in the data table to not refer to the previous version.

Changed parent

The item is an orphan because its parent was deleted or renamed in another changeset.

Changed parent/child relationship

New items were added in another changeset while you made changes to the parent data item.

Changed references

- The item was included in a collection or deployable in another changeset.
- The item cannot be deleted because it is included in a collection or deployable in another changeset.

Duplicate

An item with the same name already exists.

Invalid includes

- The component or collection to which the include referred was deleted in another changeset.
- The component or collection to which the include referred was renamed in another changeset.
- A descendent of the component to be included is already included in the collection in another changeset.

## View the history of changes to a changeset

View the details of changes in each committed changeset on the **Activities** tab.

## Before you begin

Role required: cdm\_viewer or cdm\_editor or cdm\_exporter\_editor or cdm\_policy\_editor or cdm\_admin

## Procedure

- While working on a changeset, select the **Activity** tab.

The **Changes** section displays the list of individual nodes for which changes were committed.

### Fields on the Changes section of the Activity tab

Header	Description
Old name	The original name of the item. The entry is listed as "(empty)" when the name of the item has not changed. Select the link to view a detailed list of changes.
New name / New value / New type / New state	<p>The setting that the item was changed to in the changeset.</p> <p>For all items, a value appears only if the data changed.</p>
Conflict / Reason for conflict	Boolean that indicates whether this changeset has settings in conflict with another committed changeset.
Path	Path to the item that changed.
Updated / Updated by	User that updated the changeset and timestamp when it was committed.

- Select the **New name** link to view the new, old, and current setting of the name, value, type, and state of a CDI or variable.

The Details dialog box displays the settings and indicates whether the changeset is in conflict with another changeset.

### Details dialog box

Field	Description
New name, New value, New type, New state	<p>The new setting that the item was changed to in this changeset.</p> <p>A value appears only if the data changed.</p>

Field	Description
	<p><b>i</b> <b>Important:</b> The value that appears in the <b>New name</b> column might represent a setting that has not changed. To determine whether the name has changed, select the link. If there is a <b>New name</b> value in the dialog box, then the name has changed.</p>
Old name, Old value, Old type, Old state	The original setting that the item had before the changes in this changeset were committed.
Current name, Current value, Current type, Current state	<p>The setting of the item in the case that a changeset was committed after this changeset was committed.</p> <p>A value appears only if a value changed in the other changeset.</p>

### Delete a changeset

Select the changeset on the Activity tab and then select Delete.

#### Before you begin

Role required: CDM Admin [sn\_cdm.cdm\_admin]

#### Procedure

Select the changeset on the **Activity** tab and then select **Delete**.

#### Delete a CDM application

Delete a CDM application to delete all associated config data and snapshots.

#### Before you begin

Role required: CDM Admin [sn\_cdm.cdm\_admin]

#### Procedure

1. Select the **Applications** icon () and then select one or more applications.
2. Select **Delete** and then confirm the delete action.

#### Result

The system performs the following operations when you delete an application:

- Change the state of the application to "Deleted".
- Disconnect the application from its associated SDLC component.
- Disconnect deployables from CMDB services.
- Return an error for any attempt to query associated snapshots.

## Compare changesets from the same or different CDM applications

Use the Config Data Analyzer tool to find similarities and differences between any two committed changesets from the same or different applications.

### Before you begin

Role required:cdm\_viewer, cdm\_editor, or cdm\_admin

### Procedure

1. While viewing an application, select **Compare config data**.

The **Compare config data** tab opens. The current application is the target application, and its name is pre-populated in the **Target application**.

2. Select the **Changeset** option in the **Compare type** field.

3. Select an application and then their snapshots to compare.

Option	Description
Reference application	Application to compare with the target application. The field lists the latest committed data for all applications including the target application.
Reference snapshot	Snapshot of the reference application to compare.
Target snapshot	Snapshot of the target application against which the reference snapshot is compared.

4. Optional: Select the folder icon () to specify a folder path for comparison.

5. Optional: Select **Apply variables** to display fully resolved variable values in the comparison results.

6. Optional: If you've the cdm\_secrets role, you can select **View encrypted data** to view encrypted values in clear text in the comparison results.

**Note:** The check box is inactive for users who do not have the cdm\_secrets role. When the values in the target and reference differ, the comparison results indicate that the values differ, but the data itself still appears as \*\*\*\*\*.

7. Select **Compare**.

The Comparison results section displays the differences between the reference changeset and the target changeset.

The letters in the following illustration identify the tools that you can use to analyze the data.

A. Navigation panel

The navigation panel displays the node structure of the applications.

- Nodes that include changed CDIs (either directly or in descendant nodes) are annotated with **Different**.
- Nodes that include newly added CDIs (either directly or in descendant nodes) are annotated with the identity of the application changeset that includes the new data, either **Reference** or **Target**.
- Use the **Search** field to search for text in the navigation panel.
- Select a node to view its contents in the Differences panel.

#### B: Differences panel

- By default, all config data is displayed. Select **Diff only** to view only data that differs between the two changesets.
- Use the **Search** icon () to search for text in the Component differences panel.
- The panel offers two tabs: **Data model** and **CDIs and variables**. In either tab, expand and close groupings with the expansion icon () .
- If a selection includes more than 50 CDIs, then CDIs are organized into pages of 50.

The **Data model** tab displays changes to the structure of the application at the folder level, such as, adding, deleting, or renaming a folder.

#### Data model tab

Column	Description
Differences	The path of the structure for which differences are described.
Description	Text that describes any differences between the reference and target changesets.
Folder	The folder in the structure.
Source level (Reference/Target)	<ul style="list-style-type: none"> <li>◦ Direct: The folder is explicitly specified or deleted in the node.</li> <li>◦ Included: The folder is included from a child folder.</li> </ul>

The **CDIs and variables** tab displays changes to individual CDIs for the node that is selected in the navigation panel.

- By default, the root node is selected and the data panel includes all CDIs for both changesets. Select a node in the navigation panel to display data for only that node and its descendants.
- When **Diff only** is selected, the number of CDIs that differ appears after the node path.
- Node paths are displayed in gray. Use the expansion icon () to view CDIs in a folder.

**CDIs and variables tab**

Column	Description
Differences	<p>Indicator to show what was the change type.</p> <ul style="list-style-type: none"> <li>◦ Target: The CDI appears only in the target changeset.</li> <li>◦ Reference: The CDI appears only in the reference changeset.</li> <li>◦ Different: For the specified CDI, the values in the target and reference changesets differ.</li> </ul>
Description	<p>Statement of how the changesets differ. For example, the text might indicate that a CDI appears only in the reference changeset or that the values for the variable differ between the changesets.</p> <p>See <a href="#">Types of differences between CDM applications</a> for the full list of difference types and their causes.</p>
Name	Name of the CDI.
Reference (R)	Value of the CDI in the reference changeset.
Target (T)	Value of the CDI in the target changeset.
Source level (R/T)	<p>The node level in the code at which the CDI is defined.</p> <ul style="list-style-type: none"> <li>◦ Direct: The CDI is defined in the selected node.</li> <li>◦ Include: The value of the CDI is obtained from a node that is included in the selected node.</li> <li>◦ Override: The value of the CDI in an included node is overridden by a value that is specified in the including node.</li> </ul> <p>Values appear in the following formats:</p>

Column	Description
	<ul style="list-style-type: none"> <li>◦ &lt;reference level / target level&gt; when a value appears in both target and reference.</li> <li>◦ &lt;reference level&gt; when a value appears in only the reference changeset.</li> <li>◦ &lt;target level&gt; when a value appears in only the target changeset.</li> </ul> <p>For example, when a CDI is defined in the reference changeset with the value true and the CDI does not appear in the target changeset, then the last three columns in the <b>CDIs and variables</b> tab will have the following values:</p> <ul style="list-style-type: none"> <li>◦ <b>Reference (R):</b> true</li> <li>◦ <b>Target (T):</b> [Empty]</li> <li>◦ <b>Source level (R/T):</b> Direct</li> </ul>

## Validating and correcting configuration data

For every change to an environment, there may be tens or hundreds of rules (policies) that need to be checked. CDM enables you to auto-validate configuration data to ensure that configuration changes are error-free and policy-compliant before the config data is consumed.

### CDM automates much of the validation process

When teams deploy an application, they typically need to ensure that no errors are introduced at the code and configuration level, and that compliance policies are respected. For every change to an environment, there may be tens or hundreds of rules (policies) that need to be checked. A manual validation process is prone to errors.

DevOps provides developers, IT admins, DevOps Config engineers, or team leads a way to ensure that configuration changes are error free and policy-compliant. In addition, after validation give developers or IT admins enough contextual information to identify and fix issues. This means that after a user commits any changes to configuration data, all relevant policies are applied, results evaluated, and appropriate actions are triggered (for example, deploy or do not deploy).

Here's the CDM process that auto-validates configuration data before it is consumed.

1. You create a policy version based on an existing policy and then activate and publish the new version.
2. You map the policy to a deployable to ensure that the policy executes against any snapshot of the deployable when requested (manual or automated request).
3. When you commit a changeset for the deployable, the system needs to validate the resulting snapshot. To auto-trigger execution of the mapped policies whenever a

changeset is committed, select the **Validate** option when committing the changeset. You also have the option to manually validate the snapshot.

If you configured the automated process to be integrated with your pipeline, then the policies return results of the validation process to the pipeline in JSON files. The pipeline decides whether to continue based on the validation decision: Compliant, Non-compliant.

4. If the process is not integrated with your pipeline, or if validation failed or did not finish, then you review the validation decision and take appropriate action.

## Use pre-defined policies as models for your custom policies

CDM includes policies that you can use as models for custom policies that meet specific needs. Note the following important patterns in the pre-defined policies:

- The policies all use the *CdmQuery* script include to retrieve config data.
- Use *CdmQuery* with Secrets to ensure all data is included.
- The *CdmPolicyUtil* script include adds warnings and failures.
- When all policies have run, if a failure has been populated in the output, the decision is non\_compliant.
- Use the logger for debugging. Remove the logs when the policy is ready to be published.
- Use mapping input parameters to supply dynamic input values.

## Guidelines for validating snapshots

- Do not manually validate snapshots that have already been validated and published.
- If you select the **Validate snapshots** or **Validate and publish snapshots** option when committing a changeset, then the system auto-validates each snapshot when it is first generated.
- To view the current validation failures or warnings for a snapshot, open the snapshot and select the **Validation Results** tab. For details, see [View the results of snapshot validation](#).
- Use the Policy Test Playground feature to re-validate snapshots while you develop a policy. The resulting validation results are flagged as test results and do not affect operations.
- If there is a requirement to revalidate all snapshots for a deployable, revalidate only after you have tested and published the policies.

## Practices to avoid

- Do not use *GlideRecord* to retrieve config data records.
- To display policy decisions on the UI, CDM requires warning and failure data in a particular structure. Use *CdmPolicyUtil* to add warnings or failures (not manual insertions). (At least understand the structure that *CdmPolicyUtil* inserts into failures and warnings.)
- In policies, do not attempt to retrieve data that does not come from CDM data. This practice can result in cross-scope access issues.
- Do not write "always-compliant" or "always-non-compliant" policies.
- Do not hard-code any names or values in policies. Instead, use mapping input parameters to supply dynamic input values.

## Snapshot status

- Not validated: This is the initial snapshot state where no policy has been executed against the snapshot.
- Requested: The validation flow has started.
- In progress: The validation flow is running and policies are currently executing.
- Passed: The snapshot has passed all policies.
- Failed: The snapshot has failed one or more policies.
- Execution error: A policy failed to run to completion.

## EXAMPLE FLOW

A user commits changes to config data before a deployment.

Data is validated against all assigned policies.

If validation fails, then deployment is stopped. A list of errors pointing to their location in the config data is given to user. Available actions should be determined in the context of the application itself and the environment where the application is deployed. For example:

- Allow deployment in TEST environment regardless of # of errors or warnings.
- Stop deployment in PROD if any error.
- Stop deployment in PROD if any warning and if application is critical.
- Allow deployment in PROD regardless of # of warnings if non critical.

User fixes errors and re-commits change.

Data is validated again and results are ok.

Snapshot is taken.

Deployment is executed.

Related topics

[CdmQuery - Scoped](#) ↗

### Validate a snapshot manually

Validate a snapshot from the Snapshot tab. For example, manually validate a snapshot to ensure that it is valid for new policy requirements.

#### Before you begin

Role required: cdm\_editor or cdm\_admin

#### About this task

- Do not manually validate snapshots that have already been validated and published.
- If you select the **Validate snapshots** or **Validate and publish snapshots** option when committing a changeset, then the system auto-validates each snapshot when it is first generated.
- To view the current validation failures or warnings for a snapshot, open the snapshot and select the **Validation Results** tab. For details, see [View the results of snapshot validation](#).

- Use the Policy Test Playground feature to re-validate snapshots while you develop a policy. The resulting validation results are flagged as test results and do not affect operations.
- If there is a requirement to revalidate all snapshots for a deployable, revalidate only after you have tested and published the policies.

## Procedure

1. Use either of the following methods to validate a snapshot:
  - Use the REST API endpoint to validate.
  - On the **Snapshots** tab for an application, select the name of the snapshot to validate and then select **Validate**.
  - On any tab for an open snapshot, select **Validate**.
2. Confirm the action on the Validate snapshot dialog box.

The system follows this procedure to validate snapshots:

- The system executes all policies that are mapped to the associated deployable.
- If the snapshot passes all policies, then the Validation status value is set to **Passed** and no entries are listed on the **Validation Results** tab. If validation was invoked with the **Publish Valid** publish option, then valid snapshots are auto-published.
- If any policy encounters issues, then the Validation status value is set to **Failed**. The **Validation Results** tab displays all failures or warnings returned by policies.
- If any policy failed to run to completion, then the Validation status value is set to **Execution error** and the **Validation Results** tab indicates that validation failed due to error.

## What to do next

If the snapshot failed validation, you can view the issues to fix them. See [View the results of snapshot validation](#).

Related topics

[View the results of snapshot validation](#)

## View the results of snapshot validation

View validation failures and warnings on the **Validation Results** tab for the snapshot.

## Before you begin

Role required: cdm\_viewer or cdm\_editor or cdm\_exporter\_editor or cdm\_policy\_editor or cdm\_admin

## Procedure

1. On the **Snapshots** tab for an application, select the name of the snapshot to open the **Validation Results** tab.
2. View failure and warning information on the **Validation Results** tab for the snapshot.

The Policies panel displays a card for each policy that is mapped to the associated deployable.

## Validation Results tab: Policies panel

Section	Description
All Policies	The card displays the total number of failures and warnings for this validation run.
Policy list	For each policy that generated a failure or warning, the system lists whether the snapshot is compliant with the policy and the number of failures and warnings that the policy generated.

- Select a box to view the associated data in the Failures and warnings panel.
- Select the menu icon () for a policy and select an action, as follows:

### Action menu

Action	Description
Execution record	View execution information for this policy run. See <a href="#">View the execution record for a policy run</a> for details.
Open policy	Open the policy.

The Failures and warnings panel displays the list of all failures and warnings for each policy that is mapped to the associated deployable.

## Validation Results tab: Failures and warnings panel

Field	Description
Description	Text that provides information about the failure or warning result.
Type	The type of validation result: <ul style="list-style-type: none"> <li>◦ (empty)</li> <li>◦ Information</li> <li>◦ Failure</li> <li>◦ Warning</li> </ul>
Policy	Name of the policy that returned a failure or warning. Select to open the policy.
Impacted data	The CDI (name: value pair) that resulted in the failure or warning.
Path	Path to the node in the data tree that includes the impacted data.

### Fix the data that caused a validation failure

When a policy that executes against a deployable snapshot fails or generates an error or warning, the policy identifies the offending CDI and the reason for the failure. While working in a changeset, you can use the Validation failures panel to navigate directly to the offending CDI so you can correct the issue.

## Before you begin

Role required: sn\_devops\_config.admin

## About this task

The Validation failures panel appears on the application tab whenever validation fails for any deployable in the application. The panel displays only snapshots with error, warning, or execution failure status for validation. Snapshots with not validated or in progress status do not appear.

**i Important:** The data changes and status settings that you update in this procedure apply only to the selected changeset. For that reason, be sure to coordinate your work with others on your team.

## Procedure

1. While working in a changeset, select the Validation failures icon () to view the list of snapshots from the application that failed validation.  
In this example, the snapshots for all three deployables in the application failed validation in some way (error, warning, or execution failure).

2. Navigate to the failure.

- a. Select a snapshot to view cards.

Each card represents a policy validation failure. Use the filter to find a card with a particular validation result. If another user has worked on the issue, you might also filter on a resolution state (resolved, unresolved, or ignored).

- b. Select a card to open the editor directly to the problematic CDI.

3. Review the data and perform the necessary updates.

4. When you finish working on the data, update the status of the error.

Open the **Actions** menu () to update the status of the error.

- **Unresolved:** This is the initial setting for a validation error. Leave this setting if you are unable to repair the data. You might work on the issue with another user.
- **Resolved:** You have fixed the error. Be sure to commit the changeset and validate the data to ensure that the data is actually correct.
- **Ignored:** Indicates that no further actions will be taken now.

## View the execution record for a policy run

View the execution record to analyze policy execution.

## Before you begin

Role required: cdm\_viewer or cdm\_editor or cdm\_exporter\_editor or cdm\_policy\_editor or cdm\_admin

## Procedure

On the **Validation Results** tab for a snapshot, select the menu icon () for a policy and select **Execution record**.

## Policy execution information on the Details tab

Field	Description
Policy	Name of the policy that executed.
Version name	Version of the policy that executed.
End time	Timestamp when execution finished.
Decision	<p>Result of policy execution.</p> <ul style="list-style-type: none"> <li>• Compliant</li> <li>• Non-compliant</li> <li>• Execution error</li> <li>• Compliant with exception</li> </ul>

## Adding warning and failure messages to validation results — CdmPolicyUtil

You use the `CdmPolicyUtil` script include to add warning and failure messages to validation results in the CDM Policy Validation Results table. CDM expects validation warnings and failures to contain a node path, a snapshot ID, and a reference to the impacted node.

### CdmPolicyUtil

- `CdmPolicyUtil` is a public script include.
- Because `CdmPolicyUtil` is a global script include, you do not need to invoke it like a new `CdmPolicyUtil()`.
- You can invoke `CdmPolicyUtil` in all scopes, but the expected invocation scope is PaCE (specifically, during the execution of PaCE policies).
- The proper call is `CdmPolicyUtil.methodName(parameters)`.

### Methods

#### addFailure

Adds a failure message. The message appears on the **Policy Validation Results** tab for the snapshot.

Parameters:

- `Output`: Policy decision for the subject snapshot (`primarySnapshotId`).
- `cdmNode`: Node (`sn_cdm_node`) that caused the failure.
- `name`: Name of the failure.
- `description`: Description of the failure.

#### addWarning

Adds a policy warning message. The message appears on the **Policy Validation Results** tab for the snapshot.

Parameters:

- *Output*: Policy decision for the subject snapshot (*primarySnapshotId*).
- *cdmNode*: Node (*sn\_cdm\_node*) that caused the warning.
- *name*: Name of the warning.
- *description*: Description of the warning.

#### getLastCreatedSnapshotIds

Returns the latest created and published snapshot IDs for the supplied *additionalDeployables*. If no published snapshot is available for a particular deployable, adds a debug message.

Parameter:

*additionalDeployables* – can be passed directly as it comes to the policy.

#### getLastPublishedSnapshotIds

Retrieves the latest and published snapshot IDs for the supplied *additionalDeployables*. If no published snapshot is available for a particular deployable, adds a debug message.

Parameter:

*additionalDeployables*: Can be passed directly as it comes to the policy.

## Sharing components among applications — Component libraries

Some applications may share the same basic structure and require nearly identical configuration data. Shared components in CDM enables you to use a component across several applications. For better organization, these shared components are managed in component libraries.

### Component libraries

Component libraries improve consistency and maintainability by ensuring a single source of truth for a component's config data across applications. You can use the unified view in the DevOps Config workspace or REST API to create and maintain these libraries.

In this example, an organization sells tea on its website. Both the *Shopping-Cart* and *Browsing-Pane* application services make use of config data for product prices and photo appearance. To ensure that the config data is identical in both DevOps applications, each application uses shared components from the *Tea-Service* component library. The components are managed in the library and the applications each use two of the components from the library.

### Working with shared components

- A user with the *sn\_cdm.cdm\_admin* role can create and manage a component library and create, add, and delete shared components in the library.
- While working in an application changeset, you can add, update, or remove a shared component.
- Applications can use any mix of components: components that are defined in the application (direct components) and components from a component library.

- While working in an application changeset, you cannot modify a shared component in the same ways as you can modify a direct component. A collection in an application can, however, override any value in a shared component.
- For a shared component to be available for use in applications, the component must be in the **Published** state and the library that holds the component must be in the **Available** state.

In the example, no application can use the *Flavor-Sort-settings* component because it has not been published.

### Working on component libraries with a unified view

The unified view of component libraries in the DevOps Config Workspace provides a single location to view and manage component libraries and the components within them.

Using the unified view, you can:

- Search for a component library or shared component.
- Navigate quickly between different component libraries or shared components.
- View a list of all applications that use shared components of a library.
- Edit the config data of a library.
- View and create changesets for the library and components.
- View different versions of a shared component, and publish or unpublish them.

### Users

The following users can view and manage the component libraries and shared components in the DevOps Config Workspace.

- CDM admin can view and manage the libraries and shared components.
- CDM editor can view and manage the shared components within the libraries.
- CDM viewers can view the libraries and shared components.

### Accessing the component libraries

To open the Component libraries, navigate to **All > DevOps Config > DevOps Config Workspace** and select the component libraries icon () .

### Component libraries list

On the Component libraries list, you can review all component libraries and shared components within them. The list pane has the following sections to enable you to access and manage the libraries and components quickly.

- **View:** Filter libraries by the categories: All libraries and Editable libraries. Editable libraries are the libraries that you create or that others create but grant you permission to view and edit through the associated authoring groups.
- **Search:** Search library or component by their names.
- **Create new library:** Create a library using this option.
- **Listing:** Select a library node or shared component node to view it in the form pane.

The following is an example of a unified view for Component libraries.

## Unified view in DevConfig Workspace to manage component libraries

### Component library form

On the Component library form pane, you can review and manage your libraries and components within them. Select a library node in the left pane to view and edit the associated data.

- **Header:** The header section shows the status and other metadata of the component library. You can take the following actions:
  - Mark a library as available or unavailable for use by using the **Available** toggle switch.
  - Save the changes to the library.
  - Edit to create or open a changeset for updating shared components.
- **Tabs:** The following tabs are available:
  - **Details:** View and update a library's details such as name, description, and authoring groups. The tab also includes a list of all applications that are using one or more components of the library.
  - **Components:** View all the components available in the library.
  - **Config data:** Review the config data in each component of the library. You can switch between Editor and List view using the **List view** toggle switch.
  - **Activity:** View a list of all changesets created for updating the library, its components, and config data.

The following is an example of a component library form in the DevOps Config Workspace.

### Component library form

### Shared component form

On the Shared component form pane, you can review and manage components within your libraries. Select a component node in the left pane to view and edit the associated data.

- **Header:** The header section shows the status and other metadata of the component library. You can take the following actions:
  - Save the changes to the component.
  - Edit to create or open a changeset for updating the component.
  - Refresh the form data.
- **Tabs:** The following tabs are available:
  - **Details:** View the component's details and update its description as required. The tab also includes a list of all applications that are using the components.
  - **Versions:** View all the published and unpublished versions of the component. You can also publish a component's version or unpublish a version.

The following is an example of a shared component form in the DevOps Config Workspace.

### Component library form

## Create or update a component library

Create a component library to manage components that can be reused across multiple applications.

### Before you begin

Role required: cdm\_admin

### Procedure

1. Navigate to **All > DevOps Config > DevOps Config Workspace**.
2. Click the component libraries icon () in the left navigation to open the **Component libraries** tab.
3. Click **Create new library** to create a component library.
4. On the form, fill in the fields.

#### Create component library form

Field	Description
Library name	<p>Name of the component library.</p> <p>This name must be unique across all CDM applications and libraries.</p>
Description	Description about the component library, such as what type of components it includes.
Authoring groups	<p>User groups with the ability to edit the library. Users in the groups with appropriate roles can view the component library.</p> <p><b>i Note:</b> If you choose an authoring group, make sure you're a member of that group, or you'll lose editing access to the library.</p>

5. Click **Create**.

### Result

A component library is created. The state of the library is set to Not Available, which means that any components added to the library cannot be used, updated, or exported.

### What to do next

1. [Add or update a shared component in a component library](#).
2. [Activate a component library](#).

### Activate a component library

Activate a component library to make all of its shared components available for use.

### Before you begin

Role required: cdm\_admin

### About this task

A library can be in the Unavailable state in the following situations:

- When a library is created for the first time
- When a library and its components are being created
- When a newer version of the library is being created

If a library is unavailable, no component within that library can be used, updated, or exported. However, it does not impact applications that are already using components within this library.

When the library is ready for use, you can activate it.

### Procedure

1. Navigate to **All > DevOps Config > DevOps Config Workspace**.
2. Click the component libraries icon () in the left navigation to open the **Component libraries** tab.
3. Select the **Available** toggle switch to make the library available so that its shared components can be used in applications.

### Result

- The library and the shared components in it become available for use.
- The status of the library is set as Available.

### Delete a component library

Delete a component library and all of its components if no application is using the library's components.

#### Before you begin

Role required: cdm\_admin

### Procedure

1. Navigate to **All > DevOps Config > DevOps Config Workspace**.
2. Click the component libraries icon () in the left navigation to open the **Component libraries** tab.
3. From the **Options** button (), select **Delete** and then confirm the delete action.

**Note:** If a component is being used in any application, the **Delete** option is not available.

### Result

The component library and all of its components are deleted.

### Add or update a shared component in a component library

Add or update a shared component in a component library.

#### Before you begin

Role required: cdm\_editor or cdm\_admin

### About this task

You can also add a component to the library as a shared component from a request. For more information, see [Accept or reject a component request for inclusion in a component library](#).

## Procedure

1. Navigate to **All > DevOps Config > DevOps Config Workspace**.
2. Click the component libraries icon () in the left navigation pane to open the **Component libraries** tab.
3. Select a component library from the **Component libraries** list tab to which you want to add a shared component.
4. Click **Edit** to create a changeset or select an open changeset to work.
5. Add a shared component to the library.
  - a. On the **Config data** tab of the changeset record, select the menu icon () for the **components** node and select **Create shared component**.
  - b. In the Create shared component dialog box, enter a unique and meaningful name for the component.
  - c. Select **Create**.  
A component is created.
6. Add config data items (CDIs) and values for the shared component.
  - a. With the shared component node selected, add one or more CDIs and values in the Editor pane.
  - b. Click **Save**.
7. Optional: Select the menu icon () for a component to rename or delete it.
  - To rename the component, select **Rename**.
  - To delete the component, select **Delete**.

**Note:** These options are available only if the component is not being used in any application.
8. Click **Commit** to commit all changes for the shared component.  
A new version of the shared components will be created.
9. In the **Additional actions** field, select an option to either publish the latest version or do it later.
  - **No additional actions:** Does not process anything for the latest version of the component.
  - **Publish latest version:** Publishes the latest version of the component.

**Note:** For a shared component, only one version can be published at a time.  
So, any existing published version will be unpublished and the new version will be published.

### Publish or unpublish a shared component version

Publish a version of a shared component in a library so that it can be used in an application.

#### Before you begin

Role required: cdm\_admin

## About this task

When a version is unpublished, it means that the component and its version can no longer be used in an application. However, it has no impact on applications that are already using the version.

**Note:** A shared component can only have one published version at a time.

## Procedure

1. Navigate to **All > DevOps Config > DevOps Config Workspace**.
2. Click the component libraries icon () in the left navigation pane to open the **Component libraries** list tab.
3. Select a component library from the **Component libraries** list tab.
4. Select the **Components** tab and then select a component name to open.
5. Select the **Versions** tab.
6. Publish or unpublish a version of the component.

Option	Description
Publish a version of a shared component	Select an unpublished version from the list and select <b>Publish</b> .  If there's any existing published version of the component, then it's unpublished before publishing the selected version. The <b>Published</b> value updates to <b>true</b> .
Unpublish a version of a shared component	Select a published version from the list, select the , and then select <b>Unpublish</b> .  The selected version of the component is unpublished and the <b>Published</b> value updates to <b>false</b> .

## Request to include a component to a component library

Submit a request for a component in an application that could be reused across multiple applications to be added to a shared component library.

## Before you begin

Role required: cdm\_admin or cdm\_editor

## About this task

While creating data models for their applications, DevOps Config editors and administrators may identify certain components could be reused across multiple applications. They can submit a request to the component library owners to add those components to the library.

**Note:** You can only submit one request for a component for the same library at a time.

## Procedure

1. Navigate to **All > DevOps Config > DevOps Config Workspace**.
2. Click the apps icon () in the left navigation to open the **Apps** tab.
3. Open an application and select the **Config data** tab.

4. Select the shared components icon () .
5. In the Shared components pane, click **+ Create request**.
6. Enter the details in the request form.

#### Create shared component request

Field	Description
Component	<p>Name of the component to be added to the request.</p> <p><b>i Note:</b> Config data from the last committed changeset is added to the request, excluding any secret data values for keys and referenced variables.</p>
Target library	<p>Component library to which the requested component should be added.</p> <p>Only existing and available libraries are listed in the field.</p>
Component description	Description about the component, such as how it works.
Additional details	More details about the component that could be helpful to the approver during approving.

7. Click **Submit**.

A request is created for the library owner to add the selected component as a shared component in the target library. The state of the request is set as Pending review and is listed in the Shared components pane.

You can see all of your requests from the last 30 days and filter them by their state.

#### What to do next

While a request is in the Pending review state, you can change its information or withdraw it by selecting the request card in the Shared components pane.

- To update the request, update the component's description or additional details in the respective fields and click **Update**.
- To withdraw the request, click **Withdraw**.

The library owner can review and approve or reject the request.

#### Accept or reject a component request for inclusion in a component library

Review a request for adding a component to a component library and approve or reject it.

#### Before you begin

Someone has requested to add a local component from an application to the component library as a shared component. If there are such requests in a library that need your approval, you see a **Requests** label () beside the library name.

Role required: cdm\_admin or cdm\_editor

## Procedure

1. Navigate to **All > DevOps Config > DevOps Config Workspace**.
  2. Click the component libraries icon () in the left navigation to open the **Component libraries** tab.
  3. Open a component library and select the **Requests** tab.
- i Note:** The **Requests** tab is available only when there's an open request for adding a component to the library.
4. Click a requests card to review its config data.  
The config data of the selected card appears in the preview pane.
  5. As required, click accept or reject the request.

Approval action	Description
Accept a request	<ul style="list-style-type: none"> <li>a. Click <b>Accept</b> on the card.</li> <li>b. As required, update the fields in the Approve Request pop-up window. <ul style="list-style-type: none"> <li>▪ Select a library in the <b>Target library</b> field to which the component will be added.</li> <li>▪ Update the name of the component in the <b>Component</b> field.</li> </ul> </li> </ul> <p><b>i Note:</b> If a component with the same name exists in the library, enter a different name.</p> <ul style="list-style-type: none"> <li>▪ Update the description of the component in the <b>Component description</b> field.</li> <li>c. Add comments about the approval for the requester to see in the request.</li> <li>d. Select an action option in the <b>Additional actions</b> field to perform after the request is processed. <ul style="list-style-type: none"> <li>▪ <b>No additional actions:</b> Does not process anything for the latest version of the component.</li> <li>▪ <b>Publish version:</b> Publishes the component version.</li> </ul> </li> <li>e. Click <b>Accept</b>.</li> </ul>
Reject a request	<ul style="list-style-type: none"> <li>a. Click <b>Reject</b> on the card.</li> <li>b. Enter the reason of rejection for the requester to see in the request.</li> <li>c. Click <b>Reject</b>.</li> </ul>

## Result

- The request is removed from the **Requests** tab.
- Any comments added in the request can be seen by the requester in the app from where the request was submitted.
- If the request is approved:
  - The state of the request updates to Accepted.
  - The component is added to the selected component library as a shared component. Any secret data values (such as account id or passwords) for keys and referenced variables are not included with the component.

If the library is available, the component is ready for use in multiple CDM apps.

  - If you selected **Publish version** in the **Additional actions** field, then the shared component is published.
- If the request is rejected, the state of the request updates to Rejected.

## Manage shared components in a CDM application

Add a shared component to an application, view shared components that are available to add, or remove a shared component from an application.

### Before you begin

Roles required: CDM Editor [sn\_cdm.cdm\_editor]

### About this task

- While working in an application changeset, you can add, update, or remove a shared component.
- While working in an application changeset, you cannot modify a shared component in the same ways as you can modify a direct component. A collection in an application can, however, override any value in a shared component.

### Procedure

- While working on the **Config data** tab for an application changeset, select the Shared components icon ()�.

- Each card on a Shared components tab displays the name, parent component library, and version for a shared component.
- Components are grouped by library and are sorted alphabetically.
- Use the **Libraries** selector to display only components in the selected library.
- Use the **Search** text box to filter the list of components or libraries.

- Available:** List of shared components that the application can include in its config data.

For a shared component to be available for use in applications, the component must be in the **Published** state and the library that holds the component must be in the **Available** state.

- In use:** List of shared components that are currently used in the application.

- Updates:** A card appears when newer published version of a component is available. Select the **refresh** icon () to ensure that you are viewing all current updates.

2. Add a shared component to application data: On the **Available** tab, select **Add**.
  - The component is added to the **components** folder in the application.
  - The shared component icon () indicates that the component comes from a component library.
  - The component is selected and the **Editor** and **Preview** panels display the code of the shared component.

3. View the config data in a shared component: On any card, select the **Actions** menu () and then select **View config data**.

A popup displays the latest published version of the shared component code. Variables are not resolved and encrypted data is not decrypted in the popup view.

4. Remove a shared component: Use either of the following methods to remove a shared component from the application data.
  - On the **In use**: tab, select the **X** in the component card and then confirm the action in the popup.
  - On the **Config data** tab for an application, select the **Actions** menu () for the component and then select **Remove**. Confirm the action in the popup.

**i Note:** Removal is not deletion. Removal takes the component's config data out of the application, but the component remains unchanged in the component library. When you remove a component from an application, the component card reappears on the **Available** tab.

5. Update a shared component.

Select the **refresh** icon () to ensure that you are viewing all current updates. The system indicates that a newer published version of a component is available in all of the following ways:

- Highlighted text "Update" appears next to the component in the **components** folder.
- The **Updates** tab displays a card for the component.
- The popup for committing a changeset includes a notification: "One or more of the shared components have updates available".

Follow this procedure to review the updated config data and then update a component:

- a. On the **Updates** tab, select **Update** for the component.
- b. Review the information on the Update a shared component popup and then select **Update**.
  - The **Details** tab displays name and version information.
  - The **Latest config data** tab displays the latest version of the shared component code. In the view, variables are not resolved and encrypted data is not decrypted.

#### Related topics

[Sharing components among applications — Component libraries](#)

#### View snapshots

View the snapshots of all deployables for an application on the **Details** tab for the application. The release pipeline can access only a snapshot that is the currently published version.

## Before you begin

Role required: cdm\_viewer or cdm\_editor or cdm\_exporter\_editor or cdm\_policy\_editor or cdm\_admin

## Procedure

1. Open an application and select the **Snapshots** tab.

The tab lists all deployables that are associated with the application. Each deployable displays a short list of its snapshots. Select **Show all** to view the full list of snapshots.

### Overview tab

Heading	Description
Name	<p>Friendly name for the snapshot.</p> <p>When a snapshot is generated, the system uses the deployable name and the snapshot number to generate a default name (for example, <code>deployableName_1</code>, <code>deployableName_2</code>). You can edit the name.</p> <p><b>Tip:</b> You can edit the name on the <b>Details</b> tab. Select the snapshot name to open the tab.</p>
Description	<p>Descriptive information that you supply to help other users.</p> <p><b>Tip:</b> You can add a description on the <b>Details</b> tab. Select the snapshot name to open the tab.</p>
Deployable	<p>Deployable on which the snapshot is based. Select the name to open the deployable.</p>
Published	<p>Boolean that indicates whether the snapshot is published. Only published snapshots can be exported.</p> <p><b>Important:</b> By default, exporters use the most recently created published snapshot.</p> <p>While defining or updating a deployable, you can specify that only validated snapshots can be published. You can unpublish a snapshot. See <a href="#">Publish or unpublish a snapshot</a>.</p>
Validation status	Validation status:

Heading	Description
	<ul style="list-style-type: none"> <li>◦ Not validated: This is the initial snapshot state where no policy has been executed against the snapshot.</li> <li>◦ Requested: The validation flow has started.</li> <li>◦ In progress: The validation flow is running and policies are currently executing.</li> <li>◦ Passed: The snapshot has passed all policies.</li> <li>◦ Failed: The snapshot has failed one or more policies.</li> <li>◦ Execution error: A policy failed to run to completion.</li> </ul>
Created	Timestamp when the snapshot was created.
Tags	

2. Select the snapshot name to view additional details and to set values on the **Details** tab.

#### Details tab for a snapshot

Heading	Description
Name	<p>Friendly name for the snapshot.</p> <p>When a snapshot is generated, the system uses the deployable name and the snapshot number to generate a default name (for example, <code>deployableName_1</code>, <code>deployableName_2</code>). You can edit the name.</p>
Description	Descriptive information that you supply to help other users.
Deployable	Deployable on which the snapshot is based. Select the icon to open the deployable.
Changeset	The auto-generated changeset number. Select the icon to open the changeset.
Number	Unique incremental snapshot number. The system auto-assigns the number <code>SNAPn</code> , where <code>n</code> is an incremented seven-digit number.

Heading	Description
Created / Created by	User that created the changeset and timestamp` of creation.
Published on	Timestamp when the snapshot was published.
Last validated	Timestamp that results were returned for the most recent validation. No value appears if the snapshot has not yet been validated.
Tags	Text tag that enables you to filter the snapshots in a list.

### Publish or unpublish a snapshot

Publish a snapshot so that it can be exported to enable the CI/CD pipeline to access and use the config data. Exporters can execute only on published snapshots.

#### Before you begin

Role required: CDM Editor [sn\_cdm.cdm\_editor] or CDM Admin [sn\_cdm.cdm\_admin]

#### About this task

- You publish a snapshot when you feel that the deployable that it represents is ready to export to the pipeline.
- While defining or updating a deployable, you can specify that only validated snapshots can be published.
- A deployable must be connected to a service to publish its snapshots.
- If a published deployable is disconnected from services, then you can perform the following actions, but cannot publish snapshots of the deployable until it is connected to a service:
  - Edit config data.
  - Create snapshots.
  - Validate snapshots.
  - Fix validation errors.

 **Tip:** You can add a text description of a snapshot or rename it on the **Details** tab for the snapshot.

- When exporters look for the latest snapshot, they should look for the latest `created` snapshot that is published.
- The `published_date` is the date at which the publish happens.
- The `unpublish` snapshot action removes the publish flag and removes the `published_date` value.

## Procedure

1. Use either of the following methods to publish a snapshot:
  - On any tab for an open snapshot, select **Publish**.
  - On the **Version** tab for an application, select **Publish** from the **Actions** menu for the snapshot.

2. View info

### Information on the snapshot header and Details tab

Changeset number	The system auto-assigns the changeset number. The system uses the number as the default changeset name.
Changeset name / Description	The system uses the changeset number as the default changeset name. Update the fields to help other users by changing to a more user-friendly name or by associating descriptive information with the changeset.
Deployable	Deployable on which the snapshot is based.
Validation status	<ul style="list-style-type: none"> <li>◦ Not validated: No policy has been executed against the snapshot.</li> <li>◦ Passed: The snapshot has passed all policy executions.</li> <li>◦ Failed: The snapshot has failed one or more policy executions.</li> <li>◦ In progress: Policies are currently executing.</li> <li>◦ Execution error: A policy failed to execute to completion.</li> </ul>
Created / Created by	User that created the changeset and timestamp` of creation.
Published on	Timestamp when the snapshot was published.

3. Use either of the following methods to unpublish a snapshot:

- On any tab for an open snapshot, select **Unpublish**.
- On the **Version** tab for an application, select **Unpublish** from the **Actions** menu for the snapshot.

## Compare snapshots from the same or different applications

Use the Config Data Analyzer tool to find similarities and differences between two snapshots from the same or different applications.

## Before you begin

Role required: cdm\_viewer, cdm\_editor, or cdm\_admin

## About this task

You can compare and contrast snapshots from any deployable of the same or different applications.

To see the changes between two snapshots from the same deployables, see [Compare two snapshots of a deployable](#).

## Procedure

1. While viewing an application, select **Compare config data**.

The **Compare config data** tab opens. The current application is the target application, and its name is pre-populated in the **Target application**.

2. Select the **Snapshot** option in the **Compare type** field.

3. Select an application and then their snapshots to compare.

Option	Description
Reference application	Application to compare with the target application. The field lists the latest committed data for all applications including the target application.
Reference snapshot	Snapshot of the reference application to compare.
Target snapshot	Snapshot of the target application against which the reference snapshot is compared.

4. Optional: If you've the cdm\_secrets role, you can select **View encrypted data** to view encrypted values in clear text in the comparison results.

**Note:** The check box is inactive for users who don't have the cdm\_secrets role. When the values in the target and reference differ, the comparison results indicate that the values differ, but the data itself still appears as \*\*\*\*\*.

5. Select **Compare**.

The Config data differences section displays the comparison results. It shows the differences between the selected reference and target snapshots.

**Important:** If you commit changes to a reference application while the **Compare config data** tab is open: When you return to the **Compare config data** tab and select **Compare**, the system uses the most recent committed changeset, performs the comparison again, and then displays the new results in the results section.

The letters in the following illustration identify the tools that you can use to analyze the data.

A: Navigation panel

The navigation panel displays the node structure of the applications.

- Nodes that include changed CDIs (either directly or in descendant nodes) are annotated with **Different**.
- Nodes that include newly added CDIs (either directly or in descendant nodes) are annotated with the identity of the application changeset that includes the new data, either **Reference** or **Target**.

You can search for a node name in the **Search** field. Select a node to view its contents in the Differences panel.

#### B: Differences panel

The **CDIs and variables** tab displays changes to individual CDIs for the node that is selected in the navigation panel.

- By default, the root node is selected and the data panel includes all CDIs for both changesets. Select a node in the navigation panel to display data for only that node and its descendants.
- When **Diff only** is selected, the number of CDIs that differ appears after the node path.
- Node paths are displayed in gray. Use the expansion icon () to view CDIs in a folder.

#### CDIs and variables tab

Column	Description
Differences	<p>Indicator to show where the differences are.</p> <ul style="list-style-type: none"> <li>◦ Target: The CDI appears only in the target changeset.</li> <li>◦ Reference: The CDI appears only in the reference changeset.</li> <li>◦ Different: For the specified CDI, the values in the target and reference changesets differ.</li> </ul>
Description	<p>Statement of how the changesets differ. For example, the text might indicate that a CDI appears only in the reference changeset or that the values for a variable differ between the changesets. See <a href="#">Types of differences between CDM applications</a> for the full list of difference types and their causes.</p>
Name	Name of the CDI.
Reference (R)	Value of the CDI in the reference changeset.
Target (T)	Value of the CDI in the target changeset.
Source level (R/T)	The node level in the code at which the CDI is defined.

Column	Description
	<ul style="list-style-type: none"> <li>◦ Direct: The CDI is defined in the selected node.</li> <li>◦ Include: The value of the CDI is obtained from a node that is included in the selected node.</li> <li>◦ Override: The value of the CDI in an included node is overridden by a value that is specified in the including node.</li> </ul> <p>Values appear in the following formats:</p> <ul style="list-style-type: none"> <li>◦ &lt;reference level / target level&gt; when a value appears in both target and reference.</li> <li>◦ &lt;reference level&gt; when a value appears in only the reference changeset.</li> <li>◦ &lt;target level&gt; when a value appears in only the target changeset.</li> </ul> <p>For example, when a CDI is defined in the reference changeset with the value true and the CDI doesn't appear in the target changeset, then the last three columns in the <b>CDIs and variables</b> tab will have the following values:</p> <ul style="list-style-type: none"> <li>◦ <b>Reference (R):</b> true</li> <li>◦ <b>Target (T):</b> [Empty]</li> <li>◦ <b>Source level (R/T):</b> Direct</li> </ul>

## Compare two snapshots of a deployable

Use the Config Data Analyzer tool to find changes between two snapshots of a deployable of an application.

### Before you begin

There must be more than one snapshot available in a deployable to compare and the target snapshot must not be the first snapshot in the deployable.

Role required: cdm\_viewer, cdm\_editor, or cdm\_admin

### About this task

To compare snapshots from different deployables or applications, see [Compare snapshots from the same or different applications](#).

### Procedure

1. While viewing an application, select the **Snapshots** tab.
2. Open a snapshot of a deployable.

3. In the Snapshot record page, select the **Config data changes** tab.  
The current snapshot is auto-populated in the **Target snapshot** field.
  
4. Select another snapshot in the **Reference snapshot** field.  
The **Reference snapshot** field lists all snapshots that are older than the target snapshot. The **Reference snapshot** field is inactive when older snapshots are not available. Even if newer snapshots created after the target snapshots exist in the deployable, they are not listed.
  
5. Optional: If you've the cdm\_secrets role, you can select **View encrypted data** to view encrypted values in clear text in the comparison results.

**Note:** The check box is inactive for users who don't have the cdm\_secrets role.  
When the values in the target and reference differ, the comparison results indicate that the values differ, but the data itself still appears as \*\*\*\*\*.

#### 6. Select **Compare**.

The Config data changes section displays the comparison results. It shows the changes between the selected reference and target snapshots.

The following image displays the Config data changes section that contains the comparison result that you can use to analyze the data.

##### A: Navigation panel

The navigation panel displays the node structure of the deployable.

- Nodes that include changed CDIs (either directly or in descendent nodes) are annotated with **Edited**.
- Nodes that include newly added CDIs (either directly or in descendent nodes) are annotated with **Added**.

You can search for a node name in the **Search** field. Select a node to view its contents in the Changes panel.

##### B: Changes panel

The panel displays a list of changes to individual CDIs for the node that is selected in the navigation panel.

- By default, the root node is selected and the data panel includes all CDIs for both snapshots. Select a node in the navigation panel to display data for only that node and its descendants.
- Node paths are displayed in gray. Use the expansion icon () to view CDIs in a folder.
- When **Changes only** is selected, the number of CDIs that have changes appears after the node path.
- When **Script view** is selected, the list view toggles to editor view.

##### List columns in the Changes panel

Column	Description
Action	Indicator to show what was the change type.

Column	Description
	<ul style="list-style-type: none"> <li>◦ Added: A CDI appears only in the target snapshot.</li> <li>◦ Deleted: A CDI appears only in the reference snapshot.</li> <li>◦ Edited: For the specified CDI, the values in the target and reference snapshots differ.</li> </ul>
Name	Name of the CDI.
Reference snapshot	Value of the CDI in the reference snapshot.
Target snapshot	Value of the CDI in the target snapshot.

## Export a snapshot

Export a snapshot to generate config data for the pipeline to use.

### Before you begin

- The cdm\_secrets role is required to export snapshots that include encrypted data.
- The cdm\_viewer can export any snapshot that does not include encrypted data.
- The cdm\_exporter\_editor can create, edit, publish, activate, and delete exporters.

Role required: cdm\_exporter\_editor or cdm\_editor or cdm\_admin

### About this task

- Exporters in the content pack have the **Source** value of **ServiceNow**. You can duplicate, but cannot delete or modify content pack exporters.
- You can execute only active published exporters.
- For export, snapshots cannot exceed 10,000 config data items (CDIs) per deployable or 100,000 CDIs per application.
- Records of exporter executions are deleted after a period of three years. For instructions on changing the default time period, see [Set the purge period for records of exporter executions](#).

### Procedure

1. Select the Admin icon () to open the **Administration** page.
2. On the **Exporters** tab, select the name of the exporter.

 **Tip:** Add commonly used exporters to the **My Exporters** list for quick access.

The **Versions** tab for the exporter opens, displaying the list of draft and published versions of the exporter.

3. Select a published version of the exporter to run.  
The exporter opens on the **Exporter Builder** tab. The text of the exporter script appears in a view-only panel. Use the Test playground panel to specify the snapshot to export and test settings before you export the config data.
4. Specify the settings for the exporter execution on the **Select inputs** page.

## Test playground settings

Field	Description
Deployable and published snapshot to export	<p>Deployable for export and the particular published snapshot that you have validated and want to use.</p> <p>Deployables appear in the list using the format &lt;AppName/DeployableName&gt;. When you select a deployable, another field lists all published snapshots for the deployable. The most recently published snapshot appears first in the list. Select the snapshot to export.</p>
Input arguments	The arguments that the exporter script uses to generate config data as output. The text box displays the default values for all arguments. Update values as needed.
Output format	Format in which to generate the config data.
Additional deployables	<p>Deployables for which to generate config data in addition to the primary deployable that you specified in the <b>Deployable and published snapshot to export</b> field.</p> <p>Only the latest published snapshots appear for this optional setting.</p>

**5.** Select **Save** to save your changes and then select **Evaluate**.

The exporter script runs using the settings that you specified and then displays the resulting config data in the Output box on the Review output page.

**6.** Review the results. To make changes, select **Select inputs**, specify new settings, and then select **Evaluate**. Repeat as often as needed.

When you are satisfied that the exporter generates correct and complete config data, you can publish it and activate it.

## Test an exporter and export a snapshot

Set and validate input settings to test an exporter before you export the config data in a snapshot.

### Before you begin

Role required: cdm\_exporter\_editor or cdm\_editor or cdm\_admin

- The cdm\_secrets role is required to export snapshots that include encrypted data.
- The cdm\_viewer can export any snapshot that does not include encrypted data.
- The cdm\_exporter\_editor can create, edit, publish and delete exporters.

## About this task

Be sure to install the exporter content pack for DevOps Config. Exporters in the content pack are good starting points for your custom exporters. Some content pack exporters generate subsets of config data and others generate data for full applications. For example, to export updated config data for a database, you might select the **returnAllDataforNodeName** exporter. In the input arguments for the exporter, you would specify the database deployable as the node and then specify the updated settings as input arguments. After you determine that the settings generate the desired config data, you can export the data to the pipeline.

- Exporters in the content pack have the **Source** value of **ServiceNow**. You can duplicate, but cannot delete or modify content pack exporters.
- You can run only published exporters.
- For export, snapshots cannot exceed 10,000 config data items (CDIs) per deployable or 100,000 CDIs per application.
- For information on creating a custom exporter, see [Create a custom exporter](#).
- Records of exporter executions are deleted after a period of three years. For instructions on changing the default time period, see [Set the purge period for records of exporter executions](#).

## Procedure

1. Select the Admin icon () to open the **Administration** page.

2. On the **Exporters** tab, select the name of the exporter.

**Tip:** Add commonly used exporters to the **My Exporters** list for quick access.

3. Select the exporter version on the **Versions** tab.

The exporter opens on the **Exporter builder** tab. The text of the exporter script appears in a view-only panel. Use the **Test playground** panel to specify the snapshot to export and test settings before you export the config data.

4. Specify the settings for the exporter execution on the **Select inputs** page.

### Test playground settings

Field	Description
Deployable and published snapshot to export	<p>Deployable for export and the particular published snapshot that you have validated and want to use.</p> <p>Deployables appear in the list using the format &lt;AppName/DeployableName&gt;. When you select a deployable, another field lists all published snapshots for the deployable. The most recently published snapshot appears first in the list. Select the snapshot to export.</p>
Input arguments	The arguments that the exporter script uses to generate config data as output. The

Field	Description
	text box displays the default values for all arguments. Update values as needed.
Output format	Format in which to generate the config data.
Additional deployables	<p>Deployables for which to generate config data in addition to the primary deployable that you specified in the <b>Deployable and published snapshot to export</b> field.</p> <p>Only the latest published snapshots appear for this optional setting.</p>

## 5. Select **Evaluate**.

- The exporter script executes using the settings that you specified.
- The **Test playground** panel displays the Review output page.
- The config data that the exporter generates appears in the **Output** box.
- The system lists information about the execution on the **Executions** tab.

## 6. Review the results.

- a. To make changes, select **Select inputs**, specify new settings, and then select **Evaluate**.

- b. Repeat the previous step as often as needed.

When you are satisfied that the exporter generates correct and complete config data, you can publish it and activate it.

## 7. Publish the exporter by selecting **Publish** on the **Exporter builder** tab.

**Note:** You must set an exporter to active to enable it to be executed in a pipeline.

### Generate API invocation code for an exporter

Generate sample code of an API invocation for an exporter. The code can invoke the exporter. Select from several commonly used languages.

#### Before you begin

Role required: cdm\_exporter\_editor or cdm\_editor or cdm\_admin

#### Procedure

##### 1. Prepare an exporter for execution as described in [Test an exporter and export a snapshot](#).

**Tip:** Test the exporter to verify its operation.

##### 2. In the **Evaluate** menu, select **Generate sample API**.

The Exporter API invocation sample dialog box opens.

##### 3. Select the language.

The system applies the settings that you specify and displays the resulting API sample in the text box.

##### 4. Select **Copy to clipboard**.

You can now paste the code into an editor for use in your code.

## Create a custom exporter

Copy an existing exporter script as a starting point for a custom exporter.

### Before you begin

Role required: cdm\_exporter\_editor or cdm\_editor or cdm\_admin

### About this task

DevOps Config content packs provide a variety of exporters. You can use any of the provided exporter scripts as a starting point for a custom exporter that you design.

### Procedure

1. Select the Admin icon () to open the **Administration** page.
2. On the **Exporters** tab, select **New**, enter a unique and meaningful name and description, and then select **Confirm**.

The first draft version of the exporter is named **Draft 0.1** and is listed on the **Versions** tab.

3. Select the **Version name** to open default exporter script on the **Exporter builder** tab.

The text of the exporter script appears in a view-only panel.

4. On the **Exporters** tab, select the name of the exporter that will act as the starting point for the new custom exporter.
5. Select the exporter version to use on the **Versions** tab for the exporter.

The exporter opens on the **Exporter Builder** tab. The text of the exporter script appears in a view-only panel.

6. Copy the text of the script for use in the new exporter.
7. Optional: To enable users to specify dynamic execution conditions or CDI values when editing the exporter, create an input argument.

- a. On the **Input arguments** tab, click **New**.

- b. On the Create argument pop-up window, define the argument.

**Create argument pop-up window**

Field	Description
(Optional) Name / Description	Meaningful name and description for the argument.
Default value	Value that the argument will have when a user opens the exporter for editing.
Mandatory	Select the check box to require that the argument value is specified.

- c. Select **Create**.

The argument now appears in the **Input arguments** field on the Test Playground panel.

### View the history of exporter executions

View a list of exporter executions to identify config updates that are successful and to isolate changes that caused failure in the pipeline.

**Before you begin**

Role required: cdm\_viewer or cdm\_exporter\_editor or cdm\_editor or cdm\_admin

**About this task****Procedure**

1. Select the Admin icon () to open the **Administration** page.
2. On the **Exporters** tab, select the name of the exporter.
3. Select the **Executions** tab to view the list of executions.

**Exporter executions list**

Column	Description
Number	Unique auto-generated identifier for the execution.
Version	Version of the exporter.
Type	<ul style="list-style-type: none"> <li>◦ Test: An execution in the test playground.</li> <li>◦ Standard: An execution that generates published config data.</li> </ul>
Executed by	User that executed the exporter.
Start time / Duration	Timestamp when the execution started and the period of time that the exporter ran.
State	Result of the execution.
Application / Deployable / Snapshot	The application, deployable, and snapshot that the exporter used.

**Delete an exporter**

Delete an exporter to ensure that users cannot use it to generate config data. To ensure that audit trails are protected, you cannot delete an exporter that has run a "standard" execution.

**Before you begin**

Role required: cdm\_exporter\_editor or cdm\_editor or cdm\_admin

**About this task**

- You cannot delete, modify, or activate exporters that are included in the CDM content pack. Exporters in the content pack have a **Source** value of **ServiceNow**.
- If an exporter has performed a standard execution, you cannot delete it. This ensures a complete and accurate audit trail.
- You can archive an exporter that has run a standard execution to ensure that users can no longer use it to generate config data.
- If an exporter does not have a standard execution, then you can delete all draft versions and published versions by deleting the exporter.
- You can delete any draft version of an exporter.

**Tip:** You cannot generate a draft version of a **ServiceNow** exporter, but you can duplicate one and edit draft versions of the duplicate. Editing a duplicate of a **ServiceNow** exporter is a good way to create a custom exporter. See [Create a custom exporter](#).

## Procedure

Delete all draft versions or only a particular draft versions of an exporter:

- To delete all draft versions of an exporter: While working on an exporter on the **Exporter builder** tab, select the exporter and then select **Delete**. All draft versions of the exporter are deleted and the exporter is no longer listed on the **Versions** tab.
- To delete only a particular draft version of an exporter: While viewing the list on the **Versions** tab, select the draft version, select **Delete draft**, and then confirm the action.

## Set the purge period for records of exporter executions

Change the default deletion period. By default, records of exporter executions are deleted after a period of three years.

### Before you begin

Role required: CDM Admin [sn\_cdm.cdm\_admin]

## Procedure

1. Navigate to **All > System Maintenance > Table Cleanup**.
2. In the **Tablename** column, select **sn\_cdm\_exporter\_execution**.
3. Change the **Age** setting to the retention period in seconds and then save the record.

Minimum: 1 year. For example:

- 1 year = 31 536 000
- 2 years = 63 072 000
- 3 years = 94 608 000

## Investigate an alert that involves a change to config data

A high percentage of alerts occur due to errors in config data. If the chain of events that resulted in an alert includes a change request that involves the same CI as the alert, then you can use a variety of tools to isolate the config changes that might have caused the alert.

### Before you begin

Roles required: Both cdm\_viewer and evt\_mgmt\_user.

### About this task

You work in the **Config changes** tab while investigating the causes of an alert. The tab includes several tools that enable you to visualize the series of config changes that might have led to the alert. You can filter, sort, and organize config data to help you identify problematic changes to the configuration data item key:value pairs (CDIs).

## Procedure

1. Use either of the following methods to view an alert:
  - On the home page () of the Service operations workspace, select an alert.
  - On the list page () of the Service operations workspace, navigate to **Alerts** and then select an alert.
 The top five most likely causes of the alert are listed in the Cause section on the **Overview** tab. If the chain of events that resulted in the alert includes a change request that involves the same CI as the alert, then that CHG record appears in the list.
  
2. On the **Changes** tab, select a change request.  
 The change request opens. If the change request involved a change to config data because a CDM snapshot was deployed, then the **Config changes** tab appears.
3. On the **Config changes** tab, work with the information in the Investigate configuration changes section.

### Investigate configuration changes section on the Config changes tab

Field	Description
Application	The CDM application that is affected by the alert.
Deployable	The CDM deployable whose snapshot was deployed.
Snapshot	<p>The CDM snapshot that was deployed.</p> <p>To open the snapshot on a new tab in the Service Operations Workspace, select the snapshot.</p>
<b>Snapshot deployment timeline</b>	
Date range / Start date / End date	<p>The time period that the timeline displays. By default, the timeline displays the period that includes the alert, the target snapshot (the snapshot that is associated with the change request), and the five most recently deployed snapshots.</p> <p>Specify a predefined period in the <b>Date range</b> box or use the calendar tools to specify custom start and end times.</p>
Timeline / Show legend	The timeline displays snapshot deployments and the alert.

Field	Description
	<p>Select <b>Show legend</b> to view the icons that identify snapshots and events on the timeline.</p> <p>In addition to the snapshot that immediately precedes the current (target) snapshot, you can compare the target snapshot to any earlier snapshot that appears on the timeline. To view additional snapshots, change the date range.</p> <p>Use the zoom icons () to shrink or grow the portion of the date range that appears on the timeline. Zoom does not change any of the other timeline settings. Use the left and right arrows to view items that have scrolled out of view.</p> <p>Select a snapshot to view its name, associated change record, and deployment date. Select the link to open the associated change request.</p> <p>Select an alert to view its alert ID and creation date.</p>
Reference snapshot / Target snapshot	<p>The target snapshot is the snapshot that is associated with the change request. (The alert can be associated with multiple change requests, each of which is associated with a target snapshot). Most often, the target snapshot immediately precedes the alert.</p> <p>By default, the system selects the snapshot that immediately preceded the target snapshot as the reference snapshot.</p> <p>While investigating the cause of the alert, you can use the <b>Reference snapshot</b> list to select any snapshot that precedes the reference snapshot in the selected date range on the timeline. (Snapshots on the timeline after the target snapshot do not appear in the list.) If you select a different reference snapshot, select <b>Compare</b> to generate the list of differences between the two selected snapshots. The difference information appears in the Configuration changes section.</p>

Field	Description
	<p>For any snapshot on the timeline, follow these steps to open it in the Service Operations workspace.</p> <ul style="list-style-type: none"> <li>a. Choose the target snapshot or select a snapshot in the <b>Reference snapshot</b> list.</li> <li>b. Select <b>Open snapshot</b>.</li> </ul>

**4.** Now work with the information in the Configuration changes section on the **Config changes** tab.

When the **Config changes** tab opens, the system immediately identifies the differences in the CDIs between the reference snapshot and the target snapshot. The Configuration changes section displays the differences.

The letters in the following illustration identify the tools you can use to analyze the data.

A: Navigation panel

The navigation panel displays the node structure of the snapshot. Select a node to view its contents in the data panel. Nodes that include changed CDIs (either directly or in descendent nodes) are annotated with **added**, **deleted**, or **edited**, as appropriate. Use the **Search** field to search for text in the navigation panel.

B: Data panel

The data panel displays groups of CDIs for the selected node. By default, the root node is selected in the node tree, and the list includes all CDIs for both snapshots. Select a node in the navigation panel to display CDIs for only that node and its descendants. You can switch from this list view of the config data to a script view, as described in [G: Script view](#).

Expand and close groupings with the expand icon (>). If a selection includes more than 50 CDIs, then CDIs are organized into groups of 50.

C: Diff only

Select **Diff only** to view only CDIs that differ between the two snapshots.

D: Filter the types of changes that should appear in the list

The condition builder icon () appears when you point to the **Actions** column name in the list view. Select the icon to specify the types of config data changes that should appear in the list.

#### "Types of changes" filters

Filter	Meaning
(empty)	[Not used.]
--	No difference in this CDI between the reference and target snapshots.
Added	The CDI was added to the target snapshot.

Filter	Meaning
Deleted	The CDI was deleted from the target snapshot.
Changed	The value of the CDI was changed in the target snapshot.

E: Filter the list based on CDI names and values

The condition builder icon () appears when you point to the **Key label** column in the list view. Select the icon to filter the list of config data changes that should appear in the list.

The sort indicator icon () indicates that CDIs are sorted alphabetically by the label of the key.

F: Find nodes with a particular kind of change

In the list view, select the find icon () and specify the type of change to isolate.

G: Script view

The data initially appears in list form. Select **Script view** to view the config data as code. Differences are indicated by symbols next to the line number. Deletions are indicated by -, additions are indicated by +, and edits are indicated by the pencil icon () plus notes in the in-line text, as shown in this "Diff only" example. Because the data structure of the `googleApiKey` CDI changed from a text value to an array, the text form is deleted and the array form is added.

## Analytics and reporting using the DevOps Config Insights dashboard

Use the DevOps Config Insights dashboard with Performance Analytics to quickly identify configuration errors and take action. View snapshot validation trend, open changesets, and failed snapshots for all applications and deployables.

Error thresholds are configurable and widgets are customizable (you can rearrange or hide them).

You can filter results on applications, deployables, and by date.

Report	Description and use case
Open changesets	<p>Open changeset details.</p> <p>Use case:</p> <p>As a DevOps engineer or App Engineer, I want to review my open changesets so that I can take action on them right away and not let them get stale.</p>

Report	Description and use case
	Accidentally committing changesets later could add risk to my data model and, by consequence, the fleet.
Failed snapshots	<p>Non-compliant snapshots for changes committed.</p> <p>Use case:</p> <p>As a DevOps engineer or App Engineer, I want to review non-compliant snapshots for changes I committed so that I ensure my changes are in line with company policies before pushing the data out to my production environment.</p>
Policy exception and recommended actions	<p>Recommended actions for policies that have exceptions.</p> <p>Use case:</p> <p>As a DevOps Config and GRC customer, I want to be able to see the list of active exceptions and, for ones that are expiring soon, what the recommended action is.</p>

## DevOps Config Insights Open changesets

Review the changeset details and either make changes to the config data and commit the changeset, or discard the changeset altogether. You generally want this number to be 0 on the dashboard.

## DevOps Config Insights Failed snapshots

Investigate the snapshots and fix misconfigurations, or identify changes that need to be made to a policy. You generally want this number to be 0 on the dashboard.

## DevOps Config Policy exceptions

### Possible actions

Nothing	If the issue is already resolved and the latest snapshot status passed; or withdraw the exception if it is no longer needed.
Fix the issue	If the latest snapshot validation status is passed with exception and there's time to fix the issue before the exception expires.

**Possible actions (continued)**

Extend the exception	If the latest snapshot validation status is passed with exception and there isn't enough time to fix the issue.
----------------------	---

## DevOps Config reference

DevOps Config reference content.

### Update the associations for a deployable

Update the name and description of a deployable as well as its associations with applications and other items. This procedure is optional.

#### Before you begin

Role required: cdm\_policy\_editor or cdm\_editor or cdm\_admin

#### About this task

This optional procedure explains how to update the details of a deployable definition. For information on configuring the data in a deployable, see [Create and update a deployable](#).

**Note:** You cannot change the SDLC environment type (development, test, or production) of a deployable because such a change would have significant impacts. For example, mapped policies and their inputs might differ significantly between test and production versions of a config dataset.

#### Procedure

1. On the **Settings** tab for an application, select **Deployables** in the tree view.
2. Select the check box for the deployable, select **Edit**, and then update the settings for the deployable.

#### Edit deployable settings

Field	Description
Name	Unique and meaningful name for the deployable.
CDM Deployable Node	Deployable in the dataset.
Node Main	
CDM Application	CDM application that includes the deployable.
Domain	
Enforce Compliance	Option to ensure that only snapshots that pass validation can be exported.
CMDB CI	Configuration item in the CMDB.
Environment	SDLC environment that the deployable implements.

Field	Description
	<ul style="list-style-type: none"> <li>◦ Development</li> <li>◦ Test</li> <li>◦ Production</li> </ul>
Description	Text description that helps other users understand the purpose, scope, and intent of the deployable.

### 3. Select **Update**.

Related topics

[Create or update a variable CDI](#)

## Types of configuration data supported by DevOps Config

DevOps Config stores configuration data of every layer of the IT stack in a centralized location.

Configuration data includes:

- App and service configuration
- Middleware configuration (databases, message queues, load balancers, for example)
- Kubernetes and Service Mesh configuration
- Cloud resources configuration (AWS, GGP, Azure DevOps, for example)
- Traditional infra configuration (servers, VMs, network, for example)

API keys	passwords	usernames	feature toggles
install paths	load balancing methods	database connections	heap sizes
scaling rules	geo regions	host names	IPs
cluster settings	sizing	sudo user lists	ssh
patching levels	firewalls	URLs	certificates
versions	build numbers	hot fixes	--

Typically configuration data is stored in files (.json, .yaml, .properties, .ini, .xml, .csv) and is located in a variety of places, including GIT repos, network folders, third-party tools (JFrog artifactory, Sonatype Nexus) other databases.

DevOps Config manages and validates your configuration data in a centralized location as the single source of truth.

## CDM data model

The CDM data model is a standardized data structure that supports the broader life cycle of software delivery — automation, quality validation, and CSDM. CDM imports existing config data, validates it using policies that you define, and exports valid config data to your organization's existing DevOps pipeline to implement applications, services, and infrastructure.

## Overview

The CDM data model does not alter the way you think about configuration. Rather, you use the CDM REST API and user interface to map your existing JSON, YAML, INI, XML, and other config data into an intuitive data structure that provides the following benefits:

- Implements rigorous and transparent version and change control.
- Enables you to encrypt sensitive data and ensures appropriate access control for the data.
- Enables automated validation of config data.
- Enables you to reuse config data structures by using variables, including values, and overlaying values.

## Definitions

### CDI

A config data item (CDI) is simple a key-value node.

### Variable

A variable is a key-value item that can be referenced within a CDI.

### Parent nodes and child (leaf) nodes

- CDIs and variables are key-value items. CDIs and variables can only be child nodes.
- Components, collections, deployables, and folders nodes can be parent nodes—nodes that can have key-value items or other parent nodes.  
Components, collections, and deployables are defined in the next section.

### Snapshot

When you commit changes to a CDM application, the system persists the changes and then generates a snapshot of every deployable that is affected by the changes. A snapshot represents a potential exportable config data set. The system validates config data by executing policies against each snapshot and returning the validation results. Snapshots that pass validation and that are published can be exported to the release pipeline as config data.

## Structure of the CDM data model

An application in CDM is the full collection of config data for an application service, application model, or dynamic CI group [infrastructure] in the CMDB. The CDM user creates an application record that includes the following empty folders in a standard hierarchical structure. After the system ingests your existing config data, you structure the data into components in the appropriate folder. You create collections of the components and then combine the collections into a deployable — a config dataset (for a development, test, or production environment) that can be deployed by your delivery process. Each component, collection, variable, and deployable is a node in the structure.

### Components

Components are the building blocks that typically represent the config data for a logical element of an application or a part of an infrastructure service.

For example, a monolithic app, a micro-service, a physical server, or a Docker template.

A component can include descendent components, either direct or included. A component can include variables that can take on different values in collections and deployables.

You can group components into a library of shared components.

**Tip:** It is often useful to define a default value for a variable in a component or collection. This is a powerful strategy because you can create a broad variety of deployables from a small set of components and collections. Deployables that inherit a component or collection can use overrides, overlays, and variable settings to meet the needs of the environment type. For example, the `Development` deployable can use the same components and collections as the `Test` deployable. `Development` uses the default `database` variable value. `Test`, in contrast, uses a different value that is appropriate for the test environment.

#### Components Vars folder

The components **Vars** folder can contain variables that any CDI in the **Components** folder can use. There is only one components **Vars** folder.

#### Collections

A collection is the set of components that together define a release — you can think of a collection as a release composition.

A collection can include variable or override settings that are specific to the particular version. For example, the VM config data used in `release-1` is different from the data used in `release-2`. `release-1` might use the value `2Gb` for the `memory` setting ("memory": "2Gb") and `release-2` might specify a different value ("memory": "4Gb"). In addition, a collection might include config settings that do not appear in its components. Such values are called overlays.

A collection might represent a particular version of an application, localization, or feature set. For example, a collection named `collection-2` might include the set of components or component versions that represent the `Release 2.0` functionality for the application. In contrast, a collection named `collection-3` that represents the `Release 3.0` functionality might include the same set of components or component versions, additional components or component versions, and other variable, override, and overlay settings.

#### Collections Vars folder

The collection **Vars** folder can contain variables that any CDI in the **Collections** folder can use. Each collection has one collection **Vars** folder. A collection variable has higher precedence than a component variable.

#### Deployables

You add and configure deployables in the data structure. A deployable is a config dataset (for a DEV, TEST, or PROD environment) that can be deployed by your delivery process. Each deployable in an application represents the configuration of a service in the CMDB.

A deployable is made up of the collection or set of collections that define the release for a particular environment. The combination of collections +environment link to an application service in the CMDB or to an infrastructure service.

A deployable can include variable or override settings that are specific to the environment. For example, the `database` variable has one value in the

development environment and a different value in the production environment. An override value in the production deployable might specify a required container parameter that is not needed in the development environment.

An example deployable named `DEV-2` would include the `collection-2` collection and would specify variable, override, and overlay settings that are specific to the development environment for release 2.0. In contrast, the deployable named `PROD-2` would also include the `collection-2` collection but, instead, would specify settings that are specific to the production environment for release 2.0.

When you are happy with a changeset, you can save and commit the changes. The system checks for conflicts with the committed changesets of other users. If there are no conflicts, the system persists the changes and then generates a snapshot of every deployable that is affected by the changes. A snapshot represents a potential exportable config data set. The system validates config data by executing policies against each snapshot and returning the validation results.

#### Deployable Vars folder

The deployable **Vars** folder can contain variables that any CDI in the **Deployables** folder can use. Each deployable has one deployable **Vars** folder. A deployable variable has higher precedence than a collection variable.

### Example

In the following diagram of the example BookStore application, the numbers identify the relationships among components, collections, and deployables.

1. Components are grouped to form collections that represent environments or versions of environments. The `FS2` (feature set 2) collection has config data for Core version 2 of the application that is currently being developed and tested. `FS1`, in contrast, holds the earlier Core version 1 that has been thoroughly tested and is currently running the application in the production environment.
2. In the example, both `FS2` (the collection that is used in test environments) and `FS1` (the collection that is used in the production environment) use config data for both `s3` and a particular *VM template*. Both the `FS1` and `FS2` collections, therefore, inherit these two components. Because the collections represent different feature sets, it is probably the case that `FS1` and `FS2` use variables or overrides to specify a few different settings for the components.
3. Each deployable includes the collection that is appropriate for its environment (development, test, or production). In the example, the `TEST` deployable uses the `FS2` collection, the newer version of the feature set and other config settings that is used in test environments. The `PROD` deployable, in contrast, uses `FS1` in the production environment. `FS1` is the earlier version of the collection of config data that had been validated for production.

In each deployable, variables are set to values that are appropriate for the environment. For example, in `PROD`, the `database` variable is set to `prod1` (the production database). The `TEST` deployable, however, specifies one of the databases that is used by the testing team, `test3`.

This diagram is simplified. In your implementation, deployables can include multiple collections, variable and override settings, and overlay settings (settings that do not appear in the components and collections that make up the deployable). In addition, you might have several deployables for each environment type.

## Related topics

[Preparing an application for config data upload](#)

[Create and update a deployable](#)

## APIs and DevOps Config

You can use DevOps Config and CDM APIs to access your config data.

### DevOps Config

Manage your application lifecycle, using delete, get, patch, and post operations.

### CDM

#### [CdmApplicationsApi](#)

Upload configuration data to the component, collection, deployable, and component variable folders found in the DevOps Config Workspace UI. Export deployable configuration data to your DevOps pipeline and manage shared components and shared applications.

#### [CdmChangesetsApi](#)

Manage your changesets, including:

- Create new changesets.
- Deploy changesets.
- Retrieve lists of or individual changesets.
- Retrieve node changes within a changeset.
- Retrieve a list of applications or deployables impacted by a changeset.
- Delete changesets.
- Return a list of shared components associated with a specified changeset.

#### [CdmEditorApi](#)

Create nodes, update nodes, include existing nodes under other nodes, delete nodes, and retrieve nodes and node includes.

#### [CdmPoliciesApi](#)

Manage policy mappings of deployables in CDM. Policies that are properly mapped to a deployable are executed when a snapshot of the deployable is validated.

#### [CdmSnapshotApi](#)

Publish, unpublish, and revalidate snapshots in CDM.

## DevOps Config roles

DevOps Config uses these roles.

Role title [name]	Description	Contains Roles
[sn_devops.app_admin]	DevOps Config app admin  Can create/read/update/delete DevOps Config apps.	None
[sn_devops_config.viewer]	DevOps Config viewer  Has read-only access in DevOps Config.	sn_cdm.cdm_viewer
[sn_devops_config.editor]	DevOps Config editor  Can manage config data, create/commit changesets, and publish/unpublish snapshots.	<ul style="list-style-type: none"> <li>• sn_devops_config.viewer</li> <li>• sn_cdm.cdm_editor</li> </ul>
[sn_devops_config.secrets]	DevOps Config secrets  Able to manage encrypted config data within an application.	sn_cdm.cdm_secrets
[sn_devops_config.admin]	DevOps Config admin  Has editor role permissions plus ability to manage policies and exporters, and applications.	<ul style="list-style-type: none"> <li>• sn_devops.app_admin</li> <li>• sn_devops_config.editor</li> <li>• sn_cdm.cdm_admin</li> </ul>

## Roles in CDM

List of roles and permissions in CDM.

## CDM roles

Role title [name]	Permissions	Contains roles
CDM Viewer [sn_cdm.cdm_viewer]	<ul style="list-style-type: none"> <li>• Read config data from any application that they have access to (governed through user groups that are set by the <b>Maintained by</b> property).</li> <li>• View the list and content of component libraries as well as the shared components contained within them.</li> <li>• View list and content of changesets.</li> </ul>	<ul style="list-style-type: none"> <li>• [sn_pace.policy_reader]</li> <li>• [itil]</li> <li>• [canvas_user]</li> </ul>

Role title [name]	Permissions	Contains roles
	<ul style="list-style-type: none"> <li>View list and content of snapshots and validation results.</li> <li>Export snapshots.</li> <li>View exporters.</li> <li>View policies and policy mappings.</li> <li>View the Investigate page for a change request (CHG) on the Service Operations Workspace.</li> </ul> <p><b>i Note:</b> If the <b>Maintained by</b> group is set at the application level to view config data, then this user must be a member of the group.</p>	
Event Management user [evt_mgmt_user]	<ul style="list-style-type: none"> <li>View contents of snapshots.</li> <li>View the Investigate page for a change request (CHG) on the Service Operations Workspace.</li> <li>View snapshots, nodes, and changesets, regardless of whether this user is a member of <b>Maintained by</b> groups set at the application level.</li> </ul>	
CDM Editor [sn_cdm.cdm_editor]	<ul style="list-style-type: none"> <li>Create/update/delete config data within components and collections, including variables, overrides, and includes.</li> <li>Create and commit changesets.</li> <li>Validate snapshots.</li> <li>Publish and unpublish snapshots.</li> <li>Create, update, and delete config data withing CDM applications.</li> </ul>	cdm_viewer

Role title [name]	Permissions	Contains roles
	<ul style="list-style-type: none"> <li>• Add and manage component libraries.</li> <li>• Add and delete shared components in a component library.</li> </ul> <p><b>i Note:</b> The cdm_editor role does not grant permission to create/update/delete an application and its deployables, or to change the <b>Enforce validation</b> setting on deployables.</p> <p>If the <b>Maintained by</b> group is set at the application level to view config data, then this user must be a member of the group.</p>	
CDM Exporter Editor [sn_cdm.cdm_exporter_editor]	Create/update/delete exporters.	cdm_viewer
CDM Policy Editor [sn_cdm.cdm_policy_editor]	<ul style="list-style-type: none"> <li>• Create/update/delete policies.</li> <li>• Map policies to deployables.</li> </ul>	<ul style="list-style-type: none"> <li>• cdm_viewer</li> <li>• [sn_pace.admin]</li> </ul>
CDM Secrets [sn_cdm.cdm_secrets]	<ul style="list-style-type: none"> <li>• Read and export encrypted data (when granted to a user with the cdm_viewer role).</li> <li>• Permanently encrypt / decrypt data (when granted to a user with the cdm_editor role).</li> <li>• Edit encrypted data (when granted to a user with the cdm_editor role).</li> </ul> <p><b>i Note:</b> The cdm_secrets role is effective only with the cdm_viewer, cdm_editor, or cdm_admin role.</p>	None

Role title [name]	Permissions	Contains roles
Application Service Admin [sn_cdm.app_service_admin]	Enables the CDM Admin to create an application service.	None
CDM Admin [sn_cdm.cdm_admin]	<ul style="list-style-type: none"> <li>• Create/update/delete applications.</li> <li>• Create/update/delete deployables.</li> <li>• Create/update/delete config data.</li> <li>• Change settings on deployables to enforce snapshot validation.</li> </ul>	<ul style="list-style-type: none"> <li>• cdm_editor</li> <li>• cdm_exporter_editor</li> <li>• cdm_policy_editor</li> <li>• app_service_admin</li> <li>• Model_manager (for create/update/delete of application model)</li> <li>• [itil] (for create/update/delete of SDLC components)</li> <li>• [itil admin]</li> </ul>

## How encrypted data is handled

By default, encrypted values appear in all views as \*\*\*\*\*. Only users with the CDM Secrets [sn\_cdm.cdm\_secrets] role can view, create, update, and delete encrypted values in config data. The system validates and exports encrypted data appropriately.

### Permissions of users with the CDM Secrets [sn\_cdm.cdm\_secrets] role

- Encrypted values appear in all views as \*\*\*\*\* by default.
- Select the **View encrypted data** menu option to display encrypted values in any view.
- Validate and export a snapshot that includes encrypted data. All exported data is readable by the CI/CD pipeline apps.
- Create, update, delete abilities:
  - View an encrypted value.
  - Edit an encrypted value.
  - While creating a CDI, specify that the value should be encrypted.
  - Permanently encrypt a value that is currently not encrypted.
  - Permanently decrypt a value that is currently encrypted.
  - Delete a CDI that has an encrypted value.

### Permissions of all other users

- Encrypted values always appear in all views as \*\*\*\*\*.
- Request manual validation of snapshots that include encrypted data.
- Delete encrypted values in config data.
- Cannot view, create, or update encrypted values in config data.
- Cannot export a snapshot that includes encrypted data.

## Domain separation and DevOps Config

Domain separation is unsupported for DevOps Config. Domain separation enables you to separate data, processes, and administrative tasks into logical groupings called domains. You can control several aspects of this separation, including which users can see and access data.

### Support level: No support

- The domain field may exist on data tables but there is no business logic to manage the data.
- This level is not considered domain-separated.

For more information on support levels, see [Application support for domain separation](#).

## Overview

The DevOps Config application validates and manages the configuration data of your enterprise applications across every stage of the DevOps pipeline. Domain separation is supported at the data segregation layer only.

Related topics

[Domain separation for service providers](#)

## Domain separation and CDM

Domain separation is supported for CDM. Domain separation enables you to separate data, processes, and administrative tasks into logical groupings called domains. You can control several aspects of this separation, including which users can see and access data.

### Support level: Basic

- Business logic: Ensure that data goes into the proper domain for the application's service provider use cases.
- The application supports domain separation at run time. The domain separation includes separation from the user interface, cache keys, reporting, rollups, and aggregations.
- The owner of the instance must set up the application to function across multiple tenants.

Sample use case: When a service provider (SP) uses chat to respond to a tenant-customer's message, the customer must be able to see the SP's response.

For more information on support levels, see [Application support for domain separation](#).

## Overview

Domain separation for CDM means that only a user in the application's domain or parent domain can access the data. All functionality for CDM works in a domain-separated environment as long as all of the following actions are performed by users within that domain:

- Create application
- Create/upload config data
- Commit changeset

- Create snapshots
- Validate snapshots
- Map policies
- Export snapshots

## How domain separation works in CDM

To set up domain separation for CDM:

1. Create your domain and then create CDM users in that domain.
2. Only a specified user (CDM admin in that domain) should populate config data in the application's domain.
3. Only users in the application's domain should upload or create data in the domain.

Follow these guidelines for successful domain separation:

- Create a separate user for each domain to perform upload, edit, or export of config data.
- Only users in the application's domain should upload or create data in the domain.
- Avoid uploading or creating config data for a given application in multiple domains.
- Policies and exporters must be either in the application's domain or in the global domain.

## Domain-separated tables in CDM

CDM tables include a **Domain** column.

- sn\_cdm\_application
- sn\_cdm\_changeset
- sn\_cdm\_deployable
- sn\_cdm\_exporter
- sn\_cdm\_exporter\_execution
- sn\_cdm\_exporter\_execution\_log
- sn\_cdm\_exporter\_version
- sn\_cdm\_exporter\_version\_argument
- sn\_cdm\_node
- sn\_cdm\_node\_main
- sn\_cdm\_pace\_policy\_mapping
- sn\_cdm\_restricted\_groups
- sn\_cdm\_snapshot

Related topics

[Domain separation for service providers](#) 

## Administering policies in DevOps Config

You can administrate the full life cycle of policies in DevOps Config by searching, reviewing, and updating existing policies, and creating policies as required. Policies in DevOps Config are powered by PACE.

## Understanding PaCE

PaCE enables you to manage, administer, and audit places from a centralized location.

A PaCE policy is a set of pre-defined rules and logic that determines the desired behavior of an application or a service. When invoked, the rules in the policy are applied on the provided input, and a decision is reached. This decision-making is the main function of PaCE and helps determine if a policy is compliant or non-compliant. The decision is then relayed to the software calling service or application, so that it can act on it to enforce a desired behavior.

PaCE provides the following capabilities:

- Full life cycle management of policies
- Policy reuse
- Audit and compliance
- Testing and validation of policies
- Central automation of compliance and regulatory processes

PaCE can be used to:

- Identify posture drifts from a desired state in the current application.
- Make decisions (compliant or non-compliant) based on a change in the application or service and enforce the decision to prevent a drift.
- Automate execution of policies and eliminate dependency on humans.
- Standardize policies so that they can be shared and reused within a service and across services.
- Increase change velocity while including guardrails with automated workflows to provide preventive controls.
- Collect evidence and proof of compliance for audit purposes. This feature can be used by internal auditors to automate the process of collecting evidence for governance and risk requirements.
- Provide business context to PaCE policies by using control objectives to connect PaCE with the Integrated Risk Management and Policy and Compliance Management workspace.

PaCE enables policy developers to view and understand an existing policy, make and assess changes, and decide if a policy can be used as a baseline for another policy. They can also use PaCE before debugging to understand how the policy should work and why it is not working as expected.

PaCE provides a centralized platform for storing, managing, and using policies. By using PaCE, policies can be:

- Well documented and all the requirements clearly defined.
- Reused across the organization.
- Tracked and new versions can be created when a policy is changed.
- Tested and validated before deployment.

Policies are also containers for all the elements that make up a policy. These elements include meta-data related to the policy, the policy versions (including policy scripts and inputs), mapping information, and policy execution history.

You can define any number of policies within PaCE.

## Key user personas and roles

This section describes different user personas and roles in PaCE. These personas are defined with the application where PaCE is being used.

All roles except the super administrator role must be assigned to a calling service or application where PaCE is being used. The assigned calling service defines the scope for the user role.

Role	High-level Permissions	Persona
sn_pace.execution_reader	A read-only user with view-only access. This user can view policies, categories, and executions.	Policy user, internal auditor.
sn_pace.code_reader	Can review PaCE versions, policy code, and run tests.	Internal auditor
sn_pace.code_editor	This user has all the sn_pace_code_reader permissions plus the ability to create PaCE policy versions.	Policy developer
sn_pace.policy_reader	This user has all the sn_pace_code_reader permissions plus the ability to review policy details and mapping information.	Policy user, internal auditor
sn_pace.policy_editor	This user has all the sn_pace_policy_reader and sn_pace.code_editor permissions plus the ability to create policies and mappings.	Policy developer
sn_pace.admin	This user has the permissions of all the other roles plus the ability to create categories, policies, mappings, and code.	Policy admin
sn_pace.super_admin	This user has all the sn_pace.admin role permissions across all calling services.	Not applicable
Maint role	Internal user who can create default content.	Not applicable

## PaCE administration life cycle

This checklist lists the tasks involved in administering PaCE policies.

Task	Description
Set up and install PaCE.	PaCE is installed as part of the DevOps Config application. See <a href="#">Install DevOps Config</a> for details.
Create a policy.	See <a href="#">Create a PaCE policy</a> .
Create a new policy version	See:

Task	Description
	<ul style="list-style-type: none"> <li>• (if required).</li> <li>• Define mappings for a PaCE policy.</li> <li>• Edit a PaCE policy script</li> </ul>
Test a policy version	Use the Test Playground for PaCE policies
Publish a new policy version	See <a href="#">Publish a PaCE policy version</a> .
Activate a published policy	See <a href="#">Activate or deactivate a PaCE policy</a> .
Use the policy.	See <a href="#">Execute policies</a> .

## Install PaCE

Install PaCE.

PaCE is installed with the DevOps Config. See [Install DevOps Config](#) for details.

## Administer PaCE policies

Use PaCE to manage the life cycle of a policy and create, update, review, and execute policies.

In addition, you can:

- Evaluate any changes to ensure they meet your organization's policy management requirements.
- Create a policy, modify policy information, copy a policy and its version to a new policy.
- Create a new policy version by copying an existing policy version and modifying it.
- View and define mapping information for any policy.
- Review executed policy activity and execution history for policies and policy versions.
- Use tags and categories to manage your policies more efficiently.

**Note:** You can create any number of policies, which can be updated, activated, or deactivated as required.

## Manage PaCE policies

You can define and manage your PaCE policies in a single management console.

Policies are containers for all the elements that make up a policy. These elements include metadata related to the policy, the policy versions (including policy scripts and inputs), mapping information, and policy execution history.

You can easily create policies, and search and review existing policies. In addition, you can:

- Assess and implement changes to each policy.
- Determine if a policy should be used as the template for another policy.
- Understand why a policy may not be working as expected.

You can create any number of policies, which can be updated as required.

## View existing PaCE policies

You can view a list of all PaCE policies created by you and other policy administrators. The PaCE management console is your gateway to viewing and updating policies, and managing policy versions.

### Before you begin

Role required: sn\_pace.admin

### Procedure

#### 1. Navigate to All > Policies > My Policies.

**Note:** To view all policies generated by admin/other users, navigate to **Policies > All Policies**. These policies are viewable to all logged in users.

On this page, you can do the following:

- [Create a PaCE policy](#)
- [Edit a PaCE policy](#)
- [Delete a PaCE policy](#)

The My Policies list displays the following columns.

Column	Description
Policy name	The name of the policy as defined by the user.
Owner	The owner of the policy.
State	<p>The current state of the policy:</p> <ul style="list-style-type: none"> <li>◦ Active: The policy is <b>Active</b> and can be used. For details on how to activate a policy, see <a href="#">Activate or deactivate a PaCE policy</a></li> <li>◦ Inactive: The policy is <b>Inactive</b>, and must be activated before it can be used.</li> </ul> <p><b>Note:</b> Each policy can only have one Current (published) policy version. Only policies that have a Current version can be activated.</p> <p>See <a href="#">Manage PaCE policy versions</a> for details.</p>
Category	<p>The category assigned to the policy. Only one category can be assigned to a policy. See <a href="#">Managing categories in PaCE</a> for details</p>
Content source	The source used to create the policy:

Column	Description
	<ul style="list-style-type: none"> <li>◦ User: This policy was created or copied by a user.</li> <li>◦ ServiceNow: This policy was shipped as part of the PaCE content pack.</li> </ul>

2. Optional: Use the Now Platform filter controls to filter the displayed list.
3. To view the details of a policy, click the required row in the **Policy Name** column.

**i Note:** Alternatively, click the information icon to see a quick glance view of the policy.

### Create a PaCE policy

You can create PaCE policies to determine if changes to a software service or application comply with a set of pre-defined rules.

#### Before you begin

Role required: sn\_pace.admin

#### About this task

A decision is then made about these changes, and if they are compliant, non-compliant, or compliant-exception according to these pre-defined rules. The policy must be mapped correctly to the relevant object (table and document ID) to invoke the policy when any changes are made to the object.

**i Note:** Policies are only used if they have active mapping records, and if there was a request to validate an object (tables and document IDs) mapped to the policy. For further information, see [Map PaCE policies](#).

#### Procedure

1. Navigate to **All > Policies > My Policies**.

**i Note:** To view all policies generated by admin/other users, navigate to **Policies > All Policies**.

2. Click **New**.
3. In the Create New Policy form, fill in the fields.

Field	Description
Policy name	<p>The policy name.</p> <p><b>i Note:</b> The policy name must be unique, and is used as the identifier of the policy.</p>
Category	Select a category from the list. Categories enable you to group and manage policies more efficiently. See <a href="#">Managing categories in PaCE</a> for details.
Created	This field is auto-generated and shows the date and time the policy was created.

Field	Description
Updated	This field is auto-generated and shows the date and time the policy was updated.
Description	Add additional details for this policy.

4. Click **Save**. The form is refreshed to display the newly created policy.
5. Optional: Click the tag icon alongside the policy name to add tags. For more information about defining tags, [Add tags to PaCE policies](#).
6. The newly created policy contains the following tabs.

Tab name	Description
Details	<p>The details of the policy including the policy name, category, date on which it was created, and a description is displayed.</p>
Versions	<p>When you create a policy, by default a draft policy version is created. Each policy version contains version meta-data, a policy script, and variable input definitions, which you can modify according to your requirements. Click the <b>Versions</b> tab to view the Versions page. You can:</p> <ul style="list-style-type: none"> <li>◦ Edit a policy version.</li> </ul> <p><b>i Note:</b> You can edit a policy version only if it is a Draft state.</p> <ul style="list-style-type: none"> <li>◦ Create a new version</li> <li>◦ Duplicate policy versions</li> <li>◦ Compare versions</li> </ul> <p>For more details, see <a href="#">Manage PaCE policy versions</a>.</p> <p><b>i Note:</b> You must publish the (See <a href="#">Publish a PaCE policy version</a>) policy version to make it <b>Current</b> before it can be used.</p>
Mapping	Click the <b>Mapping</b> tab to define the object (table and document ID) to which the policy is to be mapped. For more information, see <a href="#">Map PaCE policies</a> .
Executions	Click the <b>Executions</b> tab to review the execution activity for the policy.

7. You can perform the following actions:

- Delete a policy that is in an inactive state. See [Delete a PaCE policy](#).
- Duplicate or create a copy of an existing policy.

### Activate or deactivate a PaCE policy

You can activate or deactivate any PaCE policy. Only activated policies can validate object (tables and document IDs) changes. Deactivated policies are not executed, even if the policy is mapped. Deactivating a policy enables you to place the policy into a maintenance-like mode.

**Before you begin**

Role required: sn\_pace.admin

**About this task**

You can deactivate or activate any policy at any time.

For activated policies:

- The policy must include a Current (published) version. For more information, see [Manage PaCE policy versions](#).
- The policy is auto-validated when you activate it (if the policy has mapping inputs defined). Validation verifies the validity of mapping inputs, and checks that all required inputs were set. For more information, see the Validate mapping inputs section.

Deactivating a policy enables you to place the policy into a maintenance-like mode. You can make changes, safe in the knowledge that no one is using the policy. While deactivated, the policy mappings are kept, but the policy cannot be executed. After you are done with any changes, you can reactivate the policy and any existing mappings are reused.

**Note:** The activation or deactivation of a policy is audited for tracking purposes. This audit information is included in the standard Now Platform auditing records.

**Procedure**1. Navigate to **All > Policies > My Policies**.

**Note:** To view all policies generated by admin/other users, navigate to **Policies > All Policies**.

## 2. Select the check box next to the policy that you want to activate or deactivate.

**Note:** To activate a policy, its current state must be . To deactivate a policy, its current state must be . You can select multiple policies, as required.

## 3. Activate or deactivate the selected policy.

Task	Description
Activate a policy	<p>a. Select an inactive policy and click <b>Activate</b>.</p> <p><b>Note:</b> If there is no Current (published) version for the selected policy, you must first publish a version before activating. For more information about publishing a policy version, see the Managing policy versions section.</p> <p>b. In the displayed confirmation message, click <b>Activate</b>.</p> <p>Validation of the activated policy and its associated mapping inputs is performed automatically. In addition, the policy is</p>

Task	Description
	<p>added to the list of active policies in the All Policies page.</p> <p><b>i Note:</b> If the mapping inputs are invalid, you are redirected to the Mappings tab to define valid mapping inputs. For example, you can define required mapping inputs that do not include default values.</p>
Deactivate a policy	<p>a. Select an active policy, and click <b>Deactivate</b>.</p> <p>b. Click <b>Deactivate</b> to confirm.</p> <p><b>i Note:</b> If a policy is set as inactive, its mapping state is also set as inactive. As a result, the policy cannot be executed.</p>

### Duplicate a PaCE policy

Create a duplicate of an existing policy.

#### Before you begin

Role required: sn\_pace\_admin

#### Procedure

1. Navigate to **All > Policies > My Policies**.

**i Note:** To view all policies generated by admin/other users, navigate to **Policies > All Policies**.

2. Click the **Policy name** link for the policy that you want to duplicate.
3. In the Details page, click the **More Actions** icon and select **Duplicate**.
4. In the Duplicate Policy window, select the version of the policy you want to copy and click **Duplicate**.
5. A new policy with the same parameters as the selected policy is created in an **Inactive** state. You must publish a version of the new policy before you can activate it.

### Edit a PaCE policy

You can update any existing PaCE policies, whether they are activated or deactivated.

#### Before you begin

Role required: sn\_pace.admin

#### Procedure

1. Navigate to **All > Policies > My Policies**.

**i Note:** To view all policies generated by admin/other users, navigate to **Policies > All Policies**.

2. Click the information icon located to the left of the policy name to see a quick glance view of the policy. The policy details display to the right of the My Policies list and can be edited. You can also review the **Activity** stream to see a list of all changes made to the policy. This list can be filtered as required.
3. Edit the relevant details for the selected policy, and click **Save**.
4. To update existing versions or add a new version, click the **Policy name** link on the All Policies page. Then click the **Versions** tab, create a copy of the version in which you want to make changes, and make the relevant changes in the new version. For more information about creating and managing versions, see [Manage PaCE policy versions](#).

**i Note:** You can't update API Variables and Config Parameters for an existing policy version that is Current (published) or archived. To modify a Current version, create a copy of that version, as described in the Create a copy of a policy version section.

5. To create mappings or update existing ones, click the **Policy name** link on the All Policies page. Then click the **Mappings** tab and make the relevant changes. For more information about defining caller and mapping inputs, see [Map PaCE policies](#).

### Delete a PaCE policy

You can delete a PaCE policy at any time. All related versions for that policy are also deleted. However, for auditing reasons you cannot delete policies that were or are currently in use.

#### Before you begin

Role required: sn\_pace.admin

#### About this task

Policies and their related versions can be deleted. However, policies that are in use, or were used can't be deleted, including:

- Policies that are mapped to an object.
- Policies that were activated and executed in Standard mode. These policies, and their policy versions, include execution data critical for auditing purposes, and therefore cannot be deleted.
- Policies that have a version in either **Current** or **Archived** state.

#### Procedure

1. Navigate to **All > Policies > My Policies**.

**i Note:** To view all policies generated by admin/other users, navigate to **Policies > All Policies**.

2. Click the **Policy name** link for the policy that you want to delete.

**i Note:** You can only delete individual policies. However, you can select multiple policies to activate or deactivate. For further information, see the Activate or deactivate a policy section.

3. In the displayed policy form, click **Delete**.

4. Click **Delete** to confirm. The policy versions that are to be deleted are displayed.

## Duplicate PaCE policy versions

You can create a duplicate policy from a policy version or multiple versions from two or more policy versions. If you want to use a locked policy, you must first create a duplicate version and then modify the policy version as required.

### Duplicate a single policy version

To create a duplicate policy from a policy version, follow these steps:

1. Navigate to **All Policies > My Policies**.

**Note:** To view all policies generated by admin/other users, navigate to **Policies > All Policies**.

2. Select the name of the policy that you want to duplicate.
3. In the Policy Details page, click the **Versions** tab and select the version you want to duplicate.
4. Click the **More Actions** icon and select **Duplicate**.
5. Select an option in the **Duplicate version** form.

◦ To new policy:

- a. Select this option to create a duplicate policy from this version.
- b. Enter the policy name, category, and description.
- c. Click **Duplicate**.

The newly created policy is a duplicate copy and contains the same parameters and mappings of the selected policy version.

◦ To existing policy:

- a. Select this option to create a duplicate version for an existing policy.
- b. In the Duplicate Version form, select the policy to which the version is to be duplicated.
- c. Click **Duplicate**.

The newly created duplicate version is displayed in the Versions page of the selected policy.

### Duplicate multiple policy versions

1. Navigate to **All Policies > My Policies**.

**Note:** To view all policies generated by admin/other users, navigate to **Policies > All Policies**.

2. Select the name of the policy that you want to duplicate.
3. In the Policy Details page, click the **Versions** tab.
4. Select the policies that you want to duplicate and from the drop-down list, select **Duplicate versions**.
5. Select an option in the **Duplicate version** form and follow the remaining steps as listed in [Duplicate a single policy version](#).

## Map PaCE policies

For a policy to be invoked correctly, it must be mapped to an object (table or document ID). The defined mapping settings are automatically verified when the policy is invoked.

Mapping a policy to an object enables you to verify updates on that object, whenever a change in the provided inputs occurs.

### Define mappings for a PaCE policy

You can map a policy to any number of objects. When you map a policy, validate the inputs, and execute the policy, a compliant, non-compliant, or compliant-exception decision is returned.

#### Before you begin

Role required: sn\_pace.admin

#### Procedure

1. Navigate to **All > Policies > My Policies**.

**Note:** To view all policies generated by admin/other users, navigate to **Policies > All Policies**.

2. Click the **Policy name** link for the policy that you want to update.

3. Click the **Mappings** tab.

If there are any existing mappings for the selected policy, they are displayed.

4. To add a new mapping input, click **Add**.

Any objects that are available for mapping are displayed.

5. Select the check box to the left of each object that you want to map the policy to, and click **Map Policy**.

The new mapping input is added to the list of policy mappings. In addition, an alert message indicates whether the mapping is valid and active. For information about editing existing mapping inputs, see [Edit mapping inputs](#).

#### Edit mapping inputs

You can edit the mapping settings that map a policy to an object.

#### Before you begin

Role required: sn\_pace.admin

#### Procedure

1. Navigate to **All > Policies > My Policies**.

**Note:** To view all policies generated by admin/other users, navigate to **Policies > All Policies**.

2. Click the **Policy name** link for the policy that you want to update.

3. Click the **Mappings** tab, and click the Mapping ID link in the Input field.

4. Click the mapping input that you want to update.

In the displayed mapping input form, edit the details as required, and click **Save**.

## Manage PaCE policy versions

PaCE policy versions contain the actual policy script, as well as API Variable and Config Parameter definitions used for validating policies. Each policy can have many versions, each of which can be easily managed.

You can create a new policy version, search and review existing versions, and test changes to the policy script before it is deployed in a production environment. You can also compare policy versions, copy a policy, and modify it as required.

**i Note:** To activate a policy, a policy version must be created, and published to be in the **Current** state.

### PaCE policy version states

A PaCE policy version can be in one of four states: Draft, Current, Archived, and Now-Content. The policy version state also defines the version numbering.

Only one policy version in each policy can be Current at any one time. There is no limit to the number of Draft or Archived policy versions.

### Policy version states

State	Description
Draft	When a version is created, its initial state is Draft. The new Draft version can be saved as a Draft for later use, or published to become the Current version. The draft versions cannot be used until they are published.
Current	<p>Draft versions can be published to Current, which means the former draft version becomes the default published version used for policy execution.</p> <p><b>i Note:</b> When publishing the version, if mapping exists, verification is performed to confirm that all required inputs for mapping inputs were provided. If you update the mapping inputs, you must update the inputs in the Mapping screen. If you updated caller inputs, you must update the PaCE API call to ensure that the policy is executed.</p>
Archived	<p>A published Current version becomes Archived when a different version is published. Only Current versions can be Archived. If necessary, you can also republish an Archived version.</p> <p><b>i Note:</b> When a version is published, it becomes the Current version irrespective of its previous states.</p>
Ready	The Ready state is assigned to optional policies available when an application content pack is installed. These policies cannot be modified. If you want to modify a policy, create a copy and make the required changes.

### Policy version numbering

There are three identifiers for each version: Version name, Version, and Revision number. These identifiers are shown as three separate columns in the **Versions** tab.

Identifier	Description
Version name	<p>For Draft versions, the version name uses Draft-V{Revision number}.</p> <p>For Current (published) or Archived versions, V{Version} is used for the name.</p>
Version	<p>The Version number is assigned only when you publish a version. The number is automatically created according to the existing version number. For the first Current (published) version, the number is 1.0 and for subsequent current versions, the number increases by +1.0. For example, 2.0, 3.0, 4.0, and so on.</p> <p>Current versions that have been Archived retain the original version number.</p> <p>Draft versions are not assigned a version number.</p>
Revision number	<p>The Revision number is derived from the version that it was created from. If an empty new policy is created, the initial draft version is assigned 0.1. Each subsequent draft version is assigned 0.2, 0.3, 0.4, and so on.</p> <p>If the version was created from a published (Current or Archived) version, it inherits the Version number. For example, you can create a draft version from the Current (published) version 2.0. The draft version is assigned the revision number 2.1.</p>

The following image illustrates the different states and their version numbering.

As shown in the preceding image, the first draft policy version created was automatically assigned the Revision number 0.1. Each subsequent draft is assigned V0.2, V0.3, and so on. Draft version 0.2 (as indicated in the Revision number column) was published and became the Current version, with the Version name of V1.0.

A draft version of this Current version was created, with the Revision number 1.1. That draft was published to become the Current version and was assigned the Revision number 1.1. The previously Current version is now Archived, but retains the Revision number 0.2.

### Create new PaCE policy versions

You can create new versions for any of your existing policies.

After you have created a policy, a draft policy version is automatically created. This policy version contains the actual policy script, and may also contain mapping inputs and caller input definitions.

You can create a policy version from scratch, using low-code, or copy an existing policy version and use that as the basis for a new version. The version can include input definitions for mapping or caller variables. These parameters are used with the logic of the policy to determine whether the policy is in compliance.

For example, for a travel expense policy you can create several policy versions. Each version can include different mapping inputs, for example, limiting the different types of expenses. In one version, the breakfast expense limit could be \$25, and the dinner expense limit could be \$50. Each time the policy (and the relevant Current version) is invoked, the various expenses are validated by the policy to ensure they are in compliance.

### Create a PaCE policy version

Create a version from scratch at any time for any of your existing PaCE policies.

## Before you begin

Role required: sn\_pace.admin

## About this task

- Note:** Your administrator may have pre-configured policy versions that contain default caller and mapping inputs. You may not be able to edit these input variables, but you can add new inputs, as required.

## Procedure

1. Navigate to **All > Policies > My Policies**.

**Note:** To view all policies generated by admin/other users, navigate to **Policies > All Policies**.

2. Click the policy name for the policy that you want to add a new version to.
3. In the Policy Details page, click the **Versions** tab.
4. Click **New**.
5. Clear the **Is low code** check box.
6. Optional: In the Create New Policy Version form, add a description in the **Description** field.

**Note:** The **Description** field is the only editable field in the form. The version and revision numbering cannot be modified and are automatically assigned.

7. Click **Save**.

## What to do next

The policy version is saved and the version is assigned a number according to policy numbering. You can:

- Define API Variables, Config Parameters, and Record References.

See [Passing parameters to PaCE policies](#).

- Modify the policy script.

See [Edit a PaCE policy script](#).

- Test your policy version to ensure it is compliant.

See [Use the Test Playground for PaCE policies](#) for details.

- Publish the policy version.

See [Publish a PaCE policy version](#).

- Review execution activity for the version.

See [Review executed policy activity](#).

- Delete the version.

See [Delete a PaCE policy version](#).

## Create a PaCE policy version using low-code

Create a new policy version using low-code for any of your existing PaCE policies.

## Before you begin

Role required: sn\_dev\_ops\_config.admin

## About this task

The low-code function enables you to write policies using simple UI elements by defining conditions for the policy to determine if it is compliant or non-compliant. If the **Is low code** check box is selected when you create a new policy version, the **Policy Builder** and **Test Playground** tabs will automatically have the Policy logic page set to low-code.

You can switch to the Policy Script to create a policy using JavaScript by selecting the **Switch to code editor** button, but you will be unable to switch back to low-code. For more information on how to write PaCE policies using JavaScript, see [How to write and test PaCE policies](#).

**i Note:** Policy Builder works best in Google Chrome and Firefox.

## Procedure

1. Navigate to **Policies > My Policies**.
2. Click the policy name for the policy that you want to add a new version to.
3. In the Policy Details page, click the **Versions** tab.
4. Click **New**.
5. Fill in the **Description** field.

**i Note:** By default, the **Is low code** check box is automatically selected.

6. Click **Save**.

## Result

The policy version is saved with an assigned number according to the policy numbering and will enable you to set the conditions using low-code.

## Policy logic condition fields

Policy logic is a set of conditions that is used for determining whether a policy is compliant or non-compliant. You can set conditions using the condition fields.

## Policy logic page

You can add a condition set by clicking the **New condition set** button or delete a condition set by clicking the X icon .

### If and else if statement fields

Field	Description
Condition description	A description of the field.
Source	The variable you want to source for the condition.
Operator	A choice list of operators to filter the source for the condition. The list will change depending on the source selected.

## If and else if statement fields (continued)

Field	Description
Value	<p>A value to enter text or click the Data picker icon to concatenate multiple text strings with multiple data pills in order to select a variable for the log.</p> <p><b>Note:</b> If your Source is choice, the Data pill picker icon is inactive.</p>

You can add a dependent condition by clicking **or** or **and** next to the condition.

## Then and else statement fields

Field	Description
Decision	Decision to determine if the policy is <b>Compliant</b> or <b>Non-compliant</b> .
Log level	The level of the log.
Log message	Log message field to enter text or click the Data picker icon to concatenate multiple text strings with multiple data pills in order to select a variable for the log.
Output type	The output type of the log. You can select the plus icon to add multiple output types or the minus icon to delete the output type.
Data	Data field to enter text or click the Data picker icon to concatenate multiple text strings with multiple data pills in order to select a variable for the log.

## Create a copy of a PaCE policy version

You can create a copy of an existing policy version. All the settings and mappings from the existing version are copied over to the new one.

### Before you begin

Role required: sn\_pace.admin

### About this task

When you create a new policy version from an existing one, the input definitions and the policy script from the original version are copied to the new version. The version is assigned a number according to policy numbering.

**Note:** The new policy version is initially created as a draft, with a revision number based on the original policy version. For further details about version states and numbering, see [PaCE policy version states](#).

## Procedure

1. Navigate to **All > Policies > My Policies**.

**Note:** To view all policies generated by admin/other users, navigate to **Policies > All Policies**.

2. Click the name of the policy for which a version is to be created.

3. In the Policy Details page, click the **Versions** tab.

4. Click the **Version name** link of the policy version that you want to copy.

5. Click **Create New Draft**. A new copy of this policy version is created.

6. Enter a description for the new version and click **Save**.

## Duplicate PaCE policy versions

You can create a duplicate policy from a policy version or multiple versions from two or more policy versions. If you want to use a locked policy, you must first create a duplicate version and then modify the policy version as required.

## Duplicate a single policy version

To create a duplicate policy from a policy version, follow these steps:

1. Navigate to **All Policies > My Policies**.

**Note:** To view all policies generated by admin/other users, navigate to **Policies > All Policies**.

2. Select the name of the policy that you want to duplicate.

3. In the Policy Details page, click the **Versions** tab and select the version you want to duplicate.

4. Click the **More Actions** icon and select **Duplicate**.

5. Select an option in the **Duplicate version** form.

◦ To new policy:

    a. Select this option to create a duplicate policy from this version.

    b. Enter the policy name, category, and description.

    c. Click **Duplicate**.

The newly created policy is a duplicate copy and contains the same parameters and mappings of the selected policy version.

◦ To existing policy:

    a. Select this option to create a duplicate version for an existing policy.

    b. In the Duplicate Version form, select the policy to which the version is to be duplicated.

    c. Click **Duplicate**.

The newly created duplicate version is displayed in the Versions page of the selected policy.

## Duplicate multiple policy versions

1. Navigate to **All Policies > My Policies**.

**Note:** To view all policies generated by admin/other users, navigate to **Policies > All Policies**.

2. Select the name of the policy that you want to duplicate.
3. In the Policy Details page, click the **Versions** tab.
4. Select the policies that you want to duplicate and from the drop-down list, select **Duplicate versions**.
5. Select an option in the **Duplicate version** form and follow the remaining steps as listed in [Duplicate a single policy version](#).

## Passing parameters to PaCE policies

Parameters can be passed to a PaCE policy to validate updates to an object (tables and document IDs). These variables apply to authoring in both low-code or JavaScript. Policy versions include three types of parameter inputs: API Variables, Config Parameters, and Record References.

### API Variables

Previously known as Caller Inputs, the API Variables is passed to the PaCE API at the time of invocation by a developer. The API Variable is a variable that enables you to pass the value to the policy whenever the policy is invoked. Specify a value for this API Variable when calling the API, otherwise the policy is not executed and no decision is reached.

For each PaCE policy, there is only one pre-defined API Variable called **SnapshotId**. This API Variable is Immutable and cannot be modified or deleted. You cannot define any additional API Variables for a policy.

**Note:** If you are using code editor to create a policy, the API Variable will still be labeled as CallerInput.

### Config Parameters

Previously known as Mapped Inputs, the Config Parameters can be passed when mapping policies to an object (tables and document IDs). When you define a Config Parameter, you are creating a parameter that enables you to pass values to the policy whenever the policy is mapped. If you define mandatory inputs, you must specify values for these inputs when mapping the policy. If the inputs you define are not mandatory, the policy is not executed (the status is set to inactive) and no decision is reached.

**Note:** If you are using the code editor to create a policy, the Config Parameter will still be labeled as mappingInput.

For example, for a travel expenses policy you can add variables to define the limits of different types of expenses. The limits are specified when mapping the policy, and set the limits on the expense when the policy is invoked on this object. The breakfast expense limit for one group of employees could be \$25, and for a different group of employees the limit could be \$50. Each time the policy is invoked, the expenses are validated by the policy according to the limits specified in the mapping.

## Record References

Record references define queries to extract data from any ServiceNow® tables and use the data to configure the policy logic. This feature enables you to retrieve additional data that may be required while defining the policy. You can define a query to perform aggregate functions for a record reference.

### Create a new variable for a policy version

Create an API Variable, Config Parameter, or Record Reference for the policy version in the Define Variables section of the Policy Builder.

#### Before you begin

If you want to create a variable that will apply to all of your policies, then do the following:

1. Go to the Variable Definition section on the main page.
2. Select the variable section you want to create.
3. Select **Add**.

When you save the variable, it will show up in the Define Variables section on the Policy Builder of the policy.

Role required: sn\_pace.admin

#### Procedure

1. In the Details form of the relevant policy, click the **Versions** tab.
2. Click the Version Number of the relevant policy version.
3. Click the **Policy Builder** tab.
4. Select the variable tab that you want to create a variable from then click **New**.  
A pop-up window will appear titled with the variable you want to create.
5. On the form, fill in the fields as needed.

#### Create a new variable form

Field	Description
Label	Label name of the variable.
Name	Name of the variable.
Type	Drop-down list of input types: <ul style="list-style-type: none"> <li><input type="radio"/> Basic Date/Time</li> <li><input type="radio"/> Basic Time</li> <li><input type="radio"/> Choice</li> <li><input type="radio"/> Data Array</li> <li><input type="radio"/> Data Object</li> <li><input type="radio"/> Decimal</li> <li><input type="radio"/> Email</li> <li><input type="radio"/> Integer</li> <li><input type="radio"/> IP Address</li> <li><input type="radio"/> JSON</li> </ul>

Field	Description
	<ul style="list-style-type: none"> <li>◦ Other Date</li> <li>◦ Reference</li> <li>◦ String</li> <li>◦ Sys ID (GUID)</li> <li>◦ True/False</li> </ul> <p>Depending on the input you choose, the Advanced Options provides different fields for some inputs. For more information, see the <a href="#">Data Type descriptions</a> topic.</p> <p><b>i Note:</b> The list only shows the most common used data types.</p>
Mandatory	<p>Option to make this variable mandatory.</p> <p>When the variable is mandatory, a value must be provided when mapping the policy. Otherwise the policy cannot be executed.</p>
Description	Description of the variable.
Default Value	Default value for the variable.
Function (Optional)	<p>Functions to aggregate your condition. Choices are as follows:</p> <ul style="list-style-type: none"> <li>◦ Average: Average value of the condition.</li> <li>◦ Count: Count of the number of non-null values.</li> <li>◦ Maximum: Largest or maximum value.</li> <li>◦ Minimum: Minimum value.</li> <li>◦ Standard deviation: Population of standard deviation.</li> <li>◦ Sum: Sum of all values.</li> </ul> <p><b>i Note:</b> This field will only show up in the Record Reference variable form with the qualifier condition fields.</p>
Match criteria	<p>A match criteria for your record reference. Choices are as follows.</p> <ul style="list-style-type: none"> <li>◦ Select first match</li> <li>◦ Error on multiple</li> </ul> <p>When you create a Record Reference, it will query multiple answers. The match criteria enables you to filter out the match by selecting the first match or return an error if there are multiple answers.</p>

Field	Description
	<p><b>i Note:</b> This field will only show up in the Record Reference variable form with the qualifier condition fields. For more information about condition fields, see <a href="#">Policy logic condition fields</a>.</p>

## 6. Click **Save**.

### Result

You can add your variables into the Policy Builder as needed. Additionally, you can delete a variable by selecting it and clicking the **Delete** button.

### Data Type descriptions

The following is a list of the most common data types for creating a variable.

### Data Types

Data Type	Description
Basic Date/Time	Day and time of day, which can be selected with a calendar widget.
Basic Time	Specific time.
Choice	Script that returns an array of choices that will be displayed for selection in a drop-down menu. If you select the <b>Multi select</b> check box, then multiple options can be selected from the drop-down. If it is unselected, then only one option can be selected.
Data Array	Data Array structure represented as a String.
Data Object	The Data Object input enables you to select the plus icon to add a new property to the data object or the minus icon to delete it.  Additionally, you can select the Star icon that will search for a name if you don't know the structure of the JSON format property. A JSON path and match criteria field will appear and allow you to search for a key in the JSON structure.
Decimal	Number with up to two digits after the decimal points.
Email	Email represented as a String.
Integer	Number with zero decimal points.
IP Address	Variable character field that stores IPv4 and IPv6 addresses.

## Data Types (continued)

Data Type	Description
JSON	JavaScript Object Notation represented as a String.
Other Date	Date, which can be selected with a calendar widget.
Reference	Query that displays records from another table. The data type includes a match criteria field and qualifier conditions you can set.
String	For 255 characters or less, the string field is a single-line text field.
Sys ID (GUID)	Field that contains a unique identifier for each record.
True/False	Boolean field that appears as a one-digit integer.

### Edit a PaCE policy script

Edit the PaCE policy script to apply any changes to the policy logic.

#### Before you begin

Role required: sn\_pace.admin

#### About this task

**Important:** For further information about testing your policy script, see [Use the Test Playground for PaCE policies](#).

#### Procedure

1. Navigate to **All > Policies > My Policies**.

**Note:** To view all policies generated by admin/other users, navigate to **Policies > All Policies**.

2. Select the policy for which you want to update the script.
3. In the Details form of the relevant policy, click the **Versions** tab and select the version you want to edit.
4. In the policy version page, click the **Policy Builder** tab. The following image shows a sample policy script.

**Note:** The script is editable for policy versions in Draft state only. For policy versions in Current or Archived state, the script is read-only.

5. Apply any changes to the policy script, as required.
6. Click **Save**.

## Publish a PaCE policy version

You can publish a policy version at any time. When the policy is published, a compliant, non-compliant, or compliant-exception decision is made for objects (tables or document IDs) mapped to the policy.

### Before you begin

Role required: sn\_pace.admin

### Procedure

1. Navigate to **All > Policies > My Policies**.

**Note:** To view all policies generated by admin/other users, navigate to **Policies > All Policies**.

2. Click the policy name for the policy that includes the version you want to publish.
3. In the Policy Details page, click the **Versions** tab.
4. Click the **Version name** that you want to publish. Typically you would publish a Draft policy, but you can also republish an Archived version.
5. Click **Publish**. The following screen is displayed.
6. Select the **Activate this policy** check box to activate the policy when it is published.
7. In the displayed confirmation message, click **Publish**. The policy version becomes the Current version and is used the next time the policy is invoked.

**Note:** If the policy has an existing mapping to document IDs, when you publish a new version it verifies that all mandatory inputs for mapping inputs are provided. If you changed a mapping input, it must be updated. If you changed a caller input, you must update the PaCE API call to ensure that the policy is executed. If there is no mapping associated with the policy, or the policy is in an Inactive state, the mapping inputs are not verified.

## Edit a draft PaCE policy version

You can edit a Draft PaCE policy version at any time. The Draft version can be published to become the Current version. You can also create multiple different Draft versions from different Archived versions.

### Before you begin

Role required: sn\_pace.admin

### Procedure

1. Navigate to **All > Policies > My Policies**.

**Note:** To view all policies generated by admin/other users, navigate to **Policies > All Policies**.

2. Click the **Policy name** for the policy that includes the version you want to edit.
3. In the Policy Details page, click the **Versions** tab.
4. Click the **Version name** that you want to edit.
5. Edit the information and inputs in the **Details** and **Policy Builder** tabs.

Tab	Editable options
Details	<p>Update the <b>Description</b> field.</p> <p><b>i Note:</b> The <b>Description</b> field is the only editable field in the Details tab. The version and revision numbering are generated automatically and cannot be modified.</p>
Policy Builder	<p>In the Define Variables section, select Config Parameters. You can:</p> <ul style="list-style-type: none"> <li>◦ Click <b>New</b> to add a new input variable. See <a href="#">Create a new variable for a policy version</a> for details on adding input variables.</li> <li>◦ Select an existing variable and click <b>Delete</b> to remove the variable from the list.</li> </ul>

#### 6. Perform one of the following actions.

Option	Description
Save the version	Click <b>Save</b> . The policy version is saved and the version page is refreshed.
Publish the version	Click <b>Publish Policy</b> . In the displayed confirmation message, click <b>Publish</b> . The policy version becomes the Current version and is used the next time the policy is invoked.
Test your updates	Click the <b>Test Playground</b> tab. The Test Playground page is displayed, where you can update the policy script and test your changes. For further information, see <a href="#">Use the Test Playground for PaCE policies</a> .
Delete the version	Click <b>Delete</b> . Only versions that have never been used can be deleted.
Review execution activity	Click the <b>Executions</b> tab to review the execution activity for this version. See <a href="#">Review executed policy activity</a> for details.

#### Edit a current or archived PaCE policy version

To edit a Current (published) or Archived PaCE policy version, you must create a copy of that version, which can then be updated as required.

#### Before you begin

Role required: sn\_pace.admin

## Procedure

1. Navigate to **All > Policies > My Policies**.

**Note:** To view all policies generated by admin/other users, navigate to **Policies > All Policies**.

2. Click the **Policy name** for the policy that includes the version you want to edit.
3. In the Policy Details page, click the **Versions** tab.
4. Click the **Version name** that you want to edit.  
As you can't update a policy version that is Current (published) or Archived, the displayed fields are read-only.
5. Click **Create New Version**.

In the Version page, define the details of the new version. For further information, see the Create a new version of a policy section. In addition, you can create multiple Draft versions from an Archived or Current (published) version and work on them in parallel.

## Compare PaCE policy versions

You can compare two versions of a PaCE policy, whatever their state. Any differences between the two selected versions are highlighted for quick analysis.

### Before you begin

Role required: sn\_pace.admin

## Procedure

1. Navigate to **All > Policies > My Policies**.

**Note:** To view all policies generated by admin/other users, navigate to **Policies > All Policies**.

2. Click the **Policy name** for the policy that includes the versions you want to compare.
3. In the Policy Details page, click the **Versions** tab.
4. Select the relevant two versions to compare, and click **Compare**.  
The **Compare Versions** tab displays several container sections that highlight any differences between the two versions.
  - The **Details** container highlights differences between the details of the two versions. For example, differences between the state, version name and numbering, and the number of test or standard runs.
  - The **Script** container highlights any differences in the policy script of the two versions.
  - The **Mapping Inputs** container also lists any differences between the two versions.
5. To make any changes based on your analysis of the two compared versions, return to the **Versions** tab, select the relevant version, and modify and test as required.

## Delete a PaCE policy version

You can delete a policy version at any time. However, only versions that have never been published can be deleted. Published versions are stored for auditing purposes.

### Before you begin

Role required: sn\_pace.admin

## About this task

- Policy versions in Draft state can be deleted. When a version is deleted, all defined caller/mapping inputs and policy metadata associated with the version are also deleted.
- Policy versions that cannot be deleted include:
  - Policy versions in Current or Archived state.
  - Policy versions that have been executed in a Standard run.

## Procedure

1. Navigate to **All > Policies > My Policies**.

 **Note:** To view all policies generated by admin/other users, navigate to **Policies > All Policies**.

2. Click the **Policy name** for the policy that includes the version you want to delete.
3. In the Policy Details page, click the **Versions** tab.
4. Select the relevant version, click **Delete** and **OK** to confirm.

### Use the Test Playground for PaCE policies

The Test Playground enables you to test updates to a PaCE policy at any time, without the need to test or evaluate them in a Production environment.

You can update API Variables, Config Parameters, and Record References, and modify the policy script. After completing your changes, evaluate the updates in the Test Playground and review the execution output to ensure that the policy is compliant. Use the Test Playground to fine-tune your policy administration. Updates to a policy can be made on-the-fly, and tested in real time in a test environment. In addition, use the Test Playground to:

- Set execution parameters.
- Set and modify variables.
- Edit the policy script and policy logic.
- Use previously evaluated variables from a previous evaluation run.
- Review the execution output and the decision of compliance (compliant or non-compliant).
- Review a log of executions.

### Test updates to a PaCE policy version

Use the Test Playground to test updates to a PaCE policy script. You can evaluate changes to the policy script in real time, enabling you to determine if the changes that are applied render the policy compliant or non-compliant.

## Before you begin

Role required: sn\_pace.admin

## About this task

The Test Playground enables you to apply changes to the policy script, and see the impact of those changes in real time. After the changes have been evaluated, save the script to ensure that the changes are implemented when the policy is next invoked.

## Procedure

1. In the **Versions** tab, click the policy version you want to update.
2. In the Details page, click **Test Playground**.
3. In the Policy logic section, modify the fields according to your requirements.
4. In the Test Parameters tab, fill in the forms as needed.

### Test options fields

Field	Description
Change Request	The change request that you want to test the policy version with.
Log level	Level of the log.
Timeout	Timeout to determine the execution threshold of the test.
Execution log	Execution tag for the policy version.
Verbose	Option to make the test display in a verbose format.

5. Under the **API Variables and Config Parameters** tab, select the variables you want to test with the policy version.
6. When you are done modifying your policy script, click **Evaluate**.  
A decision is reached regarding the compliance or non-compliance of the policy, based on the changes you made to the script. In the Test Parameters section, either **Compliant** or **Non-compliant** is displayed, along with additional output on any warnings or failures, and how long the evaluation took.

### Review execution output and decision

After evaluating any changes made to the PaCE policy script, you can review the execution output and compliance decision reached. You can review this output immediately after evaluating any policy script changes.

## Before you begin

Role required: sn\_pace.admin

## Procedure

1. In the **Versions** tab, click the policy version you want to evaluate.
2. In the Details page, click **Test Playground**.
3. After modifying the policy script according to your requirements, click **Evaluate**.  
The **Output** tab is displayed automatically, and shows the output and the decision reached.
4. Review the execution output in the **Output** tab:

Output tab section	Description
Compliance decision	<p>The decision can be:</p> <ul style="list-style-type: none"> <li>◦ <b>Compliant:</b> The changes you applied to the policy script are compliant with the policy logic. When you click <b>Save Script</b>, the changes are saved, and the changes implemented the next time the policy is invoked.</li> <li>◦ <b>Non-compliant:</b> The changes you applied to the policy script are not compliant with the policy logic.</li> <li>◦ <b>Compliant-exception:</b> In this case, if a policy exception has been approved, any policies that are non-compliant are set to the compliant-exception state.</li> </ul>
Warnings/ Failures	Lists the number of warnings or failures generated during the execution.
Evaluation time	Shows how long the evaluation took, in milliseconds.
Execution output	Shows the execution output in JSON format. You can integrate this output with other applications and services and present it in your required format.

If required, modify the policy script to resolve any issues that arose during the evaluation. You can also review policy execution logs for further analysis.

### Review execution log

After evaluating any changes made to the PaCE policy script, you can review the execution log generated.

#### Before you begin

Role required: sn\_pace.admin

#### Procedure

1. In the **Versions** tab, click the policy version you want to review/evaluate.
2. In the Details page, click **Test Playground**.
3. Click the **Logs** tab to view the execution logs generated.

The displayed logs include all execution data generated during the evaluation. To browse the logs, use the arrow icons at the bottom of the tab.

#### Execute policies

Execute a policy to determine if the policy is compliant or non-compliant. Executions can be run as part of your policy evaluation process, or in real time when mapped to an object (table or document ID).

Before you publish a policy version, you can run a test evaluation of the version to determine if it is compliant, non-compliant, or compliant-exception. Evaluating a policy enables you to test repeatedly and apply any required changes to the policy version, before moving it to a production environment. For further information about testing your policy version, see [Use the Test Playground for PaCE policies](#).

When the version is ready to be moved into the production environment, make sure that all policy mappings are in place and verified. When the policy is invoked by the relevant object,

its execution output is stored for review. For further information, see [Review executed policy activity](#).

### Review executed policy activity

You can review the execution activity of any policy at any time. Recorded execution activity includes the time taken to execute, decisions of compliance, execution types, and the outcome of the execution.

#### Before you begin

Role required: sn\_pace.admin

#### Procedure

1. Navigate to **All > Policies > My Policies**.

**Note:** To view all policies generated by admin/other users, navigate to **Policies > All Policies**.

2. Click **Executions** to display the execution activity for the selected policy.

3. Review the execution information in the following columns.

Column	Description
Number	The automatically generated execution number. Click the link to review the following execution information: <ul style="list-style-type: none"> <li>◦ Caller and mapping inputs used in the execution.</li> <li>◦ The JSON output generated on execution.</li> <li>◦ The execution log.</li> <li>◦ The policy script used in the execution.</li> </ul>
Version	The policy version used in the execution. Click the link to review the version details.
Type	The type of execution performed: <ul style="list-style-type: none"> <li>◦ Test</li> <li>◦ Standard</li> </ul>
Start time	The time and date the execution was initiated.
Processing time (ms)	The execution processing time, in milliseconds.
Decision	The decision reached on execution: <ul style="list-style-type: none"> <li>◦ Compliant</li> <li>◦ Non-compliant</li> <li>◦ Compliant-exception</li> </ul>
State	The state of the execution.

Column	Description
CDM application	The name of the relevant application. Click the link to review and update the application details.
Name	The name of the object. Click the link to review and update the object details.

## Manage PaCE policy categories and tags

Use categories and tags to manage and group your PaCE policies more efficiently. As the policy administrator, you can create and edit any number of categories and tags.

You must have the sn\_pace.admin role to manage categories and tags.

### Policy categories and how to use them

Policy categories enable you to categorize your policies into relevant groups or sections.

You can create any number of categories, but only one category can be assigned at any one time to a policy. Categories can include product or industry-specific terms, such as PCI, HIPAA, or FedRAMP. If a PCI or HIPAA-specific update must be applied to a policy, select the relevant category to locate and update the policies under that category.

When creating or updating the policy with a category, the category must be pre-defined.

### Policy tags and how to use them

Policy tags enable you to define a policy with relevant reference tags. These tags can be defined according to your requirements, providing an element of flexibility to your policy management. For example, if a group of users have access to some policies, you can create a meaningful tag or a set of tags only for those policies.

You can add any number of tags to a policy, and also configure who can view the tags. You can also add tags on-the-fly as you view or edit policies.

Tags can be especially useful in collaboration efforts, in turn making the entire policy life-cycle easier to manage.

### Managing categories in PaCE

Use the Categories option to group your PaCE policies more efficiently. As the policy administrator, you can create and edit any number of categories.

Categories are especially useful for grouping your policies into relevant sections. You can create and edit categories as required, but only one category can be assigned at any one time to a policy.

In addition, when creating a policy, the category must be pre-defined to ensure it can be assigned to the policy.

### Create a category

Create PaCE policy categories to assign to a policy when creating or updating a policy. Categories enable you to categorize and manage policies more efficiently.

### Before you begin

Role required: sn\_pace.admin

## Procedure

1. Navigate to **All > Categories > All Categories**.
2. Click **New**.
3. In the Create New Category form, define a name and description for the category.
4. Click **Save**.  
The new category is added to the All Categories form.

## Edit categories

Update categories for a PaCE policy as required. You can also update categories for Active policies.

### Before you begin

Role required: sn\_pace.admin

## Procedure

1. Navigate to **All > Categories > All Categories**.
2. Click the name of the category that you want to edit.
3. In the Details form for the selected category, edit the category **Name** and **Description** fields, as required.  
When updating a category name, the update is applied to all policies assigned with that category, whatever their state.
4. Click **Save**.  
The edited category is updated in the All Categories form.

**Note:** You can also delete a category, as long as the category is not assigned to an Active policy. To delete a category, in the Details form for the selected category, click **Delete**.

## View PaCE policies associated with a category

You can view the PaCE policies that have been assigned to a specific category. All the categories created, together with their associated policies, can be viewed in the All Categories table.

### Before you begin

Role required: sn\_pace.admin

## Procedure

1. Navigate to **All > Categories > All Categories**.
2. In the All Categories table, click the relevant category name.
3. In the displayed Details form, click the **Policies** tab.  
The policies associated with the selected category are displayed.
4. Click a policy name to view the policy and its settings.

## Managing tags in PaCE

Use tags to manage your PaCE policies more efficiently. As the policy administrator, you can create and edit any number of tags.

Policy tags enable you to add any number of reference tags to a policy. Tags can be especially useful in collaboration efforts, and can easily be viewed and edited. They can also be used to group policies based on any number of tags.

### Add tags to PaCE policies

Add or create policy tags on-the-fly for any PaCE policy, including currently Active policies. Tags enable you to manage policies more efficiently.

#### Before you begin

Role required: sn\_pace.admin

#### Procedure

1. Navigate to **All > Policies > My Policies**.

**Note:** To view all policies generated by admin/other users, navigate to **Policies > All Policies**.

2. Click the policy name for the policy that you want to add a tag to.

3. In the Policy Details page, click the tag icon.

**Note:** Unlike categories, you do not have to pre-define a tag before adding it to a policy.

4. In the **Add Tag** field, enter the required tag.

5. Press **Enter** to add the tag.

The tag is added to the **Tags** dialog.

**Note:** By default, the added tags are viewable to you only. Edit the tag to modify who can view the tag.

6. Repeat as required for additional tags.

#### Note:

- Tags are not case-sensitive, you cannot create tags using different capitalization.
- Tags are automatically assigned to the policy when added in the **Tags** dialog, you don't need to save the policy to save the assigned tags.

### Edit PaCE policy tags

Edit and remove tags from a PaCE policy as required. When editing a tag, you can also define who can view the tag.

#### Before you begin

Role required: sn\_pace.admin

#### Procedure

1. Navigate to **All > Policies > My Policies**.

**Note:** To view all policies generated by admin/other users, navigate to **Policies > All Policies**.

2. Click the policy name for the policy that you want to edit or remove tags from.

3. In the Policy Details page, click the tag icon.
4. To edit a tag:
  - a. Click the tag.
  - b. In the Edit Tag dialog, edit the tag name.
  - c. In the Viewable by list, select from **Me**, **Groups and Users**, or **Everyone**. By default, **Me** is selected when initially creating a tag.

**Note:** When you select **Groups and Users**, you are prompted to choose the Now Platform groups and users who can view the tag.

- d. Click **Save**.
5. To remove a tag, click the x on the tag itself.

The tag is removed from the policy.

### View PaCE policies associated with a tag

You can view the PaCE policies that have been assigned to a specific tag. All tags together with their associated policies can be viewed in the All Tags table.

#### Before you begin

Role required: sn\_pace.admin

#### Procedure

1. Navigate to **All > Tags > All Tags**.
2. In the All Tags table, click the name of the relevant tag.
3. In the displayed Details form, click the **Policies** tab.  
The policies associated with the selected tag are displayed.
4. Click a policy name to view the policy and its settings.

#### How to write and test PaCE policies

This section provides guidelines on how to write and test PaCE policies.

You can write PaCE policies using JavaScript and create the policy record within your instance application scope. PaCE policies can be executed within the scope in which they are created and all functions available in this scope are available to the policy.

When a policy is invoked, a default set of parameters is passed and these parameters can be used in the policy logic as required. A decision indicating whether the policy is compliant, non-compliant, or compliant-exception and additional information is returned using the decision object.

#### Structure of a PaCE policy script

This section describes the structure of a PaCE policy script.

When a policy is executed, a set of parameters is passed and the policy developer can use these parameters in this policy script to make a decision whether it is compliant, non-compliant, or compliant-exception and return this decision back to the calling service. The following image shows a sample script:

#### Sample policy script

The following table lists the policy function parameters and how can they be used in your policy script.

Variable Name	Description
logger	<p>Logger is an object that the policy coder can use to log messages. The log messages are stored in the <code>sn_pace_execution_log</code> table. You can review these log messages for debugging, tracking, or monitoring purposes. The log messages can be logged with one of the following levels:</p> <ul style="list-style-type: none"> <li>• 1: info</li> <li>• 2: debug</li> <li>• 3: warning</li> <li>• 4: error</li> </ul> <p>When a policy is invoked through the API, you can specify the desired log level. For example: The format is as follows:</p> <pre>logger.info("** snapshotId is: "+snapshotId);</pre>
current Record	<p><code>currentRecord</code> is an object of the current version of the policy that is executed when the policy is used. To view the <code>currentRecord</code> details, navigate to the <b>Versions</b> tab on the Policy Home page and locate the policy whose version state is set to <b>Current</b>.</p> <p>In the image below, you can see that the current active version of the policy highlighted. It shows the following details:</p> <ul style="list-style-type: none"> <li>• version number</li> <li>• last update date</li> <li>• the name of the user who updated the version</li> <li>• the number of times it was executed by the calling application</li> <li>• the number of times it was run in the test environment</li> </ul> <p>The policy script can access this data during policy execution time by interacting with the <code>currentRecord</code> object passed to the policy.</p> <p>This example shows how a policy developer can access the properties of the policy version record:</p> <pre>(function(logger, currentRecord, documentRecord, callerInput, mappedInput, childrenOutputs, output) {     //retrieve the input values     logger.info(currentRecord.getValue('name')); } )(logger, currentRecord, documentRecord, callerInput, mappedInput, childrenOutputs, output);</pre>
document Record	<p>The <code>documentRecord</code> is used to map the policy to the relevant object (table and documentID) that is being validated. The <code>documentRecord</code> is a combination of the <code>table_name</code> and <code>sysID</code>. Based on the properties of the object, the policy logic is used to manage and interact with the object being validated to determine the right decision.</p>

Variable Name	Description
	<p>For example: In the DevOps Config environment, a policy can be mapped to the deployable. When the API is invoked, a query is initiated on the document (deployable) table and the Sys ID of the deployable.</p> <pre data-bbox="362 297 1367 438">{     "table": "sn_cdm_deployable",     "sysId": "d1be8f5e87d80110eec7dbdd3fbb357d" }</pre>
	<p>This example shows how the documentRecord can be used in a policy script:</p> <pre data-bbox="362 508 1367 908">(function(logger, currentRecord, documentRecord, callerInput, mappedInput, childrenOutputs, output) {     //assuming that associated document has state field     var documentState = documentRecord.getValue("state");     if (documentState == "new")         ...     else         ... } (logger, currentRecord, documentRecord, callerInput, mappedInput, childrenOutputs, output);</pre>
	<p><b>callerInput</b> The callerInput is passed to the PaCE API when it is invoked. It includes all the caller input variables defined in the policy version. See &lt;Defining Caller Inputs&gt; section for details</p> <p>The <a href="#">Sample policy script</a> shows how policies can be used to validate configuration data in the DevOps environment. In the sample script, the callerInput variable is defined as</p> <pre data-bbox="362 1219 1367 1256">var snapshotId = callerInput.snapshotId;</pre> <p>where the specified <code>snapshotId</code> is mapped to the corresponding <code>snapshotId</code> of the DevOps Config deployable being validated based on specified criteria.</p> <p>The policy developer can define logic in the policy script to use the callerInput values passed when the API is invoked to determine the decision. For example, the SnapshotID passed is used to identify key-values related to the specific snapshotID for a deployable passed in the documentRecord object.</p>
<b>mapped Input</b>	<p>Mapping inputs are variables are passed when mapping the policy and includes all the mapping input variables defined for a specific version of the policy.</p> <p>The <code>mappedInput</code> variable is defined as follows in the <a href="#">Sample policy script</a>.</p> <pre data-bbox="362 1731 1367 1769">var dbPort = mappedInput.dbPort;</pre> <p>The policy developer can define the logic in the policy script to use the values passed at mapping to determine the decision. For example, the <code>dbPort</code> number must be less than 30000, otherwise the policy is considered to be non_compliant.</p>
children	Not supported in this version.

Variable Name	Description
Outputs	<p>output</p> <p>This parameter is used to pass the output of the policy execution that includes the decision back to the calling service. It provides a decision related to this policy, with additional information such as errors, warnings, and result details.</p>
	<p>This example shows a sample output with compliant and non_compliant decisions:</p> <pre>{   "decision": "compliant",   "results": [],   "warnings": [],   "failures": [],   "state": "complete" }  {   "decision": "non_compliant",   "results": [],   "warnings": [],   "failures": ["Failed to validate key"],   "state": "complete" }</pre>
output.	<p>The decision property can be set to:</p> <ul style="list-style-type: none"> <li>• Compliant: Determines that the policy complies with requirements.</li> <li>• Non-compliant: Determines that the policy does not comply with the requirements.</li> <li>• Compliant-exception: Determines that an exception to the policy has been approved and any policies that are non-compliant are set to the compliant-exception state.</li> </ul> <p>The decision is returned in a JSON format back to the calling service.</p> <p><b>i Note:</b> If a value is not specified in the <code>output.decision</code> field in the script, when the policy is executed, by default this field is set to <b>compliant</b> if there are no failures.</p>
output.	<p>results</p> <p>The results property can be used to pass data back to the calling service on decisions made during the policy validation phase. A list object with a list of results can be passed back to the calling service.</p>
output.	<p>warnings</p> <p>The warnings property can be used to pass data back to the calling service on any warnings that occurred during the policy validation phase.</p>

Variable Name	Description
output.failures	The <code>failures</code> property can be used to pass data back to the calling service on any errors that occurred during the policy validation phase.
<b>i Note:</b> The following fields are automatically filled in when the PaCE policy script is executed.	
output.name	The name (current version) of the policy being executed.
output.state	It indicates the state of the policy invocation. <ul style="list-style-type: none"> <li>• Complete: The invocation was completed successfully.</li> <li>• Pending: Indicates that the policy could not be successfully executed and is an incomplete state.</li> </ul>

### **i Note:**

- Auto-complete suggestions are available in the policy script editor for the `logger`, `callerInput` and `mappedInput` parameter.
- To view additional information for a parameter, enter the parameter name and select one of the options as follows:

### Policy invocation API

When a PaCE policy is invoked, the calling service passes the relevant object (tables and document IDs) that is being validated by providing the `documentRecord` object. PaCE determines which policies must be executed by using the mapping table and the `documentRecord`.

All selected policies are executed concurrently. When all the policies have been executed, a final single decision is returned (in JSON format) with information on all executed policies and policy-level decisions. The final decision can be:

- compliant: Indicates that all the selected policies comply with the requirements and have returned a compliant decision.
- non\_compliant: Indicates that one or more of the selected policies do not comply with the requirements and have returned a non\_compliant decision.
- compliant-exception: Indicates that a policy exception has been approved and any policies that are non-compliant are set to the compliant-exception state.

Policy invocations can be in one of the following states:

- new
- in\_progress
- complete
- canceled
- error
- timeout

## Sample API

The following is a sample invocation request when the PaCE API is executed.

```

var request = {"documentIds": 

[{"table":"sn_cdm_deployable","sysId":"d1be8f5e87d80110eec7dbdd3fbb357d"}, {"service":"CDM", "category":"CDM", "data":{"snapshotId":"2bcec7de87d80110eec7dbdd3fbb3529"}, "options":{"verboseResponse":true,"logLevel":"debug", "type":"standard", "failFast":false, "executionTag":"cdm"}}

try {

    var api = new sn_pace.InvocationAPI();

    var execResponse = api.execute(request);

    gs.info(" Got execution response:\n " + 
JSON.stringify(execResponse,null,4));

} catch (err) {

    //var errJSON = JSON.parse(err);

    //gs.info(errJSON.code);

    gs.info(" Error message is:" + err + "\n stack:" + 
err.stack);

    // gs.info(" Error is:" + err);

}

}

```

## PaCE API invocation parameters

This table describes the properties that can be configured when the PaCE API is invoked. These properties can be set whenever the API is invoked or when you test a policy in the Test Playgroun

Property name	Required	Default value	Description
service	yes		The calling service that is invoking the API.
category	no		The category type.
type	no	standard	Can be standard or test.
documentIds	yes (array of table, sysID)		The documentIds is an array of the table and sysID (of the deployable).

**i Note:** The properties shown in the preceding table cannot be configured in the Test Playground. They can only be used when the PaCE API is invoked.

data	yes		The policy-specific data that includes all possible caller inputs.
executionTag	no		Can be used to tag one or more policy executions records resulting from this invocation. It can be used to retrieve execution details of the policy based on the executionTag.
verboseResponse	no	false	<p>Default value is <b>false</b>. If set to <b>true</b>, additional debug information (in this case the ids of the mapped policies) is returned.</p> <p>A sample invocation response with options.verboseResponse:true is shown here:</p> <pre>// Sample execution response {   "rootExecutionId": "50202b3ec37b101017e06c576e40dd81",   "mappedPolicies": [     {       "policyId": "3d1de0a8c363101017e06c576e40dde4",       "policyName": "Production Database Policy",       "shortDescription": "Policy to check database passwords",       "versionId": "4ed08520c3a3101017e06c576e40ddcb",       "versionType": "4f2cb75473331010ce6c39282bf6a753",       "versionState": "published",     }   ] }</pre>

Property name	Required	Default value	Description
			<pre>     "versionApproval": "approved"     },     {         "policyId": "add2ca8c363101017e06c576e40dd12",         "policyName": "API Policy",         "shortDescription": "Policy to check API policies",         "versionId": "a6418920c3a3101017e06c576e40dd57",         "versionType": "4f2cb75473331010ce6c39282bf6a753",         "versionState": "published",         "versionApproval": "approved"     } ], "logs": [] } </pre> <p>If <code>options.verboseResponse:false</code>, the mapped policies are not included in the response.</p>
failFast	no	false	The default value is <b>false</b> . If this property is set to <b>true</b> , the first failed policy stops the execution of all the mapped policies.
logLevel	no	info	<p>This parameter can be used to specify how the log messages should be displayed.</p> <ul style="list-style-type: none"> <li>• info</li> <li>• debug</li> <li>• warn</li> <li>• error</li> </ul>
timeout	no	3600	The timeout period in seconds. The default value is 3600.
waitForResult	no	3600	<p>The waitForResult period in seconds.</p> <p><b>i Note:</b> This parameter is available only in the Test Playground.</p>

## About policy exceptions

By integrating Policy as Code Engine with Integrated Risk Management and Policy and Compliance Management workspace, the Compliance administrator can manage an organization's overall compliance posture and provide a business and regulatory context for codified policies.

The Compliance administrator can associate PaCE policies with control objectives and map policies to DevOps Config deployables. It enables preventive controls and ensures that no risky or non-compliant changes are introduced in the production environment. If required, you can also grant and monitor exceptions on these policies. This provides real-time risk visibility of control objectives and exception management to accelerate the DevOps pipeline process. Refer to the Governance, Risk, and Compliance documentation for more information on the exception management process.

### Enable exceptions for a policy

Enable exceptions for a PaCE policy and associate it with a control objective in GRC.

#### Before you begin

- Role required: sn\_pace.admin
- Governance, Risk, and Compliance 14.1 or later

#### Procedure

1. Navigate to **All > Policies > My Policies**.

 **Note:** To view all policies generated by admin/other users, navigate to **Policies > All Policies**.

2. Click the policy for which you want to enable exceptions. The Details page is displayed.

As you can see in this image, the Exception status is **Disabled**.

3. Click the **More Actions** icon and select **Enable exceptions**. The Exception status is changed to **Enabled**.

 **Note:** After an exception has been enabled, the policy must be configured in the Policy and Compliance Management workspace and connected to a control objective. See [Policy as Code Engine for Preventive compliance management](#) for details.

4. After the policy has been configured in Policy and Compliance Management workspace, the Exception status for the policy is updated to **Enabled and Configured**. The sn\_pace.admin can then request exceptions to this policy.

### Disable exceptions for a policy

You can disable exceptions for a policy for which exceptions have been granted. When this option is selected, you can no longer request exceptions and control objectives can no longer be associated with the policy.

#### Before you begin

- Role required: sn\_pace.admin

## Procedure

1. Navigate to **All > Policies > My Policies**.

**i Note:** To view all policies generated by admin/other users, navigate to **Policies > All Policies**.

2. Click the policy for which an exception has already been granted. The Details page is displayed.
3. Click the **More Actions** icon and select **Disable exceptions**. The Exception status is changed to **Disabled**.  
You can no longer request exceptions for this policy.

## Request exception for a policy

After a policy has been enabled and configured for exceptions, you can map a policy to a deployable and request an exception.

### Before you begin

Role required: sn\_pace.admin

## Procedure

1. Navigate to the Mappings page.

2. Select a mapping from the list and click **Request exception..**

3. In the Request policy exception window, specify the following:

- Reason for exception: Select a reason for the exception from the drop-down list. Refer to the Governance, Risk, and Compliance for details.
- Business justification: Enter a business justification for the exception.
- Start and End Date: Specify the period for which the exception should be valid. The default validity period is 30 days. The default period can be modified by the Compliance administrator (sn\_compliance.admin).

**i Note:**

- You can also specify a Start Date that is in the future.
- When the request is approved, the sn\_compliance.admin can override the requested dates.

4. Click **Request** and navigate to the **Exceptions** tab to view the requested exceptions.

You can see that the exception status is **Pending**. The Compliance Manager assesses this exception request and can choose to approve or reject the request. See [Policy as Code Engine for Preventive compliance management](#) for details. For each exception request, the following details are displayed:

- Exception ID: Click the Exception ID to view detailed information about the exception request.
- Deployable: Click the link to view the details of the deployable mapped to the policy.
- Exception active: If the exception has been granted, the status is set to **Yes**.

**i Note:** If the exception has been granted for a future date, the status is set to **No** until the effective Start Date.

- Status: The status of the request:
  - Pending: An exception request has been created but has not been approved.
  - Granted: The exception request has been approved.
  - Extension Pending: An extension request for a granted exception has been created.
  - Extension Granted: The extension request has been approved.
  - Declined: The exception request was not approved.
  - Extension Declined: The request to extend a granted exception was declined.
  - Expired: The period for which the exception was granted has expired.
  - Withdrawn: The granted exception request is withdrawn.
- Expires in: The number of days after which the exception request expires.

5. After the request is approved, navigate to the **Exceptions** tab. You can see that the exception request status has changed to **Granted**. You can also see the number of days and minutes for which the exception is valid.
6. After an exception has been granted, you can request an extension before the end of the validity period. Select a request with a **Granted** status and click **Extend**.

**i Note:** You can request an extension for only one policy at a time.

7. In the Request extension window, specify a business justification, an end date for the request and click **Request**.
8. Navigate to the **Exceptions** tab. The exception status has changed to **Extension Pending**. The sn\_compliance.admin assesses this extension request and can choose to approve or reject the request. See Governance, Risk, and Compliance for details.
9. After the request is approved, navigate to the **Exceptions** tab. You can see that the exception request status has changed to **Extension Granted**. You can also see the number of days for which the extension is valid.

**i Note:** When you execute a policy with exceptions, it is validated and returns a **Compliant-exception** result.

### Withdraw an exception request

If an exception is no longer required, you can withdraw a granted exception request for a policy.

#### Before you begin

Role required: sn\_pace.admin

#### Procedure

1. Navigate to **All > Policies > My Policies**.
2. Navigate to the policy that has been enabled and configured for exceptions and click the **Exceptions** tab.
3. Select the granted exception request and click **Withdraw**.
4. A confirmation message appears. Click **Withdraw** to withdraw the exception request. The request status is changed to **Withdrawn**.

# Index

S

Shared library  
Component libraries  
Shared components  
Configuration Data  
Management  
CDM  
393