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.

If you have comments about this documentation, submit your feedback to:
docfeedback@servicenow.com
# Table of Contents

<table>
<thead>
<tr>
<th>Section</th>
<th>Page</th>
</tr>
</thead>
<tbody>
<tr>
<td>DevOps</td>
<td>4</td>
</tr>
<tr>
<td>DevOps overview</td>
<td>9</td>
</tr>
<tr>
<td>Install DevOps</td>
<td>11</td>
</tr>
<tr>
<td>Setting up DevOps tools</td>
<td>15</td>
</tr>
<tr>
<td>Accelerating DevOps change</td>
<td>17</td>
</tr>
<tr>
<td>DevOps dashboard and Pipeline UI</td>
<td>100</td>
</tr>
<tr>
<td>User-created DevOps integrations</td>
<td>125</td>
</tr>
<tr>
<td>DevOps reference</td>
<td>148</td>
</tr>
<tr>
<td>DevOps reference</td>
<td>194</td>
</tr>
</tbody>
</table>
DevOps

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

ServiceNow DevOps approach

Connect to CI/CD toolchain

Automate Change Management

DevOps Change Request
- Risk assessed
- Impact understood
- Build successful
- Unit tests passed
- Test coverage >80%
- Security tests passed
- Auto Approve

Share Insights

Days/weeks to minutes with automated change approval, chain of custody audit transparency

© 2020 ServiceNow, Inc. All rights reserved. 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 names, product names, and logos may be trademarks of the respective companies with which they are associated.
Achieve enterprise DevOps

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

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

- Analytics, Intelligence, and Reporting Performance Analytics
- IT Service Management Change Management
- IT Business Management Agile Development
- Governance, Risk, and Compliance Risk Management

View and download the full infocard for a highlight of the DevOps features.

<table>
<thead>
<tr>
<th>Feature</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>Automate deployments by accelerating change</td>
<td>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.</td>
</tr>
<tr>
<td>Evaluate and monitor DevOps process and results</td>
<td>Use the data-driven DevOps Insights dashboard to view change results reports, pipeline value stream, and evaluate the overall DevOps process.</td>
</tr>
<tr>
<td>Visualize interactions and results across a pipeline execution</td>
<td>Use the DevOps Pipeline UI view to show pipeline stage progression and details for each app.</td>
</tr>
<tr>
<td>Set up user-created integrations for additional planning, coding, orchestration, and test tools</td>
<td>Integrate planning, coding, orchestration, and test tools not included in the integrations provided with the DevOps application.</td>
</tr>
</tbody>
</table>
Automate deployments by accelerating change

Change requests are automatically created for stages under change control. You can also enable automatic approval of DevOps changes using workflows, such as Change Approval Policy and DevOps Change Policy.
Evaluate and monitor DevOps process and results
Use the DevOps Insights dashboards to analyze operational and business reports and determine the overall efficiency and growth associated with your development processes.

**Visualize interactions and results across a pipeline execution**

Quickly view how everything is connected to see exactly what's happening with the pipeline and when. View the commits, the committers, and other details for a change request in one place.
Set up user-created integrations for additional planning, coding, and orchestration tools

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

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

DevOps

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

The DevOps application collects data across the entire set of lifecycle activities to provide visibility for DevOps teams so they can own the end-to-end process (plan, develop, build, test, deploy, and operate). Visibility across the entire DevOps toolchain, versus a single tool, exposes overall performance, including areas in the process that may need improvement.

As a result of this visibility, enterprise governance can be accelerated using DevOps, including automating change decisions.
To learn more about how your ServiceNow instance supports DevOps, see *DevOps overview* for a general understanding of the application and how it is used.

**Start gaining insight using DevOps**

Connect your tools and configure your pipeline to automate lifecycle activities. View pipeline stage progression and details for each app using the DevOps Pipeline UI. Monitor pipeline modeling and change acceleration benefits in your environment using the DevOps Insights dashboard.

- Connect your planning, coding, and orchestration tools, and model your pipeline
  Tool support includes Azure DevOps Boards, Jira, and ServiceNow Agile Development 2.0 planning tools, Azure DevOps Repos, GitHub, Bitbucket Server (version 5.10.0), and GitLab SCM coding tools, and Azure DevOps Pipelines, Jenkins, and GitLab CI/CD orchestration tools.
- Automate deployments by accelerating change
  Automatically create change requests at any stage for deployments that require change control in your environment, and use change approval policies to automate change request approval to continue deployment through the execution pipeline automatically.
- Evaluate and monitor DevOps process and results
  Use the data-driven DevOps Insights dashboard to view change results reports, pipeline value stream, and evaluate the overall DevOps process.
- Visualize interactions and results across a pipeline execution
  Use the DevOps Pipeline UI view to show pipeline stage progression and details for each app.
- Set up user-created integrations for additional planning and coding tools
  Integrate planning, coding, and orchestration tools not included in the integrations provided with the DevOps application.
Understand DevOps concepts

- DevOps overview and getting started
- Install DevOps
- Setting up DevOps tools
- Accelerating DevOps change
- DevOps dashboard and Pipeline UI
- User-created DevOps integrations
- DevOps API

Find DevOps resources

- DevOps release notes
- Ask or answer questions in the Now Community

Get help from DevOps resources

- Search the HI Knowledge Base for known error articles
- Contact ServiceNow Technical Support
- ServiceNow upgrades

DevOps overview

Set up DevOps roles, connections, integrate with external tools, then use the Insights dashboard to analyze operational and business reports and gain insight into your DevOps environment.

Use the change acceleration feature of DevOps to automatically create a change request for a stage in your development pipeline to accelerate change.

DevOps integrations

<table>
<thead>
<tr>
<th>Tool Type</th>
<th>Tool</th>
</tr>
</thead>
<tbody>
<tr>
<td>Planning</td>
<td>• Azure DevOps Boards</td>
</tr>
<tr>
<td></td>
<td>• Jira (ver 7.x and 8.x)</td>
</tr>
<tr>
<td></td>
<td>• ServiceNow Agile Development (ver 2.x)</td>
</tr>
<tr>
<td>Coding</td>
<td>• Azure DevOps Repos</td>
</tr>
<tr>
<td></td>
<td>• Bitbucket Server/Enterprise (ver 5.10.0)</td>
</tr>
<tr>
<td></td>
<td>• GitHub and GitHub Enterprise (ver 2.x)</td>
</tr>
<tr>
<td></td>
<td>• Basic authentication</td>
</tr>
<tr>
<td></td>
<td>• OAuth</td>
</tr>
<tr>
<td></td>
<td>• GitLab SCM (ver 12.8.0)</td>
</tr>
</tbody>
</table>
### DevOps applications

The **DevOps application** uses these related DevOps applications:

- **DevOps Insights application**
  Use the DevOps Insights dashboard to manage and track results.
- **DevOps Integrations application**
  Use DevOps Integrations application for *Azure DevOps integration with DevOps* and *Jenkins integration with DevOps* (starting with version 1.12), and *GitLab integration with DevOps* (starting with version 1.16).
**DevOps extensions**

Use these extensions to model your pipeline in DevOps:

**ServiceNow DevOps extension for Azure DevOps**

Starting with version 1.12, the ServiceNow DevOps extension on Visual Studio Marketplace is required to integrate your Azure DevOps pipeline with ServiceNow DevOps.

**Jenkins plugin for ServiceNow DevOps**

A Jenkins plugin 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.

**DevOps workflow**

DevOps integrates with external planning, coding, and orchestration tools and automatically creates change requests at any stage for deployments that require change control.

Change approval policies can be used to automate change request approval to continue deployment through the execution pipeline automatically.

Performance and efficiency in your DevOps environment are monitored and analyzed using the DevOps Insights dashboard. Pipeline executions are visualized on the DevOps Pipeline UI view.

**DevOps Concepts**

These concepts are useful to understand with respect to the DevOps application.

**Pipeline**

A pipeline is a set of steps that begins with planning (plans for the work to be done). An app is the item being worked on, and the work is done via commits to a code repository. Once committed, an orchestration tool picks up the change and sends it through a series of pipeline steps, including production.

Pipeline steps can include quality checks, like functional, security, load, and behavioral tests as well as deployments, and infrastructure provisioning. The result is fully tested development features delivered as quickly as possible to production.

**Integrations**

The DevOps application integrates with external tools by exposing REST endpoints to receive webhook notifications, or direct REST calls from tools.

**API**

The DevOps application includes a DevOps API that allows integration with any coding, planning, or orchestration tools.

**Getting started with DevOps**

Get started using the DevOps application to integrate with your existing DevOps toolchain, manage processes, accelerate changes, and view operational and business insights.

Integrate with external planning, coding, orchestration, and testing tools. Out-of-the-box integrations collect lifecycle events and data from your existing DevOps toolchain.
Tool support includes Azure DevOps Boards, Jira, and ServiceNow Agile Development 2.0 planning tools, Azure DevOps Repos, GitHub, Bitbucket Server (version 5.10.0) and GitLab SCM coding tools, and Azure DevOps Pipelines, Jenkins, and GitLab CI/CD orchestration tools.

**DevOps requirements**

- ServiceNow IntegrationHub Runtime (com.glide.hub.integration.runtime) plugin
- ServiceNow IntegrationHub Action Step - REST (com.glide.hub.action_step.rest) plugin
- MID Server

**Note:** You must set up a MID Server for on-premise tools (for example, GitHub, Jira, Jenkins).

**DevOps users**

DevOps integration and system users are automatically created in the DevOps application and are used to integrate with external tools.

- DevOps Integration User [devops.integration.user]

**Note:** You must configure the password before the DevOps Integration User can configure a tool.

- DevOps System [devops.system] user

**DevOps roles**

Before you can use the DevOps application, an admin user must assign DevOps roles.

- DevOps Administrator [sn_devops.admin] role is required for DevOps administration and setting up tools.
- DevOps integration [sn_devops.integration] role is required for DevOps inbound access to the integration tools in your environment.
- DevOps manager [sn_devops.manager] role is required for DevOps operation management, including monitoring performance in your DevOps environment.
- DevOps viewer [sn_devops.viewer] role is required to access the DevOps navigation menu and view DevOps information.

**Initial DevOps Setup**

Once the DevOps application is installed and roles are assigned, complete the setup configuration.

System admin:

- Configure an HTTP connection in CreateDevOpsTool (DevOps connection and credential alias provided) to automatically connect to your tools
- Add admin credentials to CreateDevOpsTool to allow access to your tools
DevOps admin:
- Create a DevOps tool record (required to connect to and discover tool configuration)
  - Application plans
  - Repositories
  - Orchestration tasks
  - Pipelines
- Model your pipeline in DevOps (associating pipeline steps is required to receive notifications from your tool)
- Set up change acceleration
- Integrate test tools, if applicable

View the DevOps Insights dashboard
Use the *DevOps Insights dashboard* to manage and track results.

View the DevOps Pipeline UI
Use the *DevOps Pipeline UI* for a graphical view of pipeline executions.

Install DevOps

Install the DevOps 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.

Role required: admin

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 application and click Install.
4. Optional: For Azure DevOps and Jenkins tool support (starting with version 1.12), and for GitLab tool support (starting with version 1.16), Install DevOps Integrations.

Install DevOps Integrations

For Azure DevOps and Jenkins tool support (starting with version 1.12), and for GitLab tool support (starting with version 1.16), install the DevOps Integrations 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.

Role required: admin

The DevOps Integrations application is required to integrate with these tools:
- Jenkins (starting with version 1.12)
- Azure DevOps (starting with version 1.12)
  - Azure Boards
  - Azure Repos
- Azure Pipelines
- GitLab (starting with version 1.16)
  - GitLab SCM
  - GitLab CI/CD (basic CI pipelines)

**Note:** You must *Install DevOps* before installing DevOps Integrations.

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 Integrations application and click *Install*.

### Install DevOps Insights

Install the DevOps Insights Standard dashboard application from ServiceNow Store applications. Visit the [ServiceNow Store](https://store.service-now.com) 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](https://store.service-now.com).

Role required: admin

The DevOps Insights application uses Performance Analytics to track performance metrics over time.

**Note:** Starting with version 1.12, the DevOps Insights Standard dashboard is the sole dashboard for the DevOps application. You must install the DevOps Insights application to view the Insights module.

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 Insights application and click *Install*.
4. 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 [DevOps] Historical Data Collection job to collect historical Insights data.

**Note:** This job 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.

a) Navigate to *Performance Analytics* > *Data Collector* > *Jobs* and open the [DevOps] Historical Data Collection record.
b) Select *Active* and set the *Run As* credentials to *System Administrator*.
c) Click *Execute Now*.

5. Starting with version 1.14, unless the [DevOps] Daily Data Collection job was previously customized, set it up to collect daily Insights data.

**Note:** This is a scheduled job (to be run on a regular basis) to collect daily DevOps data. For optimal performance, set this job to run during periods of low usage.

a) Navigate to *Performance Analytics* > *Data Collector* > *Jobs* and open the [DevOps] Daily Data Collection record.
b) Select *Active* and set the *Run As* credentials to *System Administrator*.
c) Set the run time of day to a period of low usage.
Setting up DevOps tools

Set up your environment to connect DevOps to your tools, discover tool configuration, and import historical data into the DevOps application.

<table>
<thead>
<tr>
<th>Tool</th>
<th>Required DevOps applications</th>
</tr>
</thead>
<tbody>
<tr>
<td>Azure DevOps</td>
<td>DevOps application and DevOps Integrations application</td>
</tr>
<tr>
<td>• Azure Boards</td>
<td></td>
</tr>
<tr>
<td>• Azure Repos</td>
<td></td>
</tr>
<tr>
<td>• Azure Pipelines</td>
<td></td>
</tr>
<tr>
<td>Bitbucket Server/Enterprise</td>
<td>DevOps application</td>
</tr>
<tr>
<td>GitHub and GitHub Enterprise</td>
<td>DevOps application</td>
</tr>
<tr>
<td>GitLab</td>
<td>DevOps application and DevOps Integrations application</td>
</tr>
<tr>
<td>• SCM</td>
<td></td>
</tr>
<tr>
<td>• CI/CD</td>
<td></td>
</tr>
<tr>
<td>Jenkins</td>
<td>DevOps application and DevOps Integrations application</td>
</tr>
<tr>
<td>Jira</td>
<td>DevOps application</td>
</tr>
<tr>
<td>ServiceNow Agile Development 2.x</td>
<td>DevOps application</td>
</tr>
</tbody>
</table>

Connection and credential alias

Before you set up your tool records in DevOps, your admin configures the DevOps connection and credential alias (CreateDevOpsTool) to gain access to the tools in your environment.

**Note:** You only need to configure the CreateDevOpsTool connection and credential alias once to set up all of your tools in DevOps.

Tool actions

A common set of actions is required to integrate DevOps with each tool in your environment.
<table>
<thead>
<tr>
<th>Action</th>
<th>Goal</th>
</tr>
</thead>
<tbody>
<tr>
<td>Connect</td>
<td>DevOps accesses the tool with correct credentials and gets the webhook URL.</td>
</tr>
</tbody>
</table>
| Discover | DevOps discovers all tool information.  
- Planning tool application plans  
- Coding tool repositories  
- Orchestration tool orchestration tasks and pipelines |
| Configure | DevOps configures the webhook URL in the source tool so notifications from the tool can be received by DevOps. |
| Import | DevOps imports all tool data and enables tracking.  
- Planning tool application plan work item data (and plan versions, features)  
- Coding tool repository branch and commit data  
- Orchestration tool task execution data |

Manual configuration mode

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.
Commits and task executions

Run commits in DevOps are associated to a task execution.

Note: 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.

<table>
<thead>
<tr>
<th>Tool</th>
<th>Max commits per payload</th>
</tr>
</thead>
<tbody>
<tr>
<td>Azure DevOps</td>
<td>25</td>
</tr>
<tr>
<td>GitHub</td>
<td>1000</td>
</tr>
<tr>
<td>GitLab</td>
<td>20</td>
</tr>
</tbody>
</table>

Azure DevOps integration with DevOps

Integrate Azure DevOps tools with DevOps using the DevOps Integrations application.

Note: You need to install the DevOps Integrations application to integrate Azure DevOps tools with DevOps.

Azure DevOps tools supported:

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

Azure DevOps extension

You can use the ServiceNow DevOps extension for Azure DevOps on Visual Studio Marketplace to integrate your Azure pipeline with the ServiceNowDevOps application.

The ServiceNow DevOps extension includes:

- ServiceNow DevOps service connection
- ServiceNow DevOps Release Gate (starting with version 1.18)
• Azure build (CI) pipeline agent and server job custom tasks

**Connect DevOps to Azure DevOps tools**

Configure a connection and credential alias and create an Azure DevOps tool record in DevOps.

Configure connection and credential alias - Azure DevOps

Before you set up a tool record in DevOps, your admin configures the DevOps CreateDevOpsTool connection and credential alias to allow access to the tools in your environment.

Starting with version 1.15, DevOps connection and credential aliases (DevOpsAlias1 through DevOpsAlias10) are used to connect automatically when you set up your DevOps planning, coding, and orchestration tools.


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.

Note: If additional sys_alias records for DevOps are needed, they must be created by an admin in the sn_devops application scope.

You can reuse a DevOpsAlias alias no longer in use by inactivating the HTTP connection, if needed.

Role required: sn_devops.admin

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: Starting with version 1.15, 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.

1. Navigate to 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: Starting with version 1.15, 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 Azure DevOps tool record in DevOps**

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

Role required: sn_devops.admin

Actions:

- **Connect** to Azure DevOps and get the webhook URL when you submit a DevOps tool record.
- **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.

1. Create a tool record in DevOps to automatically connect to Azure DevOps and get the webhook URL.

   **Note:** You need at least two unused connection and credential aliases to create an Azure DevOps tool (one for a build pipeline and one for a release pipeline).

   a) Navigate to **DevOps > Tools > Create New** and create a record.

   b) Enter a **Tool Name** and fill in the tool details.

<table>
<thead>
<tr>
<th>Tool Integration</th>
<th>Azure DevOps</th>
</tr>
</thead>
<tbody>
<tr>
<td>Tool URL</td>
<td>Azure DevOps tool URL in the format: <a href="https://dev.azure.com/organization/project">https://dev.azure.com/organization/project</a></td>
</tr>
<tr>
<td>Tool Username</td>
<td>Azure DevOps username</td>
</tr>
<tr>
<td>Tool Password / Access Token</td>
<td><strong>Personal access token (PAT)</strong></td>
</tr>
</tbody>
</table>

   **Note:** Only a personal access token is supported.

   When you generate a Personal access token (PAT) for Azure DevOps, you must select the scopes to authorize if you are not granting complete access.

   *Configuration Item* and *MID Server* are optional. Select MID Server for tools on premise attached to a MID Server.

   c) Click **Submit**.

   The tool is automatically **Connected Successfully** using a connection alias, and HTTP tool connection (basic authentication credential).

   **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.

2. Click **Discover** to discover tool configuration, including existing application plans (projects associated with the tool), repositories, orchestration tasks, and pipelines.
Records are added to the corresponding related lists.

3. Automatically configure webhook URLs in Azure DevOps.
   a) Click Configure.
   b) Enter your Integration user (DevOps Integration User) and Password.
      The DevOps Integration User is provided with the DevOps application and is used for inbound authentication from your tool to your ServiceNow instance.

      Note: You must configure the password before the DevOps Integration User can configure a tool.

      Notifications are sent from Azure DevOps tools to DevOps as the DevOps Integration User.

      The Track field is automatically set to True in configured plan and repository records.

4. For discovered plans, import historical data from the tool and add an app to tie your tools together.
   a) Open the plan record from the Plans related list and click 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 is not supported for Agile Boards CMMI process.

   b) In the Apps related list, click Edit... to select an App to associate with the plan (project), or click New to create one.

      Note: A plan must have an associated App record to complete the planning tool setup.

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.
Azure DevOps tool record in DevOps
For Azure pipelines, *Model an Azure pipeline in DevOps* to complete the configuration in DevOps.

- 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**

Starting with version 1.12, model an Azure pipeline by mapping the pipeline to an app, and mapping DevOps pipeline steps to Azure pipeline jobs.

Role required: sn_devops.admin

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 project name and pipeline name as specified in Azure DevOps Pipelines in path format.

      For example, My Project/My Classic Build Pipeline.

      
      **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 create DevOps steps to map to each Azure pipeline job so an orchestration task can be created.

   Steps can be created in one of the following ways.

   - Starting with version 1.18, automatically create and map pipeline steps in DevOps by running your Azure pipeline.

      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).

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

<table>
<thead>
<tr>
<th>Name</th>
<th>Name of the pipeline step.</th>
</tr>
</thead>
<tbody>
<tr>
<td>Type</td>
<td>Pipeline step type.</td>
</tr>
<tr>
<td></td>
<td>• Build and Test</td>
</tr>
<tr>
<td></td>
<td>• Test</td>
</tr>
<tr>
<td></td>
<td>• Deploy</td>
</tr>
<tr>
<td></td>
<td>• Deploy and Test</td>
</tr>
<tr>
<td></td>
<td>• Manual</td>
</tr>
<tr>
<td></td>
<td>• Prod Deploy</td>
</tr>
</tbody>
</table>
Order | Order in which the steps are run.  

**Note:** The step order determines the order of the cards in the Pipeline UI. Starting with version 1.18, the order of the cards in the Pipeline UI is by task execution.

Orchestration stage | Azure pipeline job name (case-sensitive).  

**Note:** For step association with Azure pipeline jobs, the Orchestration stage field must be configured.

---

Once orchestration tasks are created, *associate* each orchestration task in the Orchestration Tasks related list with a DevOps pipeline step.

3. Optional: Enable change control in a step for *change acceleration*.

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.

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

Change control | Enable *change acceleration* for the step.  

- True
- False  

**Note:** Change control for release pipelines is supported only in the first step of the required stage.

Change approval group | Approval group for the change request. The change approval group becomes the Assignment group in the DevOps change request.

**Note:** 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.  

- Normal (default)
- Standard
- Emergency

Template | List of templates to use to auto populate fields for Normal or Emergency change requests. Select a template or create a new one.

**Note:** This field is shown only when Change type is Normal or Emergency.
### Standard change template

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

List of standard change templates to use for Standard change requests.

**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.

**Note:** Change control check box must be selected.

You can set up change control in your Azure DevOps YAMLS or Classic Azure pipeline using the Azure Invoke REST API or the ServiceNow DevOps extension for Azure DevOps.

4. Navigate to **DevOps > Tools > Orchestration Tools** and in the Azure tool record, copy the DevOps Webhook URL field value.

The webhook URL contains the DevOps location for Azure DevOps Pipelines to send messages, including the sys_id for the tool:

```plaintext
https://<devops.integration.user>:<password>@<your-instance>.service-now.com/api/sn_devops/v1/devops/tool/event/{sys_id of the record azure tool that was created}
```
DevOps app
### DevOps pipeline

In the image, we see a screen from a DevOps pipeline management tool, likely ServiceNow, given the context and logos. The pipeline is configured for an application named CorpSite, and the pipeline steps are:

1. **Build**
   - Order: 100
   - Change control: False
   - Change approval group: Empty
   - Configuration item: Empty
   - Change controlled branches:

2. **Test**
   - Order: 200
   - Change control: False
   - Change approval group: Empty
   - Configuration item: Empty
   - Change controlled branches:

3. **Server**
   - Order: 300
   - Change control: True
   - Change approval group: Change Management
   - Configuration item: All
   - Change controlled branches:

The pipeline is set up with three main steps: Build, Test, and Server, each with specific settings for order, change control, and approval groups.
## DevOps pipeline step

**Configure the Azure pipeline for DevOps**
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 (starting with version 1.18)
- 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.

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

<table>
<thead>
<tr>
<th>Azure pipeline type</th>
<th>Values</th>
</tr>
</thead>
<tbody>
<tr>
<td>Build</td>
<td>• buildNumber</td>
</tr>
<tr>
<td></td>
<td>• isMultiBranch</td>
</tr>
<tr>
<td></td>
<td>• branchName</td>
</tr>
<tr>
<td>Release</td>
<td>• releaseNumber</td>
</tr>
<tr>
<td></td>
<td>• projectName</td>
</tr>
</tbody>
</table>

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

Build pipeline:

```powershell
- task: InvokeRESTAPI@1
  inputs:
    connectionType: 'connectedServiceName'
    serviceConnection: 'change1'
    method: 'POST'
    body: |
    {
      "buildNumber": "$(build.buildId)",
      "isMultiBranch": "true",
      "branchName": "$(build.sourceBranchName)"
    }
  waitForCompletion: 'true'
```

Release pipeline:

```powershell
- 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 ServiceNow DevOps extension for Azure DevOps to send build and release notifications from your Azure pipeline to ServiceNow DevOps application.

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

- Starting with version 1.18, **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 Job Notification** custom task
    
    Required (in DevOps versions 1.16 and earlier) for agentless (server) jobs to send notifications from the Azure pipeline to ServiceNow DevOps.

  - **Note**: Starting with version 1.17, job notification tasks are no longer required.

  - **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.

  - **Note**: The ServiceNow DevOps Server Change Acceleration custom task can only be used with ServiceNow DevOps Server Job Notification custom tasks.

  - Starting with version 1.13, **ServiceNow DevOps Server Package Registration** custom task
    
    Required for agentless (server) jobs to register a package in the ServiceNow instance

  - Starting with version 1.13, **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 Job Notification** custom task
    
    Required (in DevOps versions 1.16 and earlier) for agent jobs to send notifications from the Azure pipeline to ServiceNow DevOps.

  - **Note**: Starting with version 1.17, job notification tasks are no longer required.

  - Starting with version 1.13, **ServiceNow DevOps Agent Package Registration** custom task
    
    Required for agent jobs to register a package in the ServiceNow instance

  - Starting with version 1.13, **ServiceNow DevOps Agent Artifact Registration** custom task
    
    Required for agent jobs to register an artifact in the ServiceNow instance

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.

<table>
<thead>
<tr>
<th><strong>ServiceNow URL</strong></th>
<th>https://&lt;your-instance&gt;.service-now.com/</th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>Tool ID</strong></td>
<td>The sys_id of the orchestration tool.</td>
</tr>
<tr>
<td></td>
<td>You can copy this value from the webhook URL (toolid value), or obtain it directly using the Copy sys_id command on the Orchestration Tool form.</td>
</tr>
<tr>
<td><strong>Username</strong></td>
<td>devops.integration.user</td>
</tr>
<tr>
<td><strong>Password</strong></td>
<td>Password for DevOps Integration User.</td>
</tr>
<tr>
<td><strong>Service connection name</strong></td>
<td>DevOps Connection</td>
</tr>
<tr>
<td><strong>Grant access permission to all pipelines</strong></td>
<td>Select check box.</td>
</tr>
</tbody>
</table>
Set up an Azure build (CI) pipeline in DevOps

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

1. For DevOps versions 1.16 and earlier, in Azure DevOps Pipelines, navigate to the Tasks section of your Azure pipeline jobs and Add (+) the appropriate ServiceNow DevOps job notification custom task at the beginning and end of each job in your build (CI) pipeline.

   Note: ServiceNow DevOps job notification custom tasks must be added twice per job (to indicate both the beginning, and end of each job) for notifications to be sent to DevOps successfully.

   - For Build and Test job types, use the ServiceNow DevOps Agent Job Notification custom task.
   - For Server job type, use the ServiceNow DevOps Server Job Notification custom task.

2. For DevOps versions 1.16 and earlier, in Azure DevOps Pipelines, configure ServiceNow DevOps job notification custom tasks.

<table>
<thead>
<tr>
<th>Display name</th>
<th>ServiceNow Job Notification</th>
</tr>
</thead>
<tbody>
<tr>
<td>ServiceNow endpoint</td>
<td>My Connection</td>
</tr>
</tbody>
</table>
Upstream Job | Indicates the previous job in line. For example, the job before Test may be Build.
---|---
Job Execution Phase | Indicates where this ServiceNow Job Notification custom task is placed in the list of job tasks.
• STARTED | First task in the job.
• COMPLETED | Last task in the job.

Note: The ServiceNow Job Notification custom task must be added to the task list as both the first task of the job and the last task of the job.

3. Optional: 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.

Note: The ServiceNow DevOps Server Change Acceleration task must be between the ServiceNow DevOps Server Job Notification tasks. You must also use the server job notification tasks when using the change acceleration task.

<table>
<thead>
<tr>
<th>Display name</th>
<th>ServiceNow Change Acceleration</th>
</tr>
</thead>
<tbody>
<tr>
<td>ServiceNow endpoint</td>
<td>My Connection</td>
</tr>
</tbody>
</table>

| Upstream job executed | Indicates the previous job in line. For example, the job before Server may be Test. |

4. Optional: 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.

Azure pipeline: ServiceNow DevOps job notification custom tasks (no longer required starting with DevOps version 1.17)
```yaml
trigger:
  - master
  name: $(Date:yyyyMMdd)(Rev:.r)

stages:
  - stage: Build
    jobs:
      - job: 'Build'
        steps:
          - task: ServiceNow DevOps Agent Job Notification@2
            inputs:
              connectedServiceName: 'devopsdev'
              Phase: 'STARTED'

          - task: Bash@3
            inputs:
              targetType: 'inline'
              script: |
    # Write your commands here
    echo 'Hello world'

          - task: ServiceNow DevOps Agent Job Notification@2
            inputs:
              connectedServiceName: 'devopsdev'
              Phase: 'COMPLETED'
```
Azure pipeline: ServiceNow DevOps Change Acceleration custom task
Set up an Azure release (CD) pipeline in DevOps
Starting with version 1.18, use the ServiceNow DevOps extension for Azure DevOps to configure change control, and artifacts and packages in your Azure release (CD) pipeline.

**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.

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.

<table>
<thead>
<tr>
<th>Time between re-evaluation of gates</th>
<th>Re-evaluation interval in minutes, hours, or days.</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>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.</td>
</tr>
<tr>
<td></td>
<td>No user intervention is required for further attempts of re-evaluation after the change request has already been approved, rejected, or canceled.</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>Timeout after which gates fail</th>
<th>Timeout value in minutes, hours, or days.</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>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.</td>
</tr>
</tbody>
</table>

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.
Azure release pipeline pre-deployment gate configuration
Artifact setup - build pipeline source

```yaml
- task: ServiceNow-DevOps-Agent-Artifact-Registration@1
  inputs:
    connectedServiceName: Artifacts payload
    artifactsPayload: |
      "artifacts": ["name": "BuildDrop",
          "version": "1.$(Build.BuildId)",
          "semanticVersion": "1.$(Build.BuildId).0",
          "BuildNumber": "$(Build.BuildNumber)",
          "repositoryName": "DeployableRepo"
      ]
```
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.

Role required: sn_devops.admin

1. Navigate to **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, since orchestration tasks are mapped to a certain step in your DevOps pipeline, you can track the activity in each stage of your pipeline.
DevOps associate steps
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

<table>
<thead>
<tr>
<th>ServiceNow DevOps</th>
<th>Azure Boards Basic</th>
<th>Azure Boards Agile</th>
<th>Azure Boards Scrum</th>
</tr>
</thead>
<tbody>
<tr>
<td>Task</td>
<td>Task</td>
<td>Task</td>
<td>Task</td>
</tr>
<tr>
<td>Bug</td>
<td>Issue</td>
<td>Bug</td>
<td>Bug</td>
</tr>
<tr>
<td>Story</td>
<td>--</td>
<td>User story</td>
<td>Product backlog item</td>
</tr>
<tr>
<td>Epic</td>
<td>Epic</td>
<td>Epic</td>
<td>Epic</td>
</tr>
<tr>
<td>Feature</td>
<td>--</td>
<td>Feature</td>
<td>Feature</td>
</tr>
</tbody>
</table>

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

Work item state mapping

<table>
<thead>
<tr>
<th>ServiceNow DevOps</th>
<th>Azure Boards Basic</th>
<th>Azure Boards Agile</th>
<th>Azure Boards Scrum</th>
</tr>
</thead>
<tbody>
<tr>
<td>Planned</td>
<td>To Do</td>
<td>New</td>
<td>New</td>
</tr>
<tr>
<td>WIP</td>
<td>Doing</td>
<td>Active</td>
<td>In Progress</td>
</tr>
<tr>
<td>Complete</td>
<td>Done</td>
<td>Ready</td>
<td>Done</td>
</tr>
<tr>
<td>Deleted</td>
<td>Deleted</td>
<td>Completed</td>
<td>Removed</td>
</tr>
</tbody>
</table>
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 included 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.

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

DevOpsAzureDevOpsWorkItemHelper.prototype =
Object.extendObject(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'
});
```

Bitbucket integration with DevOps

Integrate Bitbucket coding tool with DevOps by configuring a connection and credential alias, and creating a tool record in DevOps.
Configure connection and credential alias - Bitbucket

Before you set up a tool record in DevOps, your admin configures the DevOps CreateDevOpsTool connection and credential alias to allow access to the tools in your environment.

Starting with version 1.15, DevOps connection and credential aliases (DevOpsAlias1 through DevOpsAlias10) are used to connect automatically when you set up your DevOps planning, coding, and orchestration tools.


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.

Note: If additional sys_alias records for DevOps are needed, they must be created by an admin in the sn_devops application scope.
You can reuse a DevOpsAlias alias no longer in use by inactivating the HTTP connection, if needed.

Role required: sn_devops.admin

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: Starting with version 1.15, 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.

1. Navigate to 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: Starting with version 1.15, 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 a Bitbucket tool record in DevOps

Create a Bitbucket tool record in DevOps to connect and discover repositories.

Role required: sn_devops.admin
Actions:
- **Connect** to Bitbucket and get the webhook URL when you submit a DevOps tool record.
- **Discover** repositories.
- **Configure** the webhook in the Bitbucket repository.
- **Import** branch and commit records.

Starting with version 1.13, capture tags from Bitbucket coding tool commits.

1. **Create a tool record in DevOps to automatically connect to Bitbucket 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.

<table>
<thead>
<tr>
<th>Tool Integration</th>
<th>Bitbucket</th>
</tr>
</thead>
<tbody>
<tr>
<td>Tool URL</td>
<td>Bitbucket tool URL</td>
</tr>
<tr>
<td></td>
<td>For example: <a href="https://bitbucket.com">https://bitbucket.com</a></td>
</tr>
<tr>
<td>Tool Username</td>
<td>Bitbucket username</td>
</tr>
<tr>
<td>Tool Password / Access Token</td>
<td>Bitbucket password or access token</td>
</tr>
</tbody>
</table>

   **Configuration Item** and **MID Server** are optional. Select MID Server for tools on premise attached to a MID Server.

   c) Click **Submit**.
   The tool is automatically **Connected Successfully** using a connection alias, and HTTP tool connection (basic authentication credential).

   **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.

2. **Click Discover** to discover existing repositories for the coding tool.
   Repository records are added to the Repositories related list.

3. **Automatically configure the webhook URL in a Bitbucket repository to send notifications to the DevOps tool.**
   a) Open the discovered repository record from the Repositories related list and click **Configure**.
   b) Enter your **Integration user** (DevOps Integration User) and **Password**.
   The DevOps Integration User is provided with the DevOps application and is used for inbound authentication from your tool to your ServiceNow instance.

   **Note:** You must configure the password before the DevOps Integration User can configure a tool.

   Notifications are sent from the repository to DevOps as the DevOps Integration User.
   The **Track** field is automatically set to **True** in the repository record.

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

   **Note:** Import by date range is not supported for Bitbucket coding tool.

   Imported branch records and commit records from the repository are added to the corresponding related lists.
5. In the **App** field, click the lookup list and select an App record to associate with the repository, or click **New** to create one.
GitHub integration with DevOps

Integrate GitHub coding tool with DevOps by configuring a connection and credential alias, and creating a tool record in DevOps.

Both GitHub and GitHub Enterprise are supported.

Connect using:

- Basic authentication
- OAuth 2.0 credentials (Starting with version 1.17)

Connect DevOps to GitHub

Configure a connection and credential alias and create a GitHub tool record in DevOps.

Configure connection and credential alias - GitHub

Before you set up a tool record in DevOps, your admin configures the DevOps CreateDevOpsTool connection and credential alias to allow access to the tools in your environment.

Starting with version 1.15, DevOps connection and credential aliases (DevOpsAlias1 through DevOpsAlias10) are used to connect automatically when you set up your DevOps planning, coding, and orchestration tools.


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.

Note: If additional sys_alias records for DevOps are needed, they must be created by an admin in the sn_devops application scope.

You can reuse a DevOpsAlias alias no longer in use by inactivating the HTTP connection, if needed.

Role required: sn_devops.admin

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: Starting with version 1.15, 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.

1. Navigate to 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).
4. On the Connection form, enter https://<instance name>.service-now.com/api/now/table for the Connection URL.

Create a GitHub tool record in DevOps
Create a GitHub tool record in DevOps to connect and discover repositories.

Role required: sn_devops.admin

Actions:
• **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.

Starting with version 1.17, using **OAuth 2.0 credentials with GitHub** integration is supported.

Starting with version 1.13, capture tags from GitHub coding tool commits.

1. Create a tool record in DevOps to automatically connect to GitHub 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.

<table>
<thead>
<tr>
<th>Tool Integration</th>
<th>GitHub</th>
</tr>
</thead>
<tbody>
<tr>
<td>(GitHub Enterprise is for on-premise configuration.)</td>
<td></td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>Tool URL</th>
<th>GitHub tool URL. For example:</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td><a href="https://api.github.com">https://api.github.com</a></td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>GitHub credential type (starting with version 1.17)</th>
<th>• Basic Auth</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>• GitHub username</td>
</tr>
<tr>
<td></td>
<td>• GitHub password or access token</td>
</tr>
<tr>
<td></td>
<td>• OAuth</td>
</tr>
<tr>
<td></td>
<td>GitHub Tool Credential</td>
</tr>
<tr>
<td></td>
<td>See Setting up GitHub OAuth 2.0 credentials for DevOps.</td>
</tr>
</tbody>
</table>

**Configuration Item** and **MID Server** are optional. Select MID Server for tools on premise attached to a MID Server.

c) **Click Submit.**

The tool is automatically **Connected Successfully** using a connection alias, and HTTP tool connection (basic authentication or OAuth credential).

**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...
2. Click **Discover** to discover existing repositories for the coding tool. Repository records are added to the Repositories related list.

3. Automatically configure the webhook URL in a GitHub repository to send notifications to the DevOps tool.
   a) Open the discovered repository from the Repositories related list and click **Configure**.
   b) Enter your **Integration user** (DevOps Integration User) and **Password**.
      
      The DevOps Integration User is provided with the DevOps application and is used for inbound authentication from your tool to your ServiceNow instance.

      **Note:** You must configure the password before the DevOps Integration User can configure a tool.

      Notifications are sent from the repository to DevOps as the DevOps Integration User.

      The **Track** field is automatically set to **True** in the repository record.

4. 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.

5. In the **App** field, click the lookup list and select an App record to associate with the repository, or click **New** to create one.
### GitHub tool record in DevOps

<table>
<thead>
<tr>
<th>Name</th>
<th>Github</th>
<th>Connection state</th>
<th>Last discovery</th>
<th>Configuration item</th>
</tr>
</thead>
<tbody>
<tr>
<td>Tool</td>
<td>Github</td>
<td>Connected</td>
<td>2020-09-05 10:43:02</td>
<td></td>
</tr>
<tr>
<td>Connection alias</td>
<td>Github</td>
<td></td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

**Tool URL**: [https://github.com](https://github.com)


<table>
<thead>
<tr>
<th>Tool Connections</th>
<th>Import Requests</th>
<th>Repositories</th>
</tr>
</thead>
<tbody>
<tr>
<td>Connections</td>
<td></td>
<td></td>
</tr>
<tr>
<td>Name</td>
<td>Connection type</td>
<td>Active</td>
</tr>
<tr>
<td>Github</td>
<td>HTTP(s) Connection</td>
<td>true</td>
</tr>
<tr>
<td>Github</td>
<td>Credential</td>
<td></td>
</tr>
<tr>
<td>Github</td>
<td>Domains</td>
<td></td>
</tr>
</tbody>
</table>
Setting up GitHub OAuth 2.0 credentials for DevOps

Starting with version 1.17, integrate your GitHub account with your ServiceNow instance by creating a custom OAuth application in GitHub and authenticating requests from ServiceNow DevOps.

You can set up OAuth 2.0 credentials for GitHub Apps and OAuth Apps.

- GitHub Apps
  - JWT
  - Authorization Code

- OAuth Apps

OAuth 2.0 credentials for GitHub Apps - JWT

Starting with version 1.17, configure your GitHub account, generate and attach the JKS certificate, create a JWT signing key and provider, register GitHub, and create a credential record for the GitHub App provider.

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.

GitHub requirements:

- GitHub account
- GitHub App configured to integrate with ServiceNow

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

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, uncheck the Expire user authorization tokens field.
5. In the Webhook section, uncheck the Active field.
6. Leave the remaining fields empty (default).
7. In the Repository permissions section, configure these settings.

<table>
<thead>
<tr>
<th>Contents</th>
<th>Read-only</th>
</tr>
</thead>
<tbody>
<tr>
<td>Metadata</td>
<td>Read-only</td>
</tr>
<tr>
<td>Webhooks</td>
<td>Read and write</td>
</tr>
</tbody>
</table>

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.

Generate the JKS certificate for GitHub

Generate a JKS certificate for the JWT authentication of ServiceNow DevOps requests for GitHub.

Role required: admin

1. Make a copy of the GitHub App private key previously generated, and rename it to use the .key extension.
2. Create a CA signed certificate using the GitHub App private key:

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

3. Enter the required details.

4. 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>.key -certfile <certificate-name>.pem -out <PKCS-12-file-name>.p12
```

5. Provide the export password.

6. Create the JKS file:

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

7. 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 instance.

Ensure the availability of a valid Java KeyStore certificate.

Role required: admin

1. Navigate to **System Definition > Certificates**.
2. Click **New**.
3. Complete the form.

**X.509 Certificate form fields**

<table>
<thead>
<tr>
<th>Field</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>Name</td>
<td>Enter a name to uniquely identify the record. For example, My GitHub App Certificate.</td>
</tr>
<tr>
<td>Notify on expiration</td>
<td>Define users to be notified when the certificate expires.</td>
</tr>
<tr>
<td>Warn in days to expire</td>
<td>Enter the number of days to send a notification before the certificate expires.</td>
</tr>
<tr>
<td>Active</td>
<td>Enable</td>
</tr>
<tr>
<td>Type</td>
<td>Select Java Key Store</td>
</tr>
<tr>
<td>Expires in days</td>
<td>Enter the number of days until the certificate expires.</td>
</tr>
<tr>
<td>Key store password</td>
<td>Enter a password associated with the certificate (hint: the destination keystore password previously created).</td>
</tr>
<tr>
<td>Short description</td>
<td>Enter a summary about the certificate.</td>
</tr>
</tbody>
</table>

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

5. Click **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.

1. Navigate to **System OAuth > JWT Keys**.
2. Click **New**.
3. Complete the form.

### JWT Keys form fields

<table>
<thead>
<tr>
<th>Field</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>Name</td>
<td>Enter a name to uniquely identify the JWT signing key. For example, <em>My GitHub App JWT Key</em>.</td>
</tr>
<tr>
<td>Signing Keystore</td>
<td>Select the valid JKS certificate attached in the previous task. For example, <em>My GitHub App Certificate</em>.</td>
</tr>
<tr>
<td>Key Id</td>
<td>Enter a key Id to identify which key is used when multiple keys are used to sign tokens.</td>
</tr>
<tr>
<td>Signing Algorithm</td>
<td>Select an algorithm to sign with the JWT key (hint: RSA 256).</td>
</tr>
<tr>
<td>Signing Key Password</td>
<td>Enter a password associated with the signing key (hint: the source keystore password previously created).</td>
</tr>
<tr>
<td>Active</td>
<td>Enable.</td>
</tr>
</tbody>
</table>

4. Click **Submit**.

Create a JWT provider for your GitHub signing key

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

1. Navigate to **System OAuth > JWT Providers**.
2. Click **New**.
3. On the **JWT Provider** form, fill the values.

### JWT Provider form fields

<table>
<thead>
<tr>
<th>Field</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>Name</td>
<td>Enter a name to uniquely identify the JWT provider. For example, <em>My GitHub App JWT Provider</em>.</td>
</tr>
<tr>
<td>Expiry Interval (sec)</td>
<td>Enter a number in seconds to set the lifespan of JWT provider tokens (Hint: You can leave it as default).</td>
</tr>
<tr>
<td>Signing Configuration</td>
<td>Select the JWT signing key previously created. For example, <em>My GitHub App JWT Key</em>.</td>
</tr>
</tbody>
</table>

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

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 a empty Application Registries form.

4. Complete the form.

<table>
<thead>
<tr>
<th>Field</th>
<th>Value required</th>
</tr>
</thead>
<tbody>
<tr>
<td>Name</td>
<td>Enter any name to uniquely identify the record. For example, enter My GitHub App Provider.</td>
</tr>
<tr>
<td>Client ID</td>
<td>Enter the client ID of your GitHub App (hint: available in the <strong>About</strong> section of your GitHub App configuration in GitHub).</td>
</tr>
<tr>
<td>Client Secret</td>
<td>Enter the client secret of your GitHub App (hint: available in the <strong>About</strong> section of your GitHub App configuration in GitHub).</td>
</tr>
<tr>
<td>OAuth API script</td>
<td>Select OAuthDevOpsGitHubHandler.</td>
</tr>
<tr>
<td>Default Grant type</td>
<td>Select JWT Bearer.</td>
</tr>
<tr>
<td>Authorization URL</td>
<td>Enter <a href="https://github.com/login/oauth/authorize">https://github.com/login/oauth/authorize</a>. For an on-premise deployment, use the proper GitHub host URL.</td>
</tr>
<tr>
<td>Token URL</td>
<td>Enter <a href="https://github.com/login/oauth/access_token">https://github.com/login/oauth/access_token</a>. For an on-premise deployment, use the proper GitHub host URL.</td>
</tr>
</tbody>
</table>

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 JWT Bearer. For example, **OAuth Entity Profile** is created with default **Name**, My GitHub App Provider default_profile

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, uncheck the **Is default** flag and save the form.
9. Back on the **Application Registry** form, in the **OAuth Entity Profiles** related list, create a new OAuth Entity Profile with the **Is default** flag set to True, and **Grant type** set to Authorization code.

Create a credential record for GitHub App provider (JWT)
Create a credential record to the GitHub App provider previously created to authorize actions.

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

**OAuth 2.0 credentials for GitHub Apps - Authorization Code**
Starting with version 1.17, configure your GitHub account, register GitHub, and create a credential record for the GitHub App provider.

**Role required:** oauth_admin

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.

**GitHub requirements:**
- GitHub account
- GitHub App configured to integrate with ServiceNow

Complete these steps from your GitHub account. See *[Building GitHub Apps](https://developer.github.com/v3/apps/)* on the GitHub Developer site for instructions on creating and configuring custom applications.

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, uncheck the **Expire user authorization tokens** field.
5. In the **Webhook** section, uncheck the **Active** field.
6. Leave the remaining fields empty (default).
7. In the **Repository permissions** section, configure these settings.

<table>
<thead>
<tr>
<th>Contents</th>
<th>Read-only</th>
</tr>
</thead>
<tbody>
<tr>
<td>Metadata</td>
<td>Read-only</td>
</tr>
<tr>
<td>Webhooks</td>
<td>Read and write</td>
</tr>
</tbody>
</table>

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.

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.
   
<table>
<thead>
<tr>
<th>Field</th>
<th>Value required</th>
</tr>
</thead>
<tbody>
<tr>
<td>Name</td>
<td>Enter any name to uniquely identify the record. For example, enter My GitHub App Provider.</td>
</tr>
<tr>
<td>Client ID</td>
<td>Enter the client ID of your GitHub App (hint: available in the About section of your GitHub App configuration in GitHub).</td>
</tr>
<tr>
<td>Client Secret</td>
<td>Enter the client secret of your GitHub App (hint: available in the About section of your GitHub App configuration in GitHub).</td>
</tr>
<tr>
<td>OAuth API script</td>
<td>Select OAuthDevOpsGitHubHandler.</td>
</tr>
<tr>
<td>Default Grant type</td>
<td>Select Authorization Code.</td>
</tr>
<tr>
<td>Authorization URL</td>
<td>Enter <a href="https://github.com/login/oauth/authorize">https://github.com/login/oauth/authorize</a>. For an on-premise deployment, use the proper GitHub host URL.</td>
</tr>
<tr>
<td>Token URL</td>
<td>Enter <a href="https://github.com/login/oauth/access_token">https://github.com/login/oauth/access_token</a>. For an on-premise deployment, use the proper GitHub host URL.</td>
</tr>
</tbody>
</table>

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 to the GitHub App provider previously created to authorize actions.

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

5. Save the record.
6. Click the Get OAuth Token related link to generate the OAuth token.

**OAuth 2.0 credentials for OAuth Apps**

Starting with version 1.17, configure your GitHub account, register GitHub, and create a credential record for the GitHub App Oauth provider.

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.

GitHub requirements:
- GitHub account
- GitHub OAuth App configured to integrate with ServiceNow

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

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

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.

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 Registrations form.
4. Complete the form.
### ServiceNow

Checker Paris DevOps

<table>
<thead>
<tr>
<th>Field</th>
<th>Value required</th>
</tr>
</thead>
<tbody>
<tr>
<td>Client Secret</td>
<td>Enter the client secret of your OAuth App.</td>
</tr>
<tr>
<td>OAuth API script</td>
<td>Select OAuthDevOpsGitHubHandler.</td>
</tr>
<tr>
<td>Default Grant type</td>
<td>Select Authorization Code.</td>
</tr>
<tr>
<td>Authorization URL</td>
<td>Enter <a href="https://github.com/login/oauth/authorize">https://github.com/login/oauth/authorize</a>. For an on-premise deployment, use the proper GitHub host URL.</td>
</tr>
<tr>
<td>Token URL</td>
<td>Enter <a href="https://github.com/login/oauth/access_token">https://github.com/login/oauth/access_token</a>. For an on-premise deployment, use the proper GitHub host URL.</td>
</tr>
</tbody>
</table>

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 Authorization callback URL 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

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.

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

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

**GitLab integration with DevOps**

Integrate GitLab tools with DevOps using the DevOps Integrations application.

**Note:** You need to install the *DevOps Integrations application* to integrate GitLab tools with DevOps.

**GitLab tools supported:**

- GitLab SCM (coding)
- GitLab CI/CD (orchestration) basic CI pipelines

**Note:** Multi-project pipelines are not supported.

**Limitations**

- A pipeline must run to completion one time before change control is enforced.
- If a manual job in GitLab is canceled, or times out before completion, the corresponding change remains in **Open** state until it is manually reconciled.
- Pipeline discovery is limited to the first 100 results using the current filter.
  - To discover additional pipelines, modify the project filter to expand the results for the discovery request.
- Historical import of commits or events is not supported.
  - Those created or updated after the project has been discovered and configured are tracked normally.
  - GitLab Issues is not supported.

**Connect DevOps to GitLab tools**

Configure a connection and credential alias and create a GitLab tool record in DevOps.

*Configure connection and credential alias - GitLab*

Before you set up a tool record in DevOps, your admin configures the DevOps CreateDevOpsTool connection and credential alias to allow access to the tools in your environment.

Starting with version 1.15, DevOps connection and credential aliases (DevOpsAlias1 through DevOpsAlias10) are used to connect automatically when you set up your DevOps planning, coding, and orchestration tools.

**Note:** DevOpsAlias aliases take the place of the sn_devops.GitHub, sn_devops.Jenkins, sn_devops.JIRA, sn_devops.BitBucket aliases in the tool setup.

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.

**Note:** If additional sys_alias records for DevOps are needed, they must be created by an admin in the **sn_devops** application scope.

You can reuse a DevOpsAlias alias no longer in use by inactivating the HTTP connection, if needed.

Role required: sn_devops.admin
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:** Starting with version 1.15, 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.

1. Navigate to **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:** Starting with version 1.15, 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 a GitLab tool record in DevOps*

Create a GitLab tool record in DevOps to connect, discover, and import GitLab tool data.

**Role required:** sn_devops.admin

- **Connect** to GitLab and get the webhook URL when you submit a DevOps tool record.
- **Discover** repositories, orchestration tasks, and pipelines.
- **Configure** webhooks in GitLab (starting with version 1.18).
- **Import** branch records, task execution and step execution records.

**Note:** Import of GitLab commits is not supported.

1. Create a tool record in DevOps to automatically connect to GitLab 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.

<table>
<thead>
<tr>
<th>Tool Integration</th>
<th>GitLab</th>
</tr>
</thead>
<tbody>
<tr>
<td>Tool URL</td>
<td>GitLab tool URL.</td>
</tr>
<tr>
<td></td>
<td>For example:</td>
</tr>
<tr>
<td></td>
<td><a href="https://gitlab.com">https://gitlab.com</a></td>
</tr>
<tr>
<td>Tool Username</td>
<td>GitLab username</td>
</tr>
</tbody>
</table>
Tool Password / Access Token | GitLab access token
--- | ---

**Note:** Only personal access token is supported.

If you are not granting complete access, select scope:

- **api:** Grants complete read/write access to the API, including all groups and projects, the container registry, and the package registry.

**Configuration Item** and **MID Server** are optional. Select MID Server for tools on premise attached to a MID Server.

c) Click **Submit**.

The tool is automatically **Connected Successfully** using a connection alias and HTTP tool connection (basic authentication credential), and an **API key credential** is created.

**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.

2. Click **Discover** to discover tool configuration, including existing repositories, orchestration tasks, and pipelines, and fill in the fields in the project Search Filter window.

**Note:** GitLab repos and pipelines discovered are determined by these filter conditions, and are limited to the first 100 results. To discover additional repos, modify the project filter to expand the results.

<table>
<thead>
<tr>
<th>Owned by me (recommended)</th>
<th>Searches for the repositories in the project that the current user owns.</th>
</tr>
</thead>
<tbody>
<tr>
<td>Currently member of</td>
<td>Searches for the repositories in the project that the current user is a member of.</td>
</tr>
<tr>
<td>Search</td>
<td>Searches for the repositories in projects with the specified text string.</td>
</tr>
</tbody>
</table>

**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.

3. Starting with version 1.18, automatically configure webhook URLs in GitLab.
   a) Click **Configure**.
   b) In the GitLab Configure window, select the applicable projects (containing existing repos and pipelines) to configure webhooks for, and click **Continue**.
   c) Enter your **Integration user** (DevOps Integration User) and **Password**.
The DevOps Integration User is provided with the DevOps application and is used for inbound authentication from your tool to your ServiceNow instance.

**Note:** You must configure the password before the DevOps Integration User can configure a tool.

Notifications are sent from GitLab tools to DevOps as the DevOps Integration User.

The **Track** field is automatically set to **True** in configured repository records.

4. 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.

**Note:** Import of GitLab commits is not supported.

Imported historical data records are added to the corresponding related lists.
For GitLab pipelines, Model a GitLab basic CI pipeline in DevOps to complete the configuration in DevOps.
• Map each pipeline to a specific app.
• Create pipeline steps and map each step to a GitLab pipeline job.
• Configure change control.

Model a GitLab basic CI pipeline in DevOps

Starting with version 1.16, model a GitLab basic CI pipeline by mapping the pipeline to an app, and mapping DevOps pipeline steps to GitLab pipeline jobs.

Role required: sn_devops.admin

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.
      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.
   • Starting with version 1.18, 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.

<table>
<thead>
<tr>
<th>Name</th>
<th>Name of the pipeline step.</th>
</tr>
</thead>
<tbody>
<tr>
<td>Type</td>
<td>Pipeline step type.</td>
</tr>
<tr>
<td></td>
<td>• Build and Test</td>
</tr>
<tr>
<td></td>
<td>• Test</td>
</tr>
<tr>
<td></td>
<td>• Deploy</td>
</tr>
<tr>
<td></td>
<td>• Deploy and Test</td>
</tr>
<tr>
<td></td>
<td>• Manual</td>
</tr>
<tr>
<td></td>
<td>• Prod Deploy</td>
</tr>
</tbody>
</table>

© 2020 ServiceNow, Inc. All rights reserved.
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 names, product names, and logos may be trademarks of the respective companies with which they are associated.
<table>
<thead>
<tr>
<th><strong>Order</strong></th>
<th><strong>Order in which the steps are run.</strong></th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td><img src="#" alt="Note" /> The step order determines the order of the cards in the Pipeline UI. Starting with version 1.18, the order of the cards in the Pipeline UI is by task execution.</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th><strong>Orchestration stage</strong></th>
<th><strong>GitLab pipeline job name (case-sensitive).</strong></th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td><img src="#" alt="Note" /> For step association with GitLab CI pipeline jobs, the Orchestration stage field must be configured.</td>
</tr>
</tbody>
</table>

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

3. Optional: Enable change control in a step for change acceleration.

![Note](#) ServiceNow Change Management must be installed for change acceleration.

<table>
<thead>
<tr>
<th><strong>Change control</strong></th>
<th><strong>Enable change acceleration for the step.</strong></th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td><img src="#" alt="Option" /></td>
</tr>
<tr>
<td></td>
<td><img src="#" alt="Option" /></td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th><strong>Change approval group</strong></th>
<th><strong>Approval group for the change request.</strong></th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>The change approval group becomes the Assignment group in the DevOps change request.</td>
</tr>
<tr>
<td></td>
<td><img src="#" alt="Note" /> Ensure that the selected group has members and a group manager so approver field is not empty.</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th><strong>Change type</strong></th>
<th><strong>Change request type to create.</strong></th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td><img src="#" alt="Option" /> Normal (default)</td>
</tr>
<tr>
<td></td>
<td><img src="#" alt="Option" /> Standard</td>
</tr>
<tr>
<td></td>
<td><img src="#" alt="Option" /> Emergency</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th><strong>Template</strong></th>
<th><strong>List of templates to use to auto populate fields for Normal or Emergency change requests.</strong></th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td><img src="#" alt="Note" /> This field is shown only when Change type is Normal or Emergency.</td>
</tr>
<tr>
<td></td>
<td>Select a template or create a new one.</td>
</tr>
</tbody>
</table>

| **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.

Note: Change control check box must be selected.

You can set up change control in GitLab for manual GitLab jobs.

4. For versions 1.17 and earlier, navigate to DevOps > Tools > Orchestration Tools and in the GitLab tool record, copy the DevOps Webhook URL field value.

The webhook URL contains the DevOps location for GitLab CI pipelines to send messages, including the sys_id for the tool:

http://<devops.integration.user>:@<your-instance>.service-now.com/api/sn_devops/v1/devops/tool/{code | plan | artifact | orchestration | test}?toolId={sys_id of the GitLab tool record}.
Configure 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

Version 1.17 and earlier, configure webhooks in GitLab to send job and push notifications to the DevOps application.

Verify that you have copied the Webhook URL field from the DevOps GitLab tool form into your clipboard.

Role required: sn_devops.admin

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.

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://<devops.integration.user>:<integration user password>@<your-instance>.service-now.com/api/sn_devops/v1/devops/tool/code?
      toolId={sys_id of the GitLab tool record}.
      
      **Note:** Your integration username and password must be specified in the URL.

   b) 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://<devops.integration.user>:<integration user password>@<your-instance>.service-now.com/api/sn_devops/v1/devops/tool/orchestration?
toolId={sys_id of the GitLab tool record}.
```

**Note:** Your integration username and password must be specified in the URL.

b) Select the **Job events** check box.
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 [GitLab 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.

<table>
<thead>
<tr>
<th>Manual execution</th>
<th>Change acceleration in step</th>
<th>Change request approved</th>
<th>Result</th>
</tr>
</thead>
<tbody>
<tr>
<td>Yes</td>
<td>Yes</td>
<td>N/A</td>
<td>If the manual job is under change control, the change is automatically created.</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td>Yes</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td>No</td>
</tr>
<tr>
<td>No</td>
<td>N/A</td>
<td></td>
<td>The manual job waits for manual intervention from the pipeline owner via the GitLab UI (default behavior).</td>
</tr>
<tr>
<td>No</td>
<td>Yes</td>
<td>N/A</td>
<td>The job runs, and the change request is created with the applicable related list data. However, approval or rejection of the change request has no impact on the pipeline run.</td>
</tr>
</tbody>
</table>

**Note:** Parallel jobs are displayed sequentially, based on the order in which the jobs are queued for execution.

### Associate GitLab 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 GitLab pipeline.

Role required: sn_devops.admin

1. Navigate to **DevOps > Tools > Orchestration Tools**, and open the GitLab 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 GitLab.

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

Integrate Jenkins orchestration tool with DevOps using the DevOps Integrations application.

**Note:** You need to install the DevOps Integrations application and the Jenkins plugin for ServiceNow DevOps to integrate Jenkins with DevOps.

Jenkins plugin for ServiceNow DevOps

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.

Connect DevOps to Jenkins

Configure a connection and credential alias and create a Jenkins tool record in DevOps.

**Configure connection and credential alias - Jenkins**

Before you set up a tool record in DevOps, your admin configures the DevOps CreateDevOpsTool connection and credential alias to allow access to the tools in your environment.

Starting with version 1.15, DevOps connection and credential aliases (DevOpsAlias1 through DevOpsAlias10) are used to connect automatically when you set up your DevOps planning, coding, and orchestration tools.

**Note:** DevOpsAlias aliases take the place of the sn_devops.GitHub, sn_devops.Jenkins, sn_devops.JIRA, sn_devops.BitBucket aliases in the tool setup.

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.

**Note:** If additional sys_alias records for DevOps are needed, they must be created by an admin in the sn_devops application scope.

You can reuse a DevOpsAlias alias no longer in use by inactivating the HTTP connection, if needed.

Role required: sn_devops.admin

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:** Starting with version 1.15, 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.

1. Navigate to 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:** Starting with version 1.15, 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 a Jenkins tool record in DevOps**
Create a Jenkins tool record in DevOps to connect, discover, and import orchestration tasks and pipelines.

Role required: sn_devops.admin

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.

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.

   **Note:** Both Freestyle project and Folder type jobs are supported. Default number of levels for Folder type is three.

<table>
<thead>
<tr>
<th>Tool Integration</th>
<th>Jenkins</th>
</tr>
</thead>
<tbody>
<tr>
<td>Tool URL</td>
<td>Jenkins tool URL</td>
</tr>
<tr>
<td></td>
<td>For example:</td>
</tr>
<tr>
<td></td>
<td><code>https://jenkins.com</code></td>
</tr>
<tr>
<td>Tool Username</td>
<td>Jenkins username</td>
</tr>
<tr>
<td>Tool Password / Access Token</td>
<td>Jenkins password or access token</td>
</tr>
</tbody>
</table>

**Configuration Item** and **MID Server** are optional. Select MID Server for tools on premise attached to a MID Server.

   c) Click **Submit**.

   **Note:** Only one Jenkins connection per connection alias is supported. You must create a new connection alias for a second Jenkins connection.

   The tool is automatically **Connected Successfully** using a connection alias, and HTTP tool connection (basic authentication credential).

   **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...
2. Click Discover to discover the existing orchestration tasks (Jenkins stages) and pipelines. Records are added to the corresponding related lists.

3. 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.
### Jenkins tool record in DevOps

<table>
<thead>
<tr>
<th>Name</th>
<th>Jenkins</th>
<th>Connection state</th>
<th>Last discovery</th>
<th>Configuration Item</th>
</tr>
</thead>
<tbody>
<tr>
<td>Tool</td>
<td>Jenkins</td>
<td>Connected</td>
<td>2020-05-15 13:09:34</td>
<td></td>
</tr>
</tbody>
</table>

**Tool URL**: [http://jenkins](http://jenkins)

**Webhook URL**: 
```
https://<DevOps Integration user ID>:<integration user password>@service-now.com/api/sn_devops/v1/devops/tool/code [plan] artifact [orchestration] [test]?toolId=8567c15f6b38c1c6c3942c5806928c6
```

---

**Table**

<table>
<thead>
<tr>
<th>Connections</th>
<th>Name</th>
<th>Connection type</th>
<th>Active</th>
<th>Connection alias</th>
<th>Credential</th>
<th>Domain</th>
</tr>
</thead>
<tbody>
<tr>
<td>Jenkins</td>
<td>HTTP(s) Connection</td>
<td>true</td>
<td>Jenkins</td>
<td>Jenkins</td>
<td>global</td>
<td></td>
</tr>
</tbody>
</table>
**Model a Jenkins pipeline in DevOps** to complete the configuration in DevOps.

- Map each pipeline to a specific app.
- Create pipeline steps and map each step to a Jenkins pipeline stage.
- Configure change control.

---

**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. 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](https://store.servicenow.com) website to download the Jenkins plugin for ServiceNow DevOps.

Role required: sn_devops.admin

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.
   c) Click **Submit**.

2. Open the pipeline record again and in the Steps related list, click **New** to create a DevOps step for each Jenkins pipeline stage so an orchestration task can be created.

Map each DevOps step to a Jenkins pipeline stage (**Orchestration stage** field) so an orchestration task can be created.

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

You can also use a **JenkinsFile** that discovers and maps orchestration tasks to steps instead of mapping manually.

<table>
<thead>
<tr>
<th>Name</th>
<th>Name of the pipeline step.</th>
</tr>
</thead>
<tbody>
<tr>
<td>Type</td>
<td>Pipeline step type.</td>
</tr>
<tr>
<td></td>
<td>• Build and Test</td>
</tr>
<tr>
<td></td>
<td>• Test</td>
</tr>
<tr>
<td></td>
<td>• Deploy</td>
</tr>
<tr>
<td></td>
<td>• Deploy and Test</td>
</tr>
<tr>
<td></td>
<td>• Manual</td>
</tr>
<tr>
<td></td>
<td>• Prod Deploy</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>Order</th>
<th>Order in which the steps are run.</th>
</tr>
</thead>
</table>

**Note:** The step order determines the order of the cards in the **Pipeline UI**.

Starting with version 1.18, the order of the cards in the Pipeline UI is by task execution.
Once orchestration tasks are created, **associate** each orchestration task in the Orchestration Tasks related list with a DevOps pipeline step.

3. **Optional:** Enable change control in a step for **change acceleration**.

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

<table>
<thead>
<tr>
<th>Change control</th>
<th>Enable <strong>change acceleration</strong> for the step.</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>• <strong>True</strong></td>
</tr>
<tr>
<td></td>
<td>• <strong>False</strong></td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>Change approval group</th>
<th>Approval group for the change request.</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>The change approval group becomes the Assignment group in the DevOps change request.</td>
</tr>
</tbody>
</table>

**Note:** Ensure that the selected group has members and a group manager so the approver field is not empty.

<table>
<thead>
<tr>
<th>Change type</th>
<th>Change request type to create.</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>• Normal (default)</td>
</tr>
<tr>
<td></td>
<td>• Standard</td>
</tr>
<tr>
<td></td>
<td>• Emergency</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>Template</th>
<th>List of templates to use to auto populate fields for Normal or Emergency change requests. Select a template or create a new one.</th>
</tr>
</thead>
</table>

**Note:** This field is shown only when Change type is Normal or Emergency.

<table>
<thead>
<tr>
<th>Standard change template</th>
<th>List of standard change templates to use for Standard change requests.</th>
</tr>
</thead>
</table>

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

<table>
<thead>
<tr>
<th>Change controlled branches</th>
<th>(Multibranch only) Comma-separated list of branches under change control. Wildcards are supported.</th>
</tr>
</thead>
</table>

**Note:** Change control check box must be selected.

4. 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}
### DevOps pipeline

<table>
<thead>
<tr>
<th>Name</th>
<th>Type</th>
<th>Order</th>
<th>Change control</th>
<th>Change approval group</th>
<th>Configuration item</th>
<th>Change controlled branches</th>
</tr>
</thead>
<tbody>
<tr>
<td>CorpSite-CI</td>
<td>Build and Test</td>
<td>100</td>
<td>false</td>
<td>(empty)</td>
<td>(empty)</td>
<td>*</td>
</tr>
<tr>
<td>CorpSite-UAT</td>
<td>Deploy</td>
<td>200</td>
<td>false</td>
<td>(empty)</td>
<td>(empty)</td>
<td>*</td>
</tr>
<tr>
<td>CorpSite-Test</td>
<td>Test</td>
<td>300</td>
<td>false</td>
<td>(empty)</td>
<td>(empty)</td>
<td>*</td>
</tr>
<tr>
<td>CorpSite-PROD</td>
<td>Prod Deploy</td>
<td>400</td>
<td>true</td>
<td>Change Management</td>
<td>(empty)</td>
<td>*</td>
</tr>
</tbody>
</table>
DevOps pipeline step
Configure the Jenkins pipeline for DevOps

Configuring the Jenkins pipeline for DevOps

Jenkins tool integration includes configuring the Jenkins pipeline.
Configure Jenkins to send notifications to DevOps
Configure Jenkins to send build notifications to the DevOps application.

Role required: sn_devops.admin

1. In Jenkins, click Manage Jenkins, select the ServiceNow DevOps Enabled check box in the ServiceNow DevOps Configuration section, and fill in the fields.

   **Note:** Values are case-sensitive.

<table>
<thead>
<tr>
<th>Field</th>
<th>URL format/value</th>
</tr>
</thead>
<tbody>
<tr>
<td>Instance URL</td>
<td>https://&lt;your-instance&gt;.servicenow.com</td>
</tr>
<tr>
<td>API Version</td>
<td>v1</td>
</tr>
<tr>
<td>Orchestration Tool ID</td>
<td>The sys_id of the orchestration tool.</td>
</tr>
<tr>
<td></td>
<td>You can copy this value from the webhook URL (toolid value), or obtain it directly using the Copy sys_id command on the Orchestration Tool form.</td>
</tr>
<tr>
<td>Artifact Tool ID</td>
<td>The sys_id of the artifact tool.</td>
</tr>
<tr>
<td>Username</td>
<td>devops.integration.user</td>
</tr>
<tr>
<td>Password</td>
<td>Password for DevOps Integration User.</td>
</tr>
<tr>
<td>Debug Mode</td>
<td>Select check box.</td>
</tr>
</tbody>
</table>

2. Click Test Connection.
   Verify that the connection successful message is shown.

3. Optional: Set ServiceNow job-level properties in the View Configuration > General tab for the job.

<table>
<thead>
<tr>
<th>ServiceNow DevOps Enabled</th>
<th>Clear the check box to disable build notifications for a specific job. Default is selected.</th>
</tr>
</thead>
<tbody>
<tr>
<td>Ignore ServiceNow DevOps errors</td>
<td>Select the check box to prevent job failure if there is an error, incorrect snDevOpsStep ID, for example. Default is not selected.</td>
</tr>
</tbody>
</table>

   **Note:** These properties can be set in a scripted or declarative pipeline, but if they are also set here, these check boxes take precedence.

4. For change control, navigate to Manage Jenkins > Configure System > Jenkins Location, and provide the hostname in the Jenkins URL field.

   **Note:** The Jenkins URL location is required for change acceleration functionality.
Set up Jenkins to send notifications to DevOps

ServiceNow DevOps Settings

- ServiceNow DevOps Enabled
- Ignore ServiceNow DevOps errors

Job-level settings
Jenkins location

Using a declarative or scripted pipeline in DevOps
You can use a Jenkinsfile that discovers orchestration tasks and maps them to a specific step of your app, instead of mapping individually.

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 is set to True by default when you discover orchestration tasks. Tracking is required to receive job notifications from Jenkins.

DevOps Jenkinsfile commands

• 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})

  Where ignoreErrors specifies the setting to prevent job failure if there is an error (true/false)

  Enables change control for each root-level stage that is mapped to a DevOps step.

  Starting with version 1.11, snDevOpsArtifact

  Registers artifacts when configuring DevOps change acceleration for releases.

  Starting with version 1.11, snDevOpsPackage

  Creates a package for artifacts when configuring DevOps change acceleration for releases.

  Note: Stage mapping is only supported for stages at the root level, not nested or parallel stages.
Example Jenkinsfile

```java
node {
  stage("build") {
    snDevOpsStep(enabled:false)
  }
  stage("test") {
    snDevOpsStep(ignoreErrors:false)
  }
  stage("deploy") {
    snDevOpsStep()
    snDevOpsChange()
  }
}
```

**Note:** Historical import of a declarative or scripted pipeline is not supported.

**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)

**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).
echo "deploy in prod"
snDevOpsChange()
}
}

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
      }
    }
  }
}
```

Associate pipeline steps and Jenkins freestyle jobs in DevOps

Associate each orchestration task in the Orchestration Tasks related list with a DevOps pipeline step to track the activity of each stage in your Jenkins freestyle pipeline.

Role required: sn_devops.admin

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

   The Track field is set to True by default when you discover orchestration tasks and pipelines. Tracking is required to receive job notifications from Jenkins.

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

Jira integration with DevOps

Integrate Jira planning tool with DevOps by configuring a connection and credential alias, and creating a tool record in DevOps.

Configure connection and credential alias - Jira

Before you set up a tool record in DevOps, your admin configures the DevOps CreateDevOpsTool connection and credential alias to allow access to the tools in your environment.

Starting with version 1.15, DevOps connection and credential aliases (DevOpsAlias1 through DevOpsAlias10) are used to connect automatically when you set up your DevOps planning, coding, and orchestration tools.


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.

Note: If additional sys_alias records for DevOps are needed, they must be created by an admin in the sn_devops application scope.

You can reuse a DevOpsAlias alias no longer in use by inactivating the HTTP connection, if needed.

Role required: sn_devops.admin

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: Starting with version 1.15, 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.

1. Navigate to 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: Starting with version 1.15, 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 a Jira tool record in DevOps

Create a Jira tool record in DevOps to connect and discover application plans.
Role required: sn_devops.admin

Actions:
- Connect to Jira, get the webhook URL, and configure the webhook in Jira when you submit a DevOps tool record.
- Discover plans.
- Import work item, plan version, and feature records.

1. Create a tool record in DevOps to automatically connect to Jira 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.

<table>
<thead>
<tr>
<th>Tool Integration</th>
<th>Jira</th>
</tr>
</thead>
<tbody>
<tr>
<td>Tool URL</td>
<td>Jira tool URL.</td>
</tr>
<tr>
<td></td>
<td>For example:</td>
</tr>
<tr>
<td></td>
<td><a href="https://jira.com">https://jira.com</a></td>
</tr>
<tr>
<td>Tool Username</td>
<td>Jira username</td>
</tr>
<tr>
<td>Tool Password / Access Token</td>
<td>Jira password or access token</td>
</tr>
</tbody>
</table>

Configuration Item and MID Server are optional. Select MID Server for tools on premise attached to a MID Server.

c) Click Submit.

The tool is automatically Connected Successfully using a connection alias, and HTTP tool connection (basic authentication credential).

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...
2. Click **Discover** to discover existing application plans (projects associated with the tool). Plan records are added to the Plans related list.

3. Automatically configure the webhook URL in Jira.
   a) Click **Configure**.
   b) Enter your **Integration user** (DevOps Integration User) and **Password**.
      The DevOps Integration User is provided with the DevOps application and is used for inbound authentication from your tool to your ServiceNow instance.

      **Note:** You must configure the password before the DevOps Integration User can configure a tool.

      Notifications are sent from the Jira tool to DevOps as the DevOps Integration User.

4. Open the discovered application plan from the Plans related list and click the **Import** related link to import historical data from the application.
   Imported work item records, plan version records, and feature records are added to the corresponding 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.

5. In the Apps related list, click **New** to create an App record to associate with the plan (project).

   **Note:** A plan must have an associated App record to complete the planning tool setup.
Jira tool record in DevOps

© 2020 ServiceNow, Inc. All rights reserved.
ServiceNow, the ServiceNow logo, Now, and other ServiceNow marks are trademarks and/or registered trademarks of ServiceNow, Inc., in the United States company names, product names, and logos may be trademarks of the respective companies with which they are associated.
ServiceNow Agile Development 2.0 integration with DevOps

Integrate Agile Development 2.0 planning tool with DevOps by configuring a connection and credential alias, and creating a tool record in DevOps.

Configure connection and credential alias - Agile Development 2.0

Before you set up a tool record in DevOps, your admin configures the DevOps CreateDevOpsTool connection and credential alias to allow access to the tools in your environment.

Starting with version 1.15, DevOps connection and credential aliases (DevOpsAlias1 through DevOpsAlias10) are used to connect automatically when you set up your DevOps planning, coding, and orchestration tools.


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.

Note: If additional sys_alias records for DevOps are needed, they must be created by an admin in the sn_devops application scope.

You can reuse a DevOpsAlias alias no longer in use by inactivating the HTTP connection, if needed.

Role required: sn_devops.admin

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: Starting with version 1.15, 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.

1. Navigate to 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: Starting with version 1.15, 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 Agile Development 2.0 tool record in DevOps

Create an Agile Development 2.0 tool record in DevOps to connect and discover application plans.

Role required: sn_devops.admin

Actions:

- **Connect** to Agile Development 2.0, get the webhook URL, and configure the webhook in Agile Development 2.0 when you submit a DevOps tool record.
- **Discover** plans.
- **Import** work item, plan version, and feature records.

1. Create a tool record in DevOps to automatically connect to Jira 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.

<table>
<thead>
<tr>
<th>Tool Integration</th>
<th>Agile Development 2.0</th>
</tr>
</thead>
<tbody>
<tr>
<td>Tool Username</td>
<td>Agile username</td>
</tr>
<tr>
<td>Tool Password / Access Token</td>
<td>Agile password or access token</td>
</tr>
</tbody>
</table>

**Configuration Item** and MID Server are optional. Select MID Server for tools on premise attached to a MID Server.

c) Click **Submit**.

The tool is automatically **Connected Successfully** using a connection alias, and HTTP tool connection (basic authentication credential).

**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.

2. Click **Discover** to discover existing application plans (projects associated with the tool).

Plan records are added to the Plans related list.

3. Automatically configure the webhook URL in Agile Development 2.0.
   a) Click **Configure**.
   b) Enter your **Integration user** (DevOps Integration User) and **Password**.

The DevOps Integration User is provided with the DevOps application and is used for inbound authentication from your tool to your ServiceNow instance.

**Note:** You must configure the password before the DevOps Integration User can configure a tool.

Notifications are sent from the Agile Development 2.0 tool to DevOps as the DevOps Integration User.

4. Open the discovered application plan from the Plans related list and click the **Import** related link to import historical data from the application.

Imported work item records, plan version records, and feature records are added to the corresponding 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.

5. On the Apps related list, click New to create an App record to associate with the plan (project).

Note: A plan must have an associated App record to complete the planning tool setup.
Agile tool record in DevOps
Accelerating DevOps change

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.

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.

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 Closed with Close code as Successful on a successful job run, or Unsuccessful on error in the job run.

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

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**

Starting with version 1.14, 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**

Starting with version 1.14, when you configure change acceleration, you can select a custom template to populate fields automatically when creating the change request. In the ServiceNow DevOps step, set the change type as standard, normal, or emergency and select the custom template for the change.

The type of change request corresponds to the change request table in global scope.

**DevOps Model Change Request flow**

Starting with version 1.14, 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 mandatory.

**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.

**Test Results (build test summaries), starting with version 1.13**
List of build test summaries for the tests run after the associated package was created and linked.
If there is no package, or the change request is created before package creation, then the list of build test summaries includes tests that were run after any associated artifacts were created and linked.

If no package or artifact versions are linked to the pipeline execution, then the list is empty.

**Artifact Versions, starting with version 1.13**

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), starting with version 1.15**

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.**
DevOps change acceleration for releases

Starting with version 1.11, 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
- Commits and Work Items related lists

Artifact setup

1. (Optional) *Create an artifact tool record in DevOps.*

   ![Note](image)
   
   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](image)
   
   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:

  ```bash
  snDevOpsArtifact(artifactsPayload: ""{"artifacts": [{"name": "sa-web.jar", "version": "1.9", "semanticVersion": "1.9.0", "repositoryName": "services-1031"}]}"")
  ```

- Freestyle job (Register Artifact build step)

  For example:

  ```json
  {"artifacts": [{"name": "sentiment-analysis-web2", "version": "1.9", "semanticVersion": "1.9.0", "repositoryName": "maven-releases"}]
  ```

© 2020 ServiceNow, Inc. All rights reserved.
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 names, product names, and logos may be trademarks of the respective companies with which they are associated.
Artifact package creation

Configure artifact package creation in a scripted pipeline or freestyle job using the DevOps API /package/registration endpoint.

**Note:** Package name must be specified.

For Jenkins pipeline:

- Freestyle job (Create Package build step)
  
  For example:

  ```json
  {"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:

  ```groovy
  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"}]}"")
  ```
**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 and Nexus, for example.

*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

Number: CHG000020
Requested by: DevOps
Category: DevOps
Service: DevOps
Configuration item:
  - Priority: 4 - Low
  - Risk: Low
  - Impact: 3 - Low

Type: Normal
State: Review
On hold: Off
Conflict status: Not Run
Conflict last run: Not applicable
Assignment group: Change Management
Assigned to: Not applicable

Short description:
DevOps orchestrationTask: CorpSiteCDMProd, action: prod_deploy

Description:
DevOps orchestrationTask: CorpSiteCDMProd, action: prod_deploy

Justification:
This is an automated change request generated by the DevOps application.
<table>
<thead>
<tr>
<th>AFFECTED CIs</th>
<th>IMPACTED SERVICE/CIS</th>
<th>APPROVERS</th>
<th>CHANGE TASKS</th>
<th>PROBLEMS</th>
<th>INCIDENTS FIXED BY CHANGE</th>
<th>INCIDENTS CAUSED BY CHANGE</th>
<th>COMMITS (?)</th>
<th>WORK ITEMS (?)</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

**Commits**

<table>
<thead>
<tr>
<th>Number</th>
<th>Work Item</th>
<th>Commit ID</th>
<th>Committer email</th>
<th>Committed at</th>
<th>Additions</th>
</tr>
</thead>
<tbody>
<tr>
<td>COM0001528</td>
<td>Create job details page</td>
<td>48d6a2c9f578696c4f75c520a7f54c0f8</td>
<td><a href="mailto:usuario@example.com">usuario@example.com</a></td>
<td>2020-02-10 13:27:24</td>
<td>1</td>
</tr>
<tr>
<td>COM0001529</td>
<td>Create a menu item, app, search and ask...</td>
<td>651b099f588d646b8723b64bf386f73</td>
<td><a href="mailto:usuario@example.com">usuario@example.com</a></td>
<td>2020-02-10 13:28:05</td>
<td>1</td>
</tr>
<tr>
<td>COM0001530</td>
<td>Create a job search form...</td>
<td>1b930c9f5672057295a0cbb14avef0bb</td>
<td><a href="mailto:usuario@example.com">usuario@example.com</a></td>
<td>2020-02-10 13:31:35</td>
<td>1</td>
</tr>
</tbody>
</table>
### DevOps artifact version list

<table>
<thead>
<tr>
<th>Name</th>
<th>Version</th>
<th>Semantic version</th>
<th>Artifact</th>
<th>Built by</th>
<th>Built on</th>
</tr>
</thead>
<tbody>
<tr>
<td>globex-web.war</td>
<td>2.30.0</td>
<td>2.30.0</td>
<td>globex-web.war</td>
<td>T6961118</td>
<td>2020-02-20 13:32:31</td>
</tr>
<tr>
<td>globex-web.war</td>
<td>2.29.0</td>
<td>2.29.0</td>
<td>globex-web.war</td>
<td>T6961114</td>
<td>2020-02-20 13:29:50</td>
</tr>
<tr>
<td>globex-web.war</td>
<td>2.28.0</td>
<td>2.28.0</td>
<td>globex-web.war</td>
<td>T6961112</td>
<td>2020-02-20 13:28:20</td>
</tr>
<tr>
<td>globex-web.war</td>
<td>2.27.0</td>
<td>2.27.0</td>
<td>globex-web.war</td>
<td>T6961108</td>
<td>2020-02-20 13:27:50</td>
</tr>
<tr>
<td>globex-web.war</td>
<td>2.26.0</td>
<td>2.26.0</td>
<td>globex-web.war</td>
<td>T6961106</td>
<td>2020-02-20 13:14:51</td>
</tr>
<tr>
<td>globex-web.war</td>
<td>2.25.0</td>
<td>2.25.0</td>
<td>globex-web.war</td>
<td>T6961104</td>
<td>2020-02-20 13:04:20</td>
</tr>
</tbody>
</table>
DevOps artifact version – commits
DevOps artifact version – packages

DevOps package
How commits are determined for a release in DevOps

Starting with version 1.11, 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 hotfix 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:

- 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)

<table>
<thead>
<tr>
<th>Commit</th>
<th>Build pipeline</th>
<th>Semantic version</th>
<th>Included in package</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>A</td>
<td>1.0.0</td>
<td>X</td>
</tr>
<tr>
<td>2</td>
<td>A</td>
<td>1.0.1</td>
<td>--</td>
</tr>
<tr>
<td>3</td>
<td>A</td>
<td>1.1.0</td>
<td>X</td>
</tr>
</tbody>
</table>

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 hotfix.

### Package 2 (A 1.2.0, B 1.1.0)

<table>
<thead>
<tr>
<th>Commit</th>
<th>Build pipeline</th>
<th>Semantic version</th>
<th>Included in package</th>
</tr>
</thead>
<tbody>
<tr>
<td>4</td>
<td>A</td>
<td>1.1.1</td>
<td>--</td>
</tr>
<tr>
<td>5</td>
<td>A</td>
<td>1.2.0</td>
<td>X</td>
</tr>
<tr>
<td>Commit</td>
<td>Build pipeline</td>
<td>Semantic version</td>
<td>Included in package</td>
</tr>
<tr>
<td>-------</td>
<td>----------------</td>
<td>-----------------</td>
<td>------------------</td>
</tr>
<tr>
<td>6</td>
<td>A</td>
<td>1.2.0</td>
<td>X</td>
</tr>
<tr>
<td>7</td>
<td>B</td>
<td>1.0.0</td>
<td>X</td>
</tr>
<tr>
<td>8</td>
<td>B</td>
<td>1.0.0</td>
<td>X</td>
</tr>
<tr>
<td>9</td>
<td>B</td>
<td>1.1.0</td>
<td>X</td>
</tr>
<tr>
<td>10</td>
<td>B</td>
<td>1.1.0</td>
<td>X</td>
</tr>
</tbody>
</table>

**Note:** Commit 4 (build A, semantic version 1.1.1) is not included because the semantic version syntax indicates a patch or hotfix.

### Package 3 (B 1.2.0, C 1.0.0)

<table>
<thead>
<tr>
<th>Commit</th>
<th>Build pipeline</th>
<th>Semantic version</th>
<th>Included in package</th>
</tr>
</thead>
<tbody>
<tr>
<td>11</td>
<td>A</td>
<td>1.3.0</td>
<td>--</td>
</tr>
<tr>
<td>12</td>
<td>B</td>
<td>1.2.0</td>
<td>X</td>
</tr>
<tr>
<td>13</td>
<td>B</td>
<td>1.2.0</td>
<td>X</td>
</tr>
<tr>
<td>14</td>
<td>C</td>
<td>1.0.0</td>
<td>X</td>
</tr>
<tr>
<td>15</td>
<td>C</td>
<td>1.0.0</td>
<td>X</td>
</tr>
<tr>
<td>16</td>
<td>C</td>
<td>1.0.0</td>
<td>X</td>
</tr>
</tbody>
</table>

**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)

<table>
<thead>
<tr>
<th>Commit</th>
<th>Build pipeline</th>
<th>Semantic version</th>
<th>Included in package</th>
</tr>
</thead>
<tbody>
<tr>
<td>17</td>
<td>A</td>
<td>1.4.0</td>
<td>X</td>
</tr>
<tr>
<td>18</td>
<td>A</td>
<td>1.4.0</td>
<td>X</td>
</tr>
<tr>
<td>19</td>
<td>B</td>
<td>1.3.0</td>
<td>X</td>
</tr>
<tr>
<td>20</td>
<td>B</td>
<td>1.3.0</td>
<td>X</td>
</tr>
<tr>
<td>21</td>
<td>C</td>
<td>1.1.0</td>
<td>X</td>
</tr>
<tr>
<td>22</td>
<td>C</td>
<td>1.1.0</td>
<td>X</td>
</tr>
<tr>
<td>23</td>
<td>C</td>
<td>1.2.0</td>
<td>X</td>
</tr>
<tr>
<td>24</td>
<td>C</td>
<td>1.2.0</td>
<td>X</td>
</tr>
</tbody>
</table>

**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

Starting with version 1.11, 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.

Starting with version 1.15, DevOps connection and credential aliases (DevOpsAlias1 through DevOpsAlias10) are used to connect automatically when you set up your DevOps planning, coding, and orchestration tools.


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.

Role required: sn_devops.admin

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: Starting with version 1.15, 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.

1. Navigate to 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: Starting with version 1.15, 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.

Role required: sn_devops.admin

- **Note:** An artifact tool is not necessary unless a webhook or user-created integration subflow configuration is required to look up artifact versions.

1. Navigate to DevOps > Tools > Create New and create a record.
2. Enter a Name, Tool Integration, and Tool URL.

   **Configuration Item** is optional.

   - **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.

   After submitting, the artifact tool is automatically **Connected Successfully** using a connection alias, and HTTP tool connection (basic authentication credential).

DevOps artifact tool setup

Using DevOps Model Change Request flow

Starting with version 1.14, 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.
**DevOps Model Change Request Flow**

**TRIGGER**
- Change Request Created where (Category is DevOps)

**ACTIONS**
1. Look Up Step Execution Record where (Change request is (Trigger = Change Request Record))
2. Wait For Condition where (Approval is Rejected, or State is one of -1,4)
3. If (Trigger = Change Request Record = State) is Implement then
   - Update State of Step execution based on change approval
   - End
4. Else If (Trigger = Change Request Record = Approval) is Rejected then
   - Update State of Step execution based on change approval
   - End
5. Else If (Trigger = Change Request Record = State) is Canceled then
   - Update State of Step execution based on change approval
   - End
6. Select to add an Action, Flow Logic, or Subflow
Calling the Flow Designer action using a script

Method to call the Flow Designer action from a script:

```javascript
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#.

![Decision Table](image)
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.
DevOps dashboard and Pipeline UI

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

DevOps Insights dashboard

The DevOps Insights dashboard consists of a flexible graphical view that you can tailor to your needs, which is helpful for DevOps administrators when planning and providing updates.

Note: You must Install DevOps Insights to view the Insights Standard dashboard.

Use this dashboard to analyze operational and business reports and determine the overall efficiency and growth associated with your development processes.

- Monitor activities, KPIs, and metrics in a central dashboard.
- Track performance over time using ServiceNow Performance Analytics.
- Access breakdowns by pipeline, app, step, or team.

DevOps Pipeline UI view

The Pipeline UI view lets you visualize interactions and results across a pipeline execution. This graphical view shows pipeline step progression and other details, including change requests, step execution, artifact versions, test summaries, and commits.

DevOps Insights Standard dashboard

Use the DevOps Insights application with ServiceNow Performance Analytics to gain insight into your DevOps environment.

Note: Starting with version 1.14, unless these jobs were previously customized, you must select Active and set the Run As credentials to System Administrator to collect daily or historical Insights data:

- [DevOps] Daily Data Collection (Daily) (inactive by default)
  
  Note: This is a scheduled job (to be run on a regular basis) 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.

  Note: This job 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

![DevOps Insights Dashboard](image)

- **Total Changes Submitted - Yearly**: May 22 - 56
- **Avg Time to Close - Last 30 Days**: May 22 - 8 Hrs
- **Change Approval Rate - Last 30 Days**: May 22 - 96.88%
- **Non-DevOps Change Approval Rate - Last 30 Days**: May 22 - 21.74%

### Change Request Volume

- **DevOps**:
  - Apr 22: 5
  - May 16: 25
  - May 18: 45

### Pending Change per Pipeline

- **Name**:
  - Sadas
  - Work Pipeline
  - Non Changes

### Avg Time to Close Changes

- **Apr 27**: 250
- **May 4**: 350
- **May 11**: 450
- **May 18**: 500

*Average time for Closed DevOps Changes*
### Change Acceleration

<table>
<thead>
<tr>
<th>Report</th>
<th>Source list</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>Total Changes Submitted - Yearly</td>
<td>Change Request</td>
<td>Total DevOps changes submitted yearly.</td>
</tr>
<tr>
<td>Avg Time to Close - Last 30 days</td>
<td>Change Request</td>
<td>Average time to close DevOps changes in the last 30 days.</td>
</tr>
</tbody>
</table>

**Note:** This widget uses average aggregation and does not support multi-element selection.

| Change Approval Rate - Last 30 days         | Change Request | DevOps average change success rate for change requests in the last 30 days: 

\[
\left( \frac{[[\text{DevOps Change Success}]]}{[[\text{DevOps Change}]]} \right) \times 100
\]

**Note:** This widget uses average aggregation and does not support multi-element selection.

| Non-DevOps Change Approval Rate - Last 30 days | Change Request | Non-DevOps change approval rate for change requests in the last 30 days: 

\[
\left( \frac{[[\text{Non-DevOps Change Approval}]]}{[[\text{Non-DevOps Change}]]} \right) \times 100
\]

**Note:** Filter is not applicable to this widget.

| Change Request Volume                       | Change Request | Volume of change requests created for DevOps in the last 7 days.  

Compare the number of change requests created after you transition to DevOps so you can see the advantage of running DevOps in your environment. |

| Pending Changes per Pipeline                | Step Execution | Number of change requests that have not been closed for each pipeline.  

See the blockages in each pipeline that are keeping the change request from being completed so you can investigate the cause. |
### Average Time to Close Changes

<table>
<thead>
<tr>
<th>Source list</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>Change Request</td>
<td>Average time to close DevOps changes by app. Compare DevOps change request statistics with non-DevOps change requests to see that DevOps change requests are getting resolved faster.</td>
</tr>
</tbody>
</table>

### Changes Awaiting Approval

<table>
<thead>
<tr>
<th>Source list</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>Change Request</td>
<td>Number of DevOps changes awaiting approval by date range. Compare DevOps change request statistics with non-DevOps change requests to see that DevOps change requests are getting resolved faster.</td>
</tr>
</tbody>
</table>

### Non-DevOps Changes Awaiting Approval

<table>
<thead>
<tr>
<th>Source list</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>Change Request</td>
<td>Number of Non-DevOps changes awaiting approval by date range. Compare DevOps change request statistics with non-DevOps change requests to see that DevOps change requests are getting resolved faster.</td>
</tr>
</tbody>
</table>

### Accelerate Metrics

Starting with version 1.15, the Accelerate Metrics tab shows deployment frequency, lead time, MTTR, and change failure rate info.
## Accelerate Metrics

<table>
<thead>
<tr>
<th>Report</th>
<th>Source list</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>Deployment Frequency - Monthly</td>
<td>Step Execution</td>
<td>Number of successful production deployments in a month. Applies to steps of type <strong>Prod Deploy</strong> that are in completed state.</td>
</tr>
</tbody>
</table>
| Average Lead Time              | Pipeline Execution                | Average of:                                                                                                                                     
<p>|                                |                                  | ([Time the code is successfully pushed to production] - [Earliest commit time]) Applies to steps of type <strong>Prod Deploy</strong> that are in completed state.                                                |
| 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.                                                                                                                        |
| Change Failure Rate - Monthly  | Change Request                   | Average change failure rate in a month.                                                                                                                                                                   |
| Deployment Frequency           | Step Execution                   | Number of successful production deployments in the last 30 days. Applies to steps of type <strong>Prod Deploy</strong> that are in completed state.                                                                    |
| Change Failure Rate            | Change Request                   | Average change failure rate in the last 30 days.                                                                                                                                                           |
| 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.                                                                                                                                    |</p>
<table>
<thead>
<tr>
<th>Report</th>
<th>Source list</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>Lead Time</td>
<td>Pipeline Execution</td>
<td>([Time the code is successfully pushed to production] - [Earliest commit time]) Applies to steps of type <strong>Prod Deploy</strong> that are in completed state.</td>
</tr>
</tbody>
</table>

**Operational Stability**

Starting with version 1.14, 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

**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.

**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

<table>
<thead>
<tr>
<th>Report</th>
<th>Source list</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>Mean Time to Resolve - Last 30 days</td>
<td>Database view joined by Incident, Change Request, Step Execution, Step, Pipeline, and App lists.</td>
<td>Average resolve time for an incident caused by a DevOps change in the last 30 days.</td>
</tr>
<tr>
<td></td>
<td></td>
<td><strong>Note:</strong> This widget uses average aggregation and does not support multi-element selection.</td>
</tr>
<tr>
<td>Incidents - Monthly</td>
<td>Incident</td>
<td>Number of incidents in a month (based on pipeline steps of type <strong>Prod Deploy</strong>) linked to business service in CMDB. This report provides an indication of environment stability.</td>
</tr>
<tr>
<td>Average Service Availability - Last 30 days</td>
<td>Database view joined by Service Availability, Service Offering, Business Service, Step, Pipeline, and App lists.</td>
<td>Average service availability in the last 30 days (based on pipeline steps of type <strong>Prod Deploy</strong>) linked to application service in CMDB. This report provides an indication of environment stability.</td>
</tr>
<tr>
<td>Outages - Monthly</td>
<td>Outage</td>
<td>Number of outages in a month (based on pipeline steps of type <strong>Prod Deploy</strong>) linked to business service in CMDB. This report provides an indication of environment stability.</td>
</tr>
<tr>
<td>Mean Time to Resolve Trend</td>
<td>Database view joined by Incident, Change Request, Step Execution, Step, Pipeline, and App lists.</td>
<td>Daily average resolve time for an incident caused by a DevOps change.</td>
</tr>
<tr>
<td>Service Availability Trend</td>
<td>Database view joined by Service Availability, Service Offering, Business Service, Step, Pipeline, and App lists.</td>
<td>Daily average service availability (based on pipeline steps of type <strong>Prod Deploy</strong>) linked to application service in CMDB. This report provides an indication of environment stability.</td>
</tr>
<tr>
<td>Incidents Trend</td>
<td>Incident</td>
<td>Daily number of incidents (based on pipeline steps of type <strong>Prod Deploy</strong>) linked to business service in CMDB. This report provides an indication of environment stability.</td>
</tr>
<tr>
<td>Report</td>
<td>Source list</td>
<td>Description</td>
</tr>
<tr>
<td>-----------------</td>
<td>-------------</td>
<td>-----------------------------------------------------------------------------</td>
</tr>
<tr>
<td>Outage Trend</td>
<td>Outage</td>
<td>Daily number of outages (based on pipeline steps of type <code>Prod Deploy</code>) linked to business service in CMDB. This report provides an indication of environment stability.</td>
</tr>
</tbody>
</table>
Development

![DevOps Insights](image)

- **Commit Frequency**
  - Count
  - Number of Commits
  - Frequency from Mar '20 to May '20

- **Average Branches per Repository**
  - Average branches per repository
  - May 20
  - 0.25

- **Average Commits per Pipeline Execution**
  - Average commits per pipeline execution
  - May 20
  - 1.09

- **Commits without Work Item**
  - Starred commits
  - Name: 47263869+rsanes+noreply.github.com
  - Name: rphan rsanes@service.com
  - Name: immanuelbeal@gmail.com
  - Name: 44116298+noreply@users.noreply.github.com

- **Work Items**
  - Completed and In Progress
  - Timeline from April 27 to May 18
## Development

<table>
<thead>
<tr>
<th>Report</th>
<th>Source list</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>Commit Frequency</td>
<td>Commit</td>
<td>Number of commits measured daily. Smaller more frequent commits are preferred over larger less frequent ones.</td>
</tr>
<tr>
<td>Average Branches per Repository</td>
<td>Branches, Repository</td>
<td>Average branches per repository on a given day.</td>
</tr>
<tr>
<td>Note: This widget uses formula and does not support multi-element selection.</td>
<td></td>
<td></td>
</tr>
</tbody>
</table>
| Average Commits per Pipeline Execution | Commit, Pipeline Execution | Average commits per pipeline on in the last 30 days: \[
\frac{[\text{Total number of commits}]}{[\text{Number of pipeline executions}]}\]  

A low number is preferable, which indicates a concentrated effort, versus switching from task to task without completion.  

Note: This widget uses average aggregation and does not support multi-element selection. |
| Commits without Work Item  | Commit                | Commits made that are not tied to a work item, grouped by committer, in the last 30 days. This report is useful for investigating and resolving why a commit is not tied to a work item, since all commits should be tied to a work item. |
| Work Items                 | Work Item             | Number of work items that are complete or working in progress in the last 30 days.  

Note: Filter is not applicable to this widget. |
## Commit Insights

### Active Committers
- **May 22**: 4
- **%**: 4% (May 22: 4)

### Average Commits per Committer
- **May 22**: 28.00

### Average Files Added per Commit
- **May 22**: 0.00

### % Commits Rewritten
- **May 22**: 0.00%

### Top Committers
<table>
<thead>
<tr>
<th>Name</th>
<th>May 22</th>
<th>Trend</th>
</tr>
</thead>
<tbody>
<tr>
<td><a href="mailto:example@example.com">example@example.com</a></td>
<td>15</td>
<td></td>
</tr>
<tr>
<td><a href="mailto:example@example.com">example@example.com</a></td>
<td>29</td>
<td></td>
</tr>
<tr>
<td><a href="mailto:example@example.com">example@example.com</a></td>
<td>25</td>
<td></td>
</tr>
<tr>
<td><a href="mailto:example@example.com">example@example.com</a></td>
<td>3</td>
<td></td>
</tr>
<tr>
<td><a href="mailto:example@example.com">example@example.com</a></td>
<td>9</td>
<td></td>
</tr>
</tbody>
</table>

### Top Reverters
<table>
<thead>
<tr>
<th>Name</th>
<th>May 22</th>
<th>Trend</th>
</tr>
</thead>
<tbody>
<tr>
<td><a href="mailto:example@example.com">example@example.com</a></td>
<td>0</td>
<td></td>
</tr>
<tr>
<td><a href="mailto:example@example.com">example@example.com</a></td>
<td>0</td>
<td></td>
</tr>
<tr>
<td><a href="mailto:example@example.com">example@example.com</a></td>
<td>0</td>
<td></td>
</tr>
<tr>
<td><a href="mailto:example@example.com">example@example.com</a></td>
<td>0</td>
<td></td>
</tr>
<tr>
<td><a href="mailto:example@example.com">example@example.com</a></td>
<td>0</td>
<td></td>
</tr>
</tbody>
</table>

### Comments Added per App
<table>
<thead>
<tr>
<th>Name</th>
<th>May 22</th>
<th>Change</th>
<th>Change %</th>
<th>Trend</th>
</tr>
</thead>
<tbody>
<tr>
<td><a href="mailto:example@example.com">example@example.com</a></td>
<td>63</td>
<td>0</td>
<td>0.0%</td>
<td></td>
</tr>
<tr>
<td><a href="mailto:example@example.com">example@example.com</a></td>
<td>25</td>
<td>0</td>
<td>-4.5%</td>
<td></td>
</tr>
<tr>
<td><a href="mailto:example@example.com">example@example.com</a></td>
<td>10</td>
<td>0</td>
<td>0.0%</td>
<td></td>
</tr>
</tbody>
</table>

---

© 2020 ServiceNow, Inc. All rights reserved. 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 names, product names, and logos may be trademarks of the respective companies with which they are associated.
## Commit Insights

<table>
<thead>
<tr>
<th>Report</th>
<th>Source list</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>Active Committers</td>
<td>Commit</td>
<td>Committers that submitted commits in the last 30 days. Shows how many active committers there are.</td>
</tr>
<tr>
<td></td>
<td></td>
<td><strong>Note:</strong> This widget uses count distinct aggregation and does not support multi-element selection.</td>
</tr>
<tr>
<td>Average Commits per Committer</td>
<td>Commit</td>
<td>Total number of commits in the last 30 days / Active committers. Shows how often committers are committing. A higher value is more favorable.</td>
</tr>
<tr>
<td></td>
<td></td>
<td><strong>Note:</strong> This widget uses formula and does not support multi-element selection.</td>
</tr>
<tr>
<td>Average Files Added per Commit</td>
<td>Commit, Commit Details</td>
<td>Total number of files added in the last 30 days / Total number of commits in the last 30 days. Shows how few files are committed at a time. A lower value is more favorable.</td>
</tr>
<tr>
<td></td>
<td></td>
<td><strong>Note:</strong> This widget uses formula and does not support multi-element selection.</td>
</tr>
<tr>
<td>% Commits Reverted</td>
<td>Commit</td>
<td>Commits reverted in the last 30 days / Total number of commits in the last 30 days. Shows how many commits have been reverted. A lower value is more favorable.</td>
</tr>
<tr>
<td></td>
<td></td>
<td><strong>Note:</strong> This widget uses formula and does not support multi-element selection.</td>
</tr>
<tr>
<td>Top Committers</td>
<td>Commit</td>
<td>Committers with the highest number of commits in the last 30 days. Provides visibility into the users that commit the most.</td>
</tr>
<tr>
<td>Report</td>
<td>Source list</td>
<td>Description</td>
</tr>
<tr>
<td>------------------------</td>
<td>------------------------------</td>
<td>-----------------------------------------------------------------------------</td>
</tr>
<tr>
<td>Top Reverters</td>
<td>Commit</td>
<td>Committers with the highest number of reverts in the last 30 days.</td>
</tr>
<tr>
<td></td>
<td></td>
<td>Provides visibility into the users that revert commits the most.</td>
</tr>
<tr>
<td>Commits Added per App</td>
<td>Commit, Commit Details</td>
<td>Number of commits added per app in the last 30 days.</td>
</tr>
<tr>
<td></td>
<td></td>
<td>Provides visibility into the development activity for each app.</td>
</tr>
<tr>
<td></td>
<td></td>
<td>Note: Filter is not applicable to this widget.</td>
</tr>
</tbody>
</table>
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

<table>
<thead>
<tr>
<th>Report</th>
<th>Source list</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>Deployment Frequency - Monthly</td>
<td>Step Execution</td>
<td>Number of successful production deployments in a month. Applies to steps of type <strong>Prod Deploy</strong> that are in completed state.</td>
</tr>
<tr>
<td>Failed Deployments</td>
<td>Step Execution</td>
<td>Number of failed production deployments in the last 30 days. Applies to steps of type <strong>Prod Deploy</strong> that are in failed or user-canceled state.</td>
</tr>
<tr>
<td>Deployment Success Rate</td>
<td>Step Execution</td>
<td>Deployments success rate over the last 30 days. Deployment Success Rate = (Number of Successful Deployments in the last 30 days / Total Number of Deployments in the last 30 days) * 100. Applies to steps of type <strong>Prod Deploy</strong> in completed state.</td>
</tr>
<tr>
<td>Average Lead Time</td>
<td>Pipeline Execution</td>
<td>Average of: ([Time the code is successfully pushed to production] - [Earliest commit time]) Applies to steps of type <strong>Prod Deploy</strong> that are in completed state.</td>
</tr>
<tr>
<td>Successful Production Deployments</td>
<td>Step Execution</td>
<td>Frequency of successful production deployments over time broken down by app. More frequent production deployments are preferred.</td>
</tr>
<tr>
<td>Report</td>
<td>Source list</td>
<td>Description</td>
</tr>
<tr>
<td>-------------------------------</td>
<td>-----------------</td>
<td>-----------------------------------------------------------------------------</td>
</tr>
<tr>
<td>Failed Production Deployments</td>
<td>Step Execution</td>
<td>Frequency of failed production deployments over time broken down by app.</td>
</tr>
<tr>
<td>Commit-to-Deploy Lead Time</td>
<td>Pipeline Execution</td>
<td>Duration from the earliest commit time to production deployment (for a successful pipeline execution). Minimizing the time it takes from committing code to successfully running it in production is preferable. 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.</td>
</tr>
</tbody>
</table>
System Health

Task Execution Success Rate

70.87%

0.56 (-0.8%) May 21: 71.45%

Number of Task Executions

Number of API calls

Processed  Not Processed
System Health

<table>
<thead>
<tr>
<th>Report</th>
<th>Source list</th>
<th>Description</th>
</tr>
</thead>
</table>
| Task Execution Success Rate | Task Executions | Success rate for tasks run by the execution tools over time: \[
\frac{[\text{Task Execution Success}]}{[\text{Task Execution}]} \times 100
\]

**Note:** This widget does not support multi-element selection.

<table>
<thead>
<tr>
<th>Number of Task Executions</th>
<th>Task Executions</th>
<th>Default number of tasks executions in the last 30 days.</th>
</tr>
</thead>
<tbody>
<tr>
<td>Number of API calls</td>
<td>Event</td>
<td>Default number of API calls in the last 30 days.</td>
</tr>
</tbody>
</table>

**Note:** Filter is not applicable to this widget.

**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.

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

**Note:** You must reload the view to update the status buttons in the pipeline execution History.

**Step run states**

<table>
<thead>
<tr>
<th>State</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>Green</td>
<td>Successful step run.</td>
</tr>
<tr>
<td>Grey</td>
<td>Step not run.</td>
</tr>
<tr>
<td>Yellow</td>
<td>Step is running.</td>
</tr>
<tr>
<td>Red</td>
<td>Step run failed.</td>
</tr>
</tbody>
</table>

**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.
Starting with version 1.18, the order the cards appear in the Pipeline UI is by task execution.

<table>
<thead>
<tr>
<th>UI feature</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>Pipeline steps</td>
<td>View pipeline execution:</td>
</tr>
<tr>
<td></td>
<td>• Timing:timing for each step, and wait times in between.</td>
</tr>
<tr>
<td></td>
<td>• State:Color coded step run states.</td>
</tr>
<tr>
<td>View change request</td>
<td>Change request record.</td>
</tr>
<tr>
<td></td>
<td>Click directly into the change request of the step that was created by DevOps to view details of the change and take action.</td>
</tr>
<tr>
<td>UI feature</td>
<td>Description</td>
</tr>
<tr>
<td>------------</td>
<td>-------------</td>
</tr>
<tr>
<td>Pipeline history</td>
<td>Pipeline Execution. Click a history tile to view the previous step details for a pipeline execution.</td>
</tr>
<tr>
<td>Artifacts</td>
<td>Artifact versions.</td>
</tr>
<tr>
<td>Quality</td>
<td>Build Test Summary. View the build test results to see what tests passed or failed. Starting with version 1.15, the quality card contains test summaries:</td>
</tr>
<tr>
<td></td>
<td>• Test type and test category in the format: test type/test category</td>
</tr>
<tr>
<td></td>
<td>• Native ID of the step</td>
</tr>
<tr>
<td></td>
<td>• Test pass percentage (unit and functional tests only)</td>
</tr>
<tr>
<td></td>
<td>• Throughput (performance tests only)</td>
</tr>
<tr>
<td></td>
<td>• Step name</td>
</tr>
</tbody>
</table>

Starting with version 1.16, click directly into DevOps change requests, step executions, artifacts, artifact versions, work items, and test summaries in flyout windows.
User-created DevOps integrations

User-created integrations are for integrating additional planning, coding, orchestration, and test tools not included in the integrations provided with the DevOps application.

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

**Note:** User-created integrations for test tools are not supported.

Tool integrations provided with the DevOps application:

- Planning (Azure DevOps Boards, Jira ver 7.x and 8.x, ServiceNow Agile Development ver 2.x)
- Coding (Azure DevOps Repos, Bitbucket Server/Enterprise ver 5.10.0, GitHub ver 2.x, GitHub Enterprise ver 2.x, GitLab SCM ver 12.8.0)
- Orchestration and Test (Azure DevOps Pipelines, Jenkins ver 2.138, GitLab CI/CD ver 12.8.0 basic CI pipelines)

DevOps integration objects

Starting with version 1.12, DevOps tool integration consists of these objects.

A tool can be mapped to multiple capabilities.
<table>
<thead>
<tr>
<th>Tool Integration</th>
<th>Tool Type Capability</th>
<th>Tool Capability Mapping</th>
</tr>
</thead>
</table>
| • Agile Development 2.0  
• Azure DevOps  
• Bitbucket  
• GitHub  
• GitHub Enterprise  
• GitLab  
• Jenkins  
• Jira | • Plan  
• Code  
• Orchestration  
• Artifact  
• Test | Plan  
• Agile Development 2.0 - Plan  
• Azure DevOps - Plan  
• Jira - Plan |
| | | Code  
• Azure DevOps - Code  
• Bitbucket - Code  
• GitHub - Code  
• GitHub Enterprise - Code  
• GitLab - Code |
| | | Orchestration  
• Azure DevOps - Orchestration  
• Jenkins - Orchestration  
• GitLab - Orchestration |
| | | Test  
• Azure DevOps - Test  
• Jenkins - Test |

A tool capability mapping can be mapped to multiple actions.
<table>
<thead>
<tr>
<th>Tool Capability Mapping</th>
<th>Tool Action</th>
<th>Integration Capability</th>
</tr>
</thead>
<tbody>
<tr>
<td>Agile Development 2.0</td>
<td>Connect</td>
<td>Agile Development 2.0 - Plan - Connect</td>
</tr>
<tr>
<td></td>
<td>Discover</td>
<td>Agile Development 2.0 - Plan - Discover</td>
</tr>
<tr>
<td></td>
<td>Import</td>
<td>Agile Development 2.0 - Plan - Import</td>
</tr>
<tr>
<td></td>
<td>Lookup</td>
<td>Agile Development 2.0 - Plan - Notification</td>
</tr>
<tr>
<td>Azure DevOps - Plan</td>
<td>Connect</td>
<td>Azure DevOps - Plan - Connect</td>
</tr>
<tr>
<td>Azure DevOps - Code</td>
<td>Discover</td>
<td>Azure DevOps - Plan - Discover</td>
</tr>
<tr>
<td>Azure DevOps - Orchestration</td>
<td>Import</td>
<td>Azure DevOps - Plan - Notification</td>
</tr>
<tr>
<td>Bitbucket - Code</td>
<td>Lookup</td>
<td>Azure DevOps - Code - Discover</td>
</tr>
<tr>
<td>GitHub - Code</td>
<td>Notification</td>
<td>Azure DevOps - Code - Notification</td>
</tr>
<tr>
<td>GitHub Enterprise - Code</td>
<td></td>
<td>Azure DevOps - Orchestration - Discover</td>
</tr>
<tr>
<td>GitLab - Code</td>
<td></td>
<td>Azure DevOps - Orchestration - Notification</td>
</tr>
<tr>
<td>GitLab - Orchestration</td>
<td></td>
<td></td>
</tr>
<tr>
<td>Jenkins - Orchestration</td>
<td></td>
<td></td>
</tr>
<tr>
<td>Jira - Plan</td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

© 2020 ServiceNow, Inc. All rights reserved.
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 names, product names, and logos may be trademarks of the respective companies with which they are associated.
Test tools

Starting with version 1.15, these test types are supported.

- Unit (JUnit)
- Functional (Integration, Regression, Smoke, System, User Acceptance)
- Performance (Load)

Use the `DevOps - POST /devops/tool/{capability}` endpoint of the `DevOps API` to integrate your test tool.

**Test tool mappings**

<table>
<thead>
<tr>
<th>Tool Integration</th>
<th>Test Type</th>
<th>Test Type Mapping</th>
</tr>
</thead>
<tbody>
<tr>
<td>• Azure DevOps</td>
<td>Unit:</td>
<td>• Azure DevOps - JUnit</td>
</tr>
<tr>
<td>• Jenkins</td>
<td>• JUnit</td>
<td>• Jenkins - JUnit</td>
</tr>
<tr>
<td></td>
<td>Functional:</td>
<td></td>
</tr>
<tr>
<td></td>
<td>• Integration</td>
<td></td>
</tr>
<tr>
<td></td>
<td>• Regression</td>
<td></td>
</tr>
<tr>
<td></td>
<td>• Smoke</td>
<td></td>
</tr>
<tr>
<td></td>
<td>• System</td>
<td></td>
</tr>
<tr>
<td></td>
<td>• User Acceptance</td>
<td></td>
</tr>
<tr>
<td></td>
<td>Performance:</td>
<td></td>
</tr>
<tr>
<td></td>
<td>• Load</td>
<td></td>
</tr>
</tbody>
</table>

The default test type is JUnit. You can change the default test type by modifying the `[sn_devops.default_test_type] DevOps property.`

View summary results:

- **DevOps > Test Results** module.
- **DevOps change request** - Test Results related list.
- **DevOps Pipeline UI** - Quality tile.

**Tool capability actions**

- Starting with version 1.9, **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.

- Starting with version 1.9, **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 (planning, coding, or orchestration).
See Discover capability subflow for more details.

Note: Pagination is not supported in the Discover action for user-created integrations.

- Starting with version 1.12, Import action:
  Import action does not support historical import functionality.

- Starting with version 1.12, Lookup action:
  Lookup master 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: Starting with version 1.11, 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 an API manager in DevOps for the tool integration.
- 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.

Note: Notifications (webhook) capability is supported. Starting with version 1.9, 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 orchestration) tool record in DevOps to connect to the tool you are integrating (source tool).

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 event**

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**

<table>
<thead>
<tr>
<th>Error</th>
<th>Action</th>
</tr>
</thead>
<tbody>
<tr>
<td>Missing required fields</td>
<td>The transformed payload does not match the standard payload.</td>
</tr>
<tr>
<td></td>
<td>Refer to the standard and JSON payloads provided.</td>
</tr>
<tr>
<td>Repository not marked for tracking</td>
<td>The commit cannot be inserted.</td>
</tr>
<tr>
<td></td>
<td>The DevOps admin needs to track the repository.</td>
</tr>
<tr>
<td>[Subflow] has not been published within application scope</td>
<td>The subflow is created but not published yet.</td>
</tr>
<tr>
<td>[app_scope]</td>
<td></td>
</tr>
<tr>
<td>Timeout exception</td>
<td>The subflow takes more time than the value set in the property: <strong>com.glide.hub.flow_api.default_execution_time</strong></td>
</tr>
<tr>
<td></td>
<td>See <strong>FlowAPI - executeSubflowQuick(String name, Map inputs, Number timeout)</strong> for more details.</td>
</tr>
<tr>
<td>Did not find a matching subflow for notification capability</td>
<td>The flow was not able to find the matching subflow.</td>
</tr>
<tr>
<td>and [tool_integration_sys_id] tool integration</td>
<td>Verify the integration setup procedure.</td>
</tr>
</tbody>
</table>

**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.

**DevOps orchestration tool integration setup**

Set up your orchestration tool integration with ServiceNow DevOps to sync important events like the start of a job, and the result of a job execution.

The integration also plays an important role in blocking the job from execution until an associated change request is approved, thereby enabling customers to gate an execution.

Release versions:

- DevOps application (version 1.13 and later)
• DevOps Integrations application (version 1.13 and later)

Prerequisites:
Prior to setting up a DevOps orchestration tool integration, you should be comfortable with these actions.
• Switching scopes in the application.
• Developing JavaScript within the ServiceNow instance.
• Verifying your development instance has the DevOps plugin installed.
• Creating subflows using the ServiceNow platform Flow Designer.
• Creating or editing ServiceNow instance components like script includes, and extension points.

Orchestration tool concepts

The ServiceNow DevOps orchestration tool is comprised of these components:

App

An App is an application or a service that is being built by the teams. It could be a microservice, like the Account Service, or it could be a monolith app that is being developed. An App is used to show the vital stats in the DevOps Insights dashboards.

Tool integration

This record holds the integration information for a tool and is required before a Tool record can be created. Integrators generally package this field as a part of the scope of the integration.
Tool integration capability

This table defines the capabilities that a tool can support. Some tools like Jenkins have just one capability whereas other tools like Azure DevOps support multiple capabilities. These capabilities are mapped in the Tool Integration Capability Mappings table.
Tool actions

Tool actions are the interactions with the source tool and the ServiceNow instance.

These capabilities are supported:

- Connect
- Discover
- Import
- Notifications

Integration capability

This table links the subflow built by the Integrator with a capability mapping, a tool integration, and an action.

This information is used by the DevOps user-created integrations framework to pick the subflows for performing a particular action.
Orchestration tool

An orchestration tool tracks and controls the flow of execution for a pipeline. Some popular orchestration tools supported by ServiceNow DevOps are Jenkins and Azure DevOps.

This tool supports a set of actions and it is the job of the Integrator to implement subflows for these actions to work. An orchestration tool needs to be connected and should be mapped to an app to get value out of the orchestration data.

Pipeline

This table holds all the static information about the pipeline that is set up in the source orchestration tool, like Jenkins or Azure DevOps.

The **Orchestration Pipeline** field must be the same as the name of the pipeline that was created in the source tool. The name should be prefixed by a path if the source tool’s pipeline is under a folder.

For example, if the Jenkins job **Unreal Deployments** is under the folder **/Minecraft/zone**, then the **Orchestration Pipeline** field is **/Minecraft/zone/Unreal Deployments**.

Steps

Steps define the stage of a respective pipeline. The **Type** field of the step categorizes the step as either build, test, deploy, or a combination thereof. The **Orchestration stage** field should match the name of the stage set in the source tool.

Steps also hold some static change request details that required if the steps need to be held until a change request is approved. This can be achieved by selecting the **Change control** check box.

Change control

A Change Control record is created for the step when the **Change control** check box is selected, and when the source tool must hold the step from executing until the change request is approved.

This record holds the callback details of the source tool which are used to notify the change request updates to the source tool. The mechanics of this interaction are a key part of the integration implementation for orchestration tools.
High-level objects
Onboarding process

Onboarding an orchestration tool is divided into these parts.

1. **Create the tool integration in the ServiceNow instance.**
2. **Create the tool in the ServiceNow instance.**
3. **Enable the source tool to call ServiceNow APIs.**
4. **Connect the tool and do post setup activities on the ServiceNow instance.**

Create the tool integration in the ServiceNow instance

DevOps tool integration includes setting up capabilities, mappings, and actions. Set up your tool integration and then create your tool record.

Role required: sn_devops.admin

1. Install the DevOps app from the ServiceNow Store using the link provided.
2. As an Integrator, create the Tool Integration record in the development instance.

3. Change the application scope to **DevOps Integration** and fill in these fields.
   - Set the **Tool Label** field to a readable value.
   - Set the **Integration version** to 1 to start with. Use a new version if you plan to add more features for that tool integration.

4. Create the API manager and set it in the Tool Integration list.
   a) Navigate to **System Extension Points > Scripted Extension Points**, search and open the **DevOpsOrchestrationToolIntegrationHandler** scripted extension point.
   b) Click **Create Implementation** and fill in the fields.

<table>
<thead>
<tr>
<th>Name</th>
<th>Name of the extension instance using the convention:</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>DevOps&lt;ToolName&gt;IntegrationHandler</td>
</tr>
</tbody>
</table>

| Accessible from | All application scopes. |
c) Implement the mandatory functions in the extension point.

**Note:** These mandatory functions must be implemented regardless of how the notification subflow is set up.

You can use the existing implementations for GitLab, Jenkins, and Azure DevOps as references when implementing these methods.
d) Set the API manager field in the Tool Integrations list.
5. Create a subflow using Flow Designer for an action.
6. Create a Tool Capability Mapping record to map a tool integration with the defined capabilities.
   This mapping adds the capabilities to the tool when it is created and linked to the Tool Integration record.
7. Create an Integration Capability record and map the action, the subflow that was created, and the Tool Capability Mapping record.
8. Link the integration capability with the tool integration.
9. Promote the set of records to the production instance.
10. To add more action support, create another subflow using Flow Designer for the action, and repeat the remaining steps.

Create the tool in the ServiceNow instance

Create a DevOps orchestration tool record to connect to your orchestration tool and generate a webhook URL.
Role required: sn_devops.admin

1. Configure the DevOps CreateDevOpsTool connection and credential alias to allow access to the tools in your environment.
   DevOps connection and credential aliases (DevOpsAlias1 through DevOpsAlias10) are used to connect automatically when you set up your DevOps tools.
   a) Navigate to Connections and Credentials > Connection & Credential Aliases and open the CreateDevOpsTool record.
   b) In the Connections related list, create a record and enter a Name for the connection.
   c) On the Connection form, click the Credential field lookup list, and then click New to create an admin credential.
   d) Click Basic Auth Credentials, enter a Name, and enter admin (or connection_admin) username and password.
   e) On the Connection form, enter https://<instance name>.service-now.com/api/now/table for the Connection URL.

2. Create a DevOps orchestration tool record to connect to your orchestration tool.
   a) Search for DevOps in the left navigation bar, and select DevOps > Tools > Create New.
b) Enter a **Tool Name** and fill in the orchestration tool details.

<table>
<thead>
<tr>
<th>Tool Integration</th>
<th>Source tool.</th>
</tr>
</thead>
<tbody>
<tr>
<td>Tool URL</td>
<td>Source tool URL.</td>
</tr>
<tr>
<td>Tool Username</td>
<td>Source tool username and access token.</td>
</tr>
<tr>
<td>Tool Password / Access Token</td>
<td></td>
</tr>
</tbody>
</table>

After submitting, the tool is automatically **Connected Successfully** using a connection alias, and HTTP tool connection (basic authentication credential).

3. Once the tool is created, you should see the **Webhook URL** field on the Tool form, for example:

   https://<DevOps.integration.user>:<password>@<instance-name>.service-now.com/api/sn_devops/v1/devops/tool/orchestration?toolId=3773bee8db14dc10b7fffb941d9619de

   This URL must be used by the source tool to publish tool events.

**Enable the source tool to call ServiceNow APIs**

Use ServiceNow APIs to pass the transactional data to the ServiceNow instance.

These APIs are required to be called by an orchestration tool. Refer to the API link in each endpoint for more details.

<table>
<thead>
<tr>
<th>Endpoint</th>
<th>Method</th>
<th>Resource</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>Step Mapping</td>
<td>GET</td>
<td>/devops/orchestration/stepMapping</td>
<td>Checks if there is an orchestration task in the ServiceNow instance.</td>
</tr>
<tr>
<td>Endpoint</td>
<td>Method</td>
<td>Resource</td>
<td>Description</td>
</tr>
<tr>
<td>-----------------</td>
<td>--------</td>
<td>-----------------------------------</td>
<td>-----------------------------------------------------------------------------</td>
</tr>
<tr>
<td>POST</td>
<td>/devops/orchestration/stepMapping</td>
<td>Creates an orchestration task and associates it to a step created in the ServiceNow instance for that particular pipeline.</td>
<td></td>
</tr>
<tr>
<td>GET</td>
<td>/devops/orchestration/changeControl</td>
<td>Used to check if the current job or orchestration task requires a change request.</td>
<td></td>
</tr>
<tr>
<td>POST</td>
<td>/devops/orchestration/changeControl</td>
<td>The source tool makes this call whenever it wants to trigger the change request creation flow. This API expects that the caller will pass a callback URL. On a successful response to this API, the source tool should block the execution of the job. The passed callback URL is used as a callback as soon as the change request has been either approved or denied. On approval, the orchestration tool can run the job. On rejection, the orchestration job should be canceled. The callback URL is thus the way to unblock a waiting job.</td>
<td></td>
</tr>
<tr>
<td>PUT</td>
<td>devops/orchestration/changeControl/{changeControlId}</td>
<td>The orchestration tool can use this API to notify the job status to the ServiceNow instance. This status is used to update the change request. This call is usually made when the change request is approved, and the job starts to execute. A typical use case is to update the change request actual start time, end time, or job status.</td>
<td></td>
</tr>
<tr>
<td>Endpoint</td>
<td>Method</td>
<td>Resource</td>
<td>Description</td>
</tr>
<tr>
<td>----------</td>
<td>--------</td>
<td>------------</td>
<td>----------------------------------------------------------------------------------------------------------------------------------------------</td>
</tr>
<tr>
<td>Webhook</td>
<td>POST</td>
<td>devops/orchestrate</td>
<td>Used to publish the start and end events of the job or the step. This API is an asynchronous API that creates an inbound event and returns the inbound event ID. The ServiceNow instance uses the Flow Designer platform solution to process the inbound event. Integration developers create the ServiceNow notification subflow as a part of the tool onboarding. This subflow is used to process the original payload and convert it into a normalized payload. Developers are not required to create the specific notification subflow if the tool can send a payload, as per the standard orchestration tool format. This handling is called user-created integrations. Refer to the <em>User-created DevOps integrations</em> section for more details.</td>
</tr>
</tbody>
</table>
pipeline {
    agent any
    tools {
        maven 'Maven'
    }
    stages {
        stage("build") {
            steps {
                echo "Building"
                sh 'mvn -X clean install -DskipTests'
            }
        }
        stage("test") {
            steps {
                echo "Testing"
                sh 'mvn test'
            }
            post {
                always {
                    junit "**/target/surefire-reports/**.xml"
                }
            }
        }
        stage("deploy") {
            steps {
                echo "deploy in prod"
            }
        }
    }
}

Sample Jenkinsfile pipeline
APIs added to the pipeline

Set up the source tool in the ServiceNow instance

Setting up the source tool in DevOps includes implementing actions, creating the pipeline and steps, and discovering orchestration tasks.
Role required: sn_devops.admin

1. Set up the orchestration tool in the ServiceNow instance.

As mentioned in the orchestration tool concepts section, orchestration tools require implementation of the action. For example, the Discover action needs to be implemented so the DevOps admin can fetch the metadata details of the pipelines and the job.

**Note:** The Notifications action is optional if the source tool can send the orchestration task event start and end notifications in the format that the ServiceNow product expects.

Refer to the standard orchestration payload section for more details. An integrator is required to create an integrations capability record with an empty Subflow name field.

Notification subflows are required if the orchestration tool cannot send the events in the standard payload form. The subflow is responsible for transforming the payload sent by the tool into the standard form, as mentioned in the standard orchestration payload section.

Refer to the User-created DevOps integrations framework section for more information on how to create all the actions.

2. Create a pipeline and steps.

a) Navigate to DevOps > Apps & Pipelines > Apps and open the application record to associate with the pipeline.

In the Pipelines related list, create a new pipeline and fill in the Orchestration pipeline field with the full project name as specified in the source tool and click Update.

b) Open the pipeline record again and in the Steps related list, create a DevOps step for each Jenkins pipeline stage so an orchestration task can be created.

Map each DevOps step to a source tool pipeline stage (Orchestration stage field). The Orchestration stage field value of each step is case-sensitive and must match the original name of the corresponding source tool pipeline stage.

<table>
<thead>
<tr>
<th>Name</th>
<th>Name of the pipeline step</th>
</tr>
</thead>
</table>
### Type

Pipeline step type.
- Build and Test
- Test
- Deploy
- Deploy and Test
- Manual
- Prod Deploy

### Order

Order in which the steps are run.

### Orchestration stage

Source tool pipeline stage name (case-sensitive).

---

3. Discover the orchestration tasks using the **Discover** UI action on the Orchestration Tool form.
   
The tool must be connected for this UI action to be visible.
   
   This step fetches all the jobs or the orchestration tasks using the Discover subflow that was created in the first step.

---

### 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

<table>
<thead>
<tr>
<th>Item</th>
<th>Expected value</th>
</tr>
</thead>
<tbody>
<tr>
<td>Inputs</td>
<td></td>
</tr>
<tr>
<td>Label</td>
<td>current</td>
</tr>
<tr>
<td>Type</td>
<td>Reference.Inbound Event</td>
</tr>
<tr>
<td>Outputs</td>
<td>N/A</td>
</tr>
</tbody>
</table>

Transform the original payload, and save the final payload in the **transformed_payload** field.

### Expected standard JSON Notification capability payload - Planning tool

```
{
  "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"
  },
```

© 2020 ServiceNow, Inc. All rights reserved.
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 names, product names, and logos may be trademarks of the respective companies with which they are associated.
"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",
  "url": "https://github.com/mycompany/mobileplatform/commit/3fa85f6457174562b3fc2c963f66afa6",
  "committedDate": "1970-01-01T08:15:30-05:00",
  "repository": {
    "name": "Platform-Mobile",
    "url": "https://github.com/mycompany/mobileplatform"
  },
  "branch": {
    "name": "master",
    "path": "refs/heads/master"
  },
  "committer": {
    "email": "lenn@smithworksinc.com"
  },
  "details": [
    {
      "additions": 0,
      "deletions": 0,
      "totalChanges": 0,
      "file": "src/test/java/com/mycompany/app/App.java",
      "action": "modified",
      "changes": "%40%40%20-30%2C6%20%2B30%2C18%20%40%40%20public%20void
%20testAppConstructor%28%29%7B%0A%20%20%20%20%7D%0A%20%20%20%20%7B%0A%20%20%20%20%20%20%20%20%20%20%20%20%20%20%20%20%20%20%20%20%20%20%20%20%20%20%20%20%20%20%20%20%20%20%20%20%20%20%20%20%20%20%20%20%20%20%20%20%20%20%20%20%20%20%20%20%20%20%20%20%20%20%20%20%20%20%20%20%20%20%20%20%20%20%20%20%20%20%20%20%20%20%20%20%20%20%20%20%20%20%20%20%20%20%20%20%20%20%20%20%20%20%20%20%20%20%20%20%20%20%20%20%20%20%20%20%20%20%20%20%20%20%20%20%20%20%20%20%20%20%20%20%20%20%20%20%20%20%20%20%20%20%20%20%20%20%20%20%20%20%20%20%20%20%20%20%20%20%20%20%20%20%20%20%20%20%20%20%20%20%20%20%20%20%20%20%20%20%20%20%20%20%20%20%20%20%20%20%20%20%20%20%20%20%20%20%20%20%20%20%20%20%20%20%20%20%20%20%20%20%20%20%20%20%20%20%20%20%20%20%20%20%20%20%20%20%20%20%20%20%20%20%20%20%20%20%20%20%20%20%20%20%20%20%20%20%20%20%20%20%20%20%20%20%20%20%20%20%20%20%20%20%20%20%20%20%20%20%20%20%20%20%20%20%20%20%20%20%20%20%20%20%20%20%20%20%20%20%20%20%20%20%20%20%20%20%20%20%20%20%20%20%20%20%20%20%20%20%20%20%20%20%20%20%20%20%20%20%20%20%20%20%20%20%20%20%20%20%20%20%20%20%20%20%20%20%20%20%20%20%20%20%20%20%20%20%20%20%20%20%20%20%20%20%20%20%20%20%20%20%20%20%20%20%20%20%20%20%20%20%20%20%20%20%20%20%20%20%20%20%20%20%20%20%20%20%20%20%20%20%20%20%20%20%20%20%20%20%20%20%20%20%20%20%20%20%20%20%20%20%20%20%20%20%20%20%20%20%20%20%20%20%20%20%20%20%20%20%20%20%20%20%20%20%20%20%20%20%20%20%20%20%20%20%20%20%20%20%20%20%20%20%20%20%20%20%20%20%20%20%20%20%20%20%20%20%20%20%20%20%20%20%20%20%20%20%20%20%20%20%20%20%20%20%20%20%20%20%20%20%20%20%20%20%20%20%20%20%20%20%20%20%20%20%20%20%20%20%20%20%20%20%20%20%20%20%20%20%20%20%20%20%20%20%20%20%20%20%20%20%20%20%20%20%20%20%20%20%20%20%20%20%20%20%20%20%20%20%20%20%20%20%20%20%20%20%20%20%20%20%20%20%20%20%20%20%20%20%20%20%20%20%20%20%20%20%20%20%20%20%20%20%20%20%20%20%20%20%20%20%20%20%20%20%20%20%20%20%20%20%20%20%20%20%20%20%20%20%20%20%20%20%20%20%20%20%20%20%20%20%20%20%20%20%20%20%20%20%20%20%20%20%20%20%20%20%20%20%20%20%20%20%20%20%20%20%20%20%20%20%20%20%20%20%20%20%20%20%2
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=HILRProd",
    "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",
}
DevOps Notification capability flow diagram example - Coding tool

Connect capability subflow

Starting with version 1.9, connect capability is supported.
<table>
<thead>
<tr>
<th>Item</th>
<th>Expected value</th>
</tr>
</thead>
</table>
| Inputs       | Label: current  
Type: Reference.DevOps Tool  
Tool record for which the **Connect** button action is clicked. |
| Outputs      | • Label: connected  
Flag indicating the success or failure of the connection made to the target tool (true/false).  
• Label: errormessage  
String message displayed on the form for connection failure. The variable is an empty string if the connection is successful. |

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

Starting with version 1.9, discover capability is supported.
### Item | Expected value
--- | ---
**Inputs** | Label: current  
Type: Reference.DevOps Tool  
| **Outputs** | Label: discoverpayload  
Type: Array of objects as JSON string.  
(JSON.stringify([]))  
Planning tool:

```javascript
[
  {
    "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"
  }
]
```

Coding tool:

```javascript
[
  {
    "name": "nvm_repo",
    "url": "https://github.com/nvm_repo/"
  },
  {
    "name": "golang_util",
    "url": "https://github.com/golang_util/"
  }
]
```

---

**Note:** Pagination is not supported for the discover action.

See subflow outputs.

---

The Discover master 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
<table>
<thead>
<tr>
<th>State</th>
<th>Message</th>
</tr>
</thead>
<tbody>
<tr>
<td>Error</td>
<td>• Unable to parse payload string from Subflow</td>
</tr>
<tr>
<td></td>
<td>• discoverpayload is expected to be an array of objects</td>
</tr>
<tr>
<td></td>
<td>• ImportRequest record does not have reference to tool table</td>
</tr>
<tr>
<td>Paused</td>
<td>--</td>
</tr>
<tr>
<td>Canceled</td>
<td>--</td>
</tr>
<tr>
<td>Unmatched</td>
<td>--</td>
</tr>
</tbody>
</table>

**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_APC_1",
      "url": "https://jenkins.wsf.xyz/job/Build_APC_1",
      "projectName": "Build_APC_1"
    },
    {
      "name": "CI_CD_Jenkins",
      "url": "https://pt1.jenkins.com/job/CI_CD_Jenkins",
      "projectName": "CI_CD_Jenkins"
    }
  ]
}
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.

<table>
<thead>
<tr>
<th>Input label</th>
<th>Type</th>
</tr>
</thead>
<tbody>
<tr>
<td>current</td>
<td>Reference. Import Request</td>
</tr>
<tr>
<td>state</td>
<td>(String)</td>
</tr>
<tr>
<td></td>
<td>• requested</td>
</tr>
<tr>
<td></td>
<td>• processing</td>
</tr>
<tr>
<td>details</td>
<td>(String)</td>
</tr>
</tbody>
</table>

Note: Flow execution is stopped (once the return is received from the subflow) when the state is set to processing.

• completed
• error
• paused
• canceled
• unmatched

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.

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

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.

For orchestration tool integration, the integration developer also creates an API manager.

Note: Notification, connect, and discover capabilities are supported.

• Your DevOps admin sets up DevOps connections (planning, coding, or orchestration tool), and configures the source tool with the webhook and credentials.
This procedure provides detailed steps to create your DevOps tool integration.

1. Integration developer:
   Configure the source tool integration capabilities and actions, create an API manager (orchestration tool only), and a subflow.
   a) 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

<table>
<thead>
<tr>
<th>Tool label</th>
<th>Sample Code Tool</th>
</tr>
</thead>
<tbody>
<tr>
<td>Table</td>
<td>Code Tool [sn_devops_tool]</td>
</tr>
<tr>
<td>Use packageable integrations</td>
<td>Selected</td>
</tr>
<tr>
<td>Integration version</td>
<td>1.0</td>
</tr>
<tr>
<td>Active</td>
<td>Selected</td>
</tr>
</tbody>
</table>

   Note: You must set the API manager field in the Tool Integrations list. If you do not have an API manager set up, create one in the next step.

   b) Navigate to System Extension Points > Scripted Extension Points, search, open the DevOpsOrchestrationToolIntegrationHandler scripted extension point, and click Create Implementation.

   Name | Name of the extension instance using the convention: DevOps<ToolName>IntegrationHandler
   Accessible from | All application scopes.

   Note: These mandatory functions must be implemented regardless of how the notification subflow is set up.

   You can use the existing implementations for Azure DevOps, GitLab, and Jenkins as references when implementing these methods.
Mandatory extension point functions

Note: You must set the API manager field in the Tool Integrations list.
c) 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

<table>
<thead>
<tr>
<th>Name</th>
<th>Code Tool Notification</th>
</tr>
</thead>
<tbody>
<tr>
<td>Application</td>
<td>Sample Integration App</td>
</tr>
<tr>
<td>Accessible From</td>
<td>All application scopes</td>
</tr>
<tr>
<td>Description</td>
<td>Code tool for integration app</td>
</tr>
<tr>
<td>Run As</td>
<td>System User</td>
</tr>
</tbody>
</table>

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 Master Flow.

d) 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

<table>
<thead>
<tr>
<th>Tool integration</th>
<th>Sample tool</th>
</tr>
</thead>
<tbody>
<tr>
<td>Tool type capability</td>
<td>Code</td>
</tr>
</tbody>
</table>

e) 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

<table>
<thead>
<tr>
<th>Tool integration</th>
<th>Sample Code Tool</th>
</tr>
</thead>
<tbody>
<tr>
<td>Capability mapping</td>
<td>Sample Code Tool-Code</td>
</tr>
</tbody>
</table>

**Note:** Do not edit tool type capability records.

<table>
<thead>
<tr>
<th>Action</th>
<th>Notification</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td></td>
</tr>
</tbody>
</table>

**Note:** Do not edit tool action records.
### Timeout (ms)
- **Note:** Starting with version 1.10.
  - Timeout for the corresponding subflow. If execution of the subflow exceeds this value, a timeout exception occurs.
  - Value is in milliseconds (ms). Default is 45,000 (45 seconds).

### Subflow name
- **x_snc_sample_integ.code_tool_notification**
  - The name is prefixed by the scope name and a dot (.) before the actual subflow name.
  - For example, given:
    - `connect_code_tool` subflow
    - `my_app_scope` scope
  - The value for this field is `my_app_scope.connect_code_tool`
  - **Note:** Starting with version 1.11, if the **Subflow name** field is left blank for a Notification capability, default handling of notifications occurs. See **Notification tool capability action**.

### Domain
- **global**

---

2. **DevOps admin:**
   - Configure the connection from DevOps to the source tool.
   - Refer to [Setting up DevOps tools](#) for more details on completing this step.
     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.

<table>
<thead>
<tr>
<th>Field</th>
<th>Read</th>
<th>Write</th>
</tr>
</thead>
<tbody>
<tr>
<td>Source Scope</td>
<td>Auto populated based on the current application</td>
<td></td>
</tr>
<tr>
<td>Target Scope</td>
<td>DevOps</td>
<td></td>
</tr>
<tr>
<td>Target Name</td>
<td>sn_devops_inbound_event</td>
<td></td>
</tr>
<tr>
<td>Field</td>
<td>Read</td>
<td>Write</td>
</tr>
<tr>
<td>-------------</td>
<td>------</td>
<td>-------</td>
</tr>
<tr>
<td>Target Type</td>
<td>Table</td>
<td></td>
</tr>
<tr>
<td>Operation</td>
<td>Read</td>
<td>Write</td>
</tr>
<tr>
<td>Status</td>
<td>Allowed</td>
<td></td>
</tr>
</tbody>
</table>
DevOps Tool Integration
Azure DevOps

This record is in the DevOps Integrations application, but DevOps is the current application. To edit this record click here.

<table>
<thead>
<tr>
<th>Tool label</th>
<th>Tool name</th>
</tr>
</thead>
<tbody>
<tr>
<td>Azure DevOps</td>
<td>sn_devops_into_Azure_DevOps</td>
</tr>
</tbody>
</table>

Integration version: 5.1

Active: ✔

Integration Capabilities (7)

<table>
<thead>
<tr>
<th>Capability mapping</th>
<th>Action</th>
<th>Subflow name</th>
<th>Domain</th>
</tr>
</thead>
<tbody>
<tr>
<td>Discover</td>
<td>sn_devops_into_Azure_devops_discover_board</td>
<td>global</td>
<td></td>
</tr>
<tr>
<td>Discover</td>
<td>sn_devops_into_Azure_devops_pipeline_di...</td>
<td>global</td>
<td></td>
</tr>
<tr>
<td>Discover</td>
<td>sn_devops_into_Azure_devops_repos_discover</td>
<td>global</td>
<td></td>
</tr>
<tr>
<td>Connect</td>
<td>sn_devops_into_Azure_devops_connect</td>
<td>global</td>
<td></td>
</tr>
<tr>
<td>sn_devops_into_adop_notification_inbound</td>
<td>global</td>
<td></td>
<td></td>
</tr>
<tr>
<td>sn_devops_into_Azure_devops_repos_notifi...</td>
<td>global</td>
<td></td>
<td></td>
</tr>
<tr>
<td>sn_devops_into_Azure_devops_boards_notifi...</td>
<td>global</td>
<td></td>
<td></td>
</tr>
</tbody>
</table>
DevOpsOrchestrationToolIntegrationHandler extension instance

API manager setting in DevOps tool integration list
Subflow Properties

<table>
<thead>
<tr>
<th><strong>Name</strong></th>
<th>Payview processor</th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>Application</strong></td>
<td>Plan View Integration</td>
</tr>
<tr>
<td><strong>Accessible From</strong></td>
<td>All application scopes</td>
</tr>
<tr>
<td><strong>Category</strong></td>
<td></td>
</tr>
<tr>
<td><strong>Description</strong></td>
<td>Describe your subflow</td>
</tr>
<tr>
<td><strong>Run As</strong></td>
<td>System User</td>
</tr>
</tbody>
</table>

Flow Designer subflow properties
Flow Designer subflow
Flow Designer DevOps Integration - Notification flow
DevOps tool capability mapping

Integration capability
<table>
<thead>
<tr>
<th>Integration Capability</th>
<th>sn_devops_ints.azure_devops_discover_boards</th>
</tr>
</thead>
<tbody>
<tr>
<td>Tool Integration</td>
<td>Azure DevOps</td>
</tr>
<tr>
<td>Capability mapping</td>
<td>Azure DevOps-Plan</td>
</tr>
<tr>
<td>Action</td>
<td>Discover</td>
</tr>
<tr>
<td>Active</td>
<td>☑</td>
</tr>
<tr>
<td>Timeout (ms)</td>
<td>45,600</td>
</tr>
<tr>
<td>Sub-flow name</td>
<td>sn_devops_ints.azure_devops_discover_boards</td>
</tr>
<tr>
<td><strong>Name</strong></td>
<td>Planning tool</td>
</tr>
<tr>
<td>----------</td>
<td>---------------</td>
</tr>
<tr>
<td><strong>Tool</strong></td>
<td>Planview</td>
</tr>
<tr>
<td><strong>Connection alias</strong></td>
<td>sr_devops.planview</td>
</tr>
<tr>
<td><strong>Tool URL</strong></td>
<td></td>
</tr>
<tr>
<td><strong>Connection state</strong></td>
<td>Disconnected</td>
</tr>
<tr>
<td><strong>Last discovery</strong></td>
<td></td>
</tr>
<tr>
<td><strong>Webhook URL</strong></td>
<td><a href="https://api.devops.com/api_devops/s1/DevOps/plans?toolId=2020c98b7dc04310183b01e2f3b8e">https://api.devops.com/api_devops/s1/DevOps/plans?toolId=2020c98b7dc04310183b01e2f3b8e</a></td>
</tr>
</tbody>
</table>

**Planning tool**
DevOps reference

DevOps reference content includes connection and troubleshooting information.

DevOps modules

Insights
Analyze operational and business reports including change acceleration, system health, and development to gain insight into your DevOps environment.

Tools
Create and set up initial configuration connections including planning tools to connect to applications, coding tools to connect to repositories, orchestration tools to connect to tasks, artifact tools to use change acceleration for releases, and test tools.

Apps & Pipelines
Manage apps and create pipelines.

Plan
Access your integrated plans, features (mapped from epics), and versions.

Develop
Access your work items (mapped from your planning application stories), and integrated coding tool repositories. Review development activity including branches, commits, tags (tied to commits), commit details, and committers.

Orchestrate
Access your integrated orchestration tasks and task executions. Review pipeline change requests, registered callbacks, and pipeline executions.

Artifact
Access artifacts and packages to view all commits and work items in a change request for a DevOps release.

**Test Results**

View test summaries and performance test summaries from your integrated test tools.

**Test**

View build test summaries and test results.

**Administration**

Access administrative content such as DevOps properties, error log, inbound events, and payloads. You can also access quick start tests.

**Integrations**

Create and access DevOps tool integration setup.

**Use a token to connect to a DevOps tool**

Connect to a DevOps tool, such as GitHub or Jenkins, with basic authentication using a token instead of using a user name and password.

Role required: sn_devops.admin

Connecting using token-based authentication is useful not only to keep from having to enter your user name and password in a third-party tool, but you can also set specific privileges per token. And, if needed, tokens can be easily revoked, which makes them a more flexible option if the sole purpose is for connecting a tool.

There are three parts to setting up a connection using a token:

- Generate a token in the tool (Jira, GitHub, or Jenkins)
- Set up basic authentication credentials in DevOps
- Use DevOps to connect to the tool

1. Generate and copy a token in GitHub for a coding tool connection, or Jenkins for an orchestration tool connection.
   - Log in to GitHub and navigate to Settings > Developer settings.
     1. Select Personal access tokens and click Generate new token.
     2. Enter ServiceNow DevOps Coding Tool for the description and select repo for the scope.
     3. Click Generate Token.
   - Log in to Jenkins and navigate to Configure > API Token and click Add new Token.
     1. Enter ServiceNow Token for the name.
     2. Click Generate.

   **Note:** You must copy the token at this time (for use in the DevOps coding configuration) because you do not have access to the token once the GitHub screen is closed.

2. Create an application connection to DevOps tools using basic authentication.
   a) Navigate to Connections and Credentials > Credentials and create a new record.
   b) Select HTTP(s) Connection and create a new Credential record using Basic Auth Credentials.
c) Enter a name, corresponding GitHub or Jenkins user name, and the token generated in the tool previously and click Submit.

3. Create a DevOps coding tool connection to connect to GitHub, or an orchestration tool connection to connect to Jenkins.
   a) For GitHub, navigate to DevOps > Configure > Coding Tools and create a record.
   b) For Jenkins, Navigate to DevOps > Configure > Orchestration Tools and create a record.
   c) Enter a name and select the corresponding tool (GitHub or Jenkins) for the Type field.
   d) Use the connection Connection alias lookup list to select the basic authentication connection created previously and click Submit.
   e) Open the new coding or orchestration tool record from the list and click Connect.
      The successful connection status message is shown.

DevOps connections setup using a token
Basic authentication setup
Tracking DevOps notifications

Turn on tracking for a DevOps planning tool app, coding tool repository, or orchestration tool task to receive specific event notifications.

Track DevOps notifications to focus on specific events when you have multiple DevOps items configured for each tool. When tracking is enabled, you can see specific event details, such as code commits, for all events moving forward.

You can track events for these tool types:

- Planning applications
- Coding repositories
- Orchestration tasks

Note: To enable the tracking feature, select the **Track** check box in the individual record (app, repository, or orchestration task). Tracking is automatically enabled after import.

Event payloads

Notifications for a tracked apps, repositories, and orchestration tasks are shown in the **Event Payloads > All Payloads** Events list.
You can view all payloads, or only unmatched payloads from the navigation menu. The unmatched payloads Events list shows events for tools not matched to a DevOps ID.

**Event payload states**

<table>
<thead>
<tr>
<th>Event State</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>Ready</td>
<td>The event was received.</td>
</tr>
<tr>
<td>Processing</td>
<td>Tracking is enabled and the event is being processed.</td>
</tr>
<tr>
<td>Processed</td>
<td>The tracked event has completed processing.</td>
</tr>
<tr>
<td>Not Connected</td>
<td>The tool for this event is not connected. In this case, the event cannot be processed.</td>
</tr>
<tr>
<td>Ignored</td>
<td>The track feature is not enabled for the tool in the event. In this case, the event is ignored.</td>
</tr>
<tr>
<td>Error</td>
<td>Event processing failed. Processing can fail if, for example, there is an incorrect link for a tool item.</td>
</tr>
</tbody>
</table>

**Quick start tests for DevOps**

Starting with version 1.10, validate that DevOps still works after you make any configuration change such as apply an upgrade or develop an application in version 1.10, 1.11, and 1.12. Copy and customize these quick start tests to pass when using your instance-specific data.

DevOps quick start tests are available when you install the DevOps application from ServiceNow Store.

**DevOps test suite**

<table>
<thead>
<tr>
<th>Test</th>
<th>Description</th>
<th>Release version</th>
</tr>
</thead>
<tbody>
<tr>
<td>DevOps Orchestration Flow with CR</td>
<td>Verify the Flow Designer flow for a DevOps orchestration tool that includes a change request.</td>
<td>Madrid</td>
</tr>
<tr>
<td>DevOps Orchestration Tool Flow</td>
<td>Verify the Flow Designer flow for a DevOps orchestration tool.</td>
<td>Madrid</td>
</tr>
<tr>
<td>DevOps Plan Tool Flow</td>
<td>Verify the Flow Designer flow for a DevOps planning tool.</td>
<td>Madrid</td>
</tr>
</tbody>
</table>

**DevOps troubleshooting**

Troubleshooting actions can help resolve common issues when setting up or running the DevOps application.
<table>
<thead>
<tr>
<th>Issue</th>
<th>Action</th>
</tr>
</thead>
</table>
| Import request not progressing | 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.  

**Note:** You must delete the existing request to retry importing the same range. |
| Tool connection fails | Remove the trailing slash (`/`) in the **Connection URL** field on the HTTP Connection form. |
| No change request is created for a Jenkins job under change control | Verify that:  
• The tool integration in your instance is set up properly.  
• The task has been synced in your instance.  
• Tasks and app steps have been configured in your instance.  

Change request creation is not supported if the task is under change control:  
• Is not part of a pipeline (is a standalone task, for example).  
• Is the first in the pipeline.  
• 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). |
| Jenkins does not block the job under change control (does not wait for change request approval) | Verify that the Jenkins location is configured:  
Navigate to Jenkins > Manage Jenkins > Configure System and provide the hostname for the **Jenkins URL** field in the Jenkins Location section.  

**Note:** To avoid caching issues, click **Save** even if the **Jenkins URL** field already contains a value when you first open the form. |
| Events occurring in the payload log with state Not Connected | If any of the following changes for a connection made manually (using manual configuration mode), the connection is automatically disconnected.  
• Alias associated with the tool  
• Type of tool  
• New active HTTP connection for the same domain added to the alias  
• Existing HTTP connection for the same domain activated  
• Connection URL of the HTTP connection  
• Credentials of the HTTP connection  
• Use MID Server setting in the HTTP connection  

Enter manual configuration mode and reconnect. |
## Properties installed with DevOps

Use these properties to configure settings in the DevOps application.

Role required: sn_devops.admin

These properties apply to the DevOps application ([DevOps > Administration > Properties](#)).

<table>
<thead>
<tr>
<th>Property</th>
<th>Description</th>
<th>Default</th>
</tr>
</thead>
<tbody>
<tr>
<td>[sn_devops.github.api_url]</td>
<td>GitHub API URL.</td>
<td><a href="https://api.github.com">https://api.github.com</a></td>
</tr>
<tr>
<td>[sn_devops.import.max.retries.per_page]</td>
<td>Maximum retries per page while importing.</td>
<td>3</td>
</tr>
<tr>
<td>[sn_devops.committer.score.default]</td>
<td>Default committer score.</td>
<td>50</td>
</tr>
<tr>
<td>[sn_devops.committer.score.multiply.factor]</td>
<td>Multiply factor for committer score.</td>
<td>1</td>
</tr>
<tr>
<td>[sn_devops.enable_debug]</td>
<td>Enable Debug Flag.</td>
<td>Yes</td>
</tr>
<tr>
<td>[sn_devops.import.max.processing.time.seconds.per_page]</td>
<td>Maximum processing time per page (in seconds), while importing.</td>
<td>300</td>
</tr>
<tr>
<td>[sn_devops.supported_webhook_capabilities]</td>
<td>Supported Webhook Capabilities</td>
<td>code,plan,orchestration,artifact,test</td>
</tr>
<tr>
<td>[sn_devops.import.coding_tool.commits.per_page]</td>
<td>Coding tool commits per page.</td>
<td>100</td>
</tr>
<tr>
<td>[sn_devops.import.coding_tool.repos.per_page]</td>
<td>Coding tool repositories per page.</td>
<td>100</td>
</tr>
<tr>
<td>[sn_devops.import.planning_tool.issues.per_page]</td>
<td>Planning tool issues per page.</td>
<td>100</td>
</tr>
<tr>
<td>[sn_devops.import.orchestration_tool.executions.per_page]</td>
<td>Orchestration tool executions per page.</td>
<td>50</td>
</tr>
<tr>
<td>[sn_devops.import.save.payloads.as.attachments]</td>
<td>Save payloads as attachments on the Import Request Page record, set the Value field to true. Anything else is considered false.</td>
<td>false</td>
</tr>
<tr>
<td>[sn_devops.default_test_type]</td>
<td>Default test type</td>
<td>JUnit</td>
</tr>
<tr>
<td>[sn_devops.import.coding_tool.branches.per_property]</td>
<td>Coding tool branches per property.</td>
<td>19</td>
</tr>
<tr>
<td>[sn_devops.import.planning_tool.issues.per_page]</td>
<td>Planning tool issues per page.</td>
<td>100</td>
</tr>
<tr>
<td>[sn_devops.discovered.user.auto.assign.role]</td>
<td>Provided sn_devops role in the value field is automatically added to users who are active DevOps users (for example, making a commit). If a role not in the sn_devops scope is provided, it is not added. Leave empty if no role should be automatically provided.</td>
<td>sn_devops.viewer</td>
</tr>
<tr>
<td>Property</td>
<td>Description</td>
<td>Default</td>
</tr>
<tr>
<td>----------</td>
<td>-------------</td>
<td>---------</td>
</tr>
<tr>
<td>[sn_devops.change_request.implement_state]</td>
<td>DevOps change request implement state. This property is useful for change management customizations.</td>
<td>Default is -1 (Implement).</td>
</tr>
<tr>
<td>[sn_devops.change_request.post_implement_state]</td>
<td>DevOps change request post implement state. This property is useful for change management customizations.</td>
<td>Default is 0 (Review).</td>
</tr>
<tr>
<td>[sn_devops.change_request.cancel_state]</td>
<td>DevOps change request cancel state. This property is useful for change management customizations.</td>
<td>Default is 4 (Canceled).</td>
</tr>
<tr>
<td>[sn_devops.change_request.handler_subflow]</td>
<td>DevOps Change Request Handler subflow.</td>
<td>Default is sn_devops.default_change_handler_subflow.</td>
</tr>
<tr>
<td>[sn_devops.change_request.approved_approval]</td>
<td>DevOps change request approval text. This property is useful for change management customizations.</td>
<td>Approved</td>
</tr>
<tr>
<td>[sn_devops.import.planning_tool.itbm.issues.max.limit]</td>
<td>ITBM Agile planning tool issues import max limit.</td>
<td>1000</td>
</tr>
</tbody>
</table>

**Components installed with DevOps**

Several types of components are installed with DevOps, including tables, users, roles, and scheduled jobs. Demo data is available for this application.

**Users installed**

<table>
<thead>
<tr>
<th>User</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>devops.integration.user</td>
<td>DevOps Integration User</td>
</tr>
<tr>
<td>devops.system</td>
<td>DevOps System user</td>
</tr>
</tbody>
</table>

**Note:** You must configure the password before the DevOps Integration User can configure a tool.
Roles installed

<table>
<thead>
<tr>
<th>Role title [name]</th>
<th>Description</th>
<th>Contains roles</th>
</tr>
</thead>
<tbody>
<tr>
<td>DevOps Administrator</td>
<td>Sets up and configures the DevOps application.</td>
<td>• sn_devops.manager&lt;br&gt;• connection_admin&lt;br&gt;• action_designer&lt;br&gt;• credential_admin&lt;br&gt;• flow_designer</td>
</tr>
<tr>
<td>[sn_devops.admin]</td>
<td></td>
<td></td>
</tr>
<tr>
<td>DevOps integration</td>
<td>Has inbound access to the tools in your environment to allow integration with the DevOps application.</td>
<td>• None.</td>
</tr>
<tr>
<td>[sn_devops.integration]</td>
<td></td>
<td></td>
</tr>
<tr>
<td>DevOps manager</td>
<td>Oversees the operation of the DevOps application and monitors performance in your DevOps environment.</td>
<td>• sn_devops.viewer&lt;br&gt;• cmdb_read</td>
</tr>
<tr>
<td>[sn_devops.manager]</td>
<td></td>
<td></td>
</tr>
<tr>
<td>DevOps viewer</td>
<td>Has access to the DevOps application to use in their environment.</td>
<td>None.</td>
</tr>
<tr>
<td>[sn_devops.viewer]</td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

Scheduled jobs installed

<table>
<thead>
<tr>
<th>Scheduled job</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>[DevOps] Historical Data Collection</td>
<td>Collects data from DevOps tools on demand. <strong>Note:</strong> You must run this scheduled job to see historical DevOps data after installation of the DevOps Insights application.</td>
</tr>
<tr>
<td>[DevOps] Daily Data Collection</td>
<td>Collects data from DevOps tools daily.</td>
</tr>
</tbody>
</table>

Tables installed

<table>
<thead>
<tr>
<th>Name</th>
<th>Table</th>
</tr>
</thead>
<tbody>
<tr>
<td>API Schema Definition</td>
<td>[sn_devops_api_schema_definition]</td>
</tr>
<tr>
<td>App</td>
<td>[sn_devops_app]</td>
</tr>
<tr>
<td>Artifact</td>
<td>[sn_devops_artifact]</td>
</tr>
<tr>
<td>Artifact Repository</td>
<td>[sn_devops_artifact_repository]</td>
</tr>
<tr>
<td>Artifact Staged Request</td>
<td>[sn_devops_artifact_staging]</td>
</tr>
<tr>
<td>Artifact Version</td>
<td>[sn_devops_artifact_version]</td>
</tr>
<tr>
<td>Name</td>
<td>Table</td>
</tr>
<tr>
<td>-----------------------------</td>
<td>------------------------------------------------</td>
</tr>
<tr>
<td>Base Planning Item</td>
<td>[sn_devops_base_planning_item]</td>
</tr>
<tr>
<td>Branch</td>
<td>[sn_devops_branch]</td>
</tr>
<tr>
<td>Build Test Result</td>
<td>[sn_devops_build_test_result]</td>
</tr>
<tr>
<td>Build Test Summary</td>
<td>[sn_devops_build_test_summary]</td>
</tr>
<tr>
<td>Callback</td>
<td>[sn_devops_callback]</td>
</tr>
<tr>
<td>Commit</td>
<td>[sn_devops_commit]</td>
</tr>
<tr>
<td>Committer</td>
<td>[sn_devops_committer]</td>
</tr>
<tr>
<td>Commit Details</td>
<td>[sn_devops_commit_details]</td>
</tr>
<tr>
<td>Contributor Score Change Factor</td>
<td>[sn_devops_contributor_score_chg_factor]</td>
</tr>
<tr>
<td>Environment</td>
<td>[sn_devops_environment]</td>
</tr>
<tr>
<td>Event</td>
<td>[sn_devops_event]</td>
</tr>
<tr>
<td>Event Processor</td>
<td>[sn_devops_event_processor]</td>
</tr>
<tr>
<td>Import Filter</td>
<td>[sn_devops_import_filter]</td>
</tr>
<tr>
<td>Import Request</td>
<td>[sn_devops_import_request]</td>
</tr>
<tr>
<td>Import Request Page</td>
<td>[sn_devops_import_request_page]</td>
</tr>
<tr>
<td>Inbound Event</td>
<td>[sn_devops_inbound_event]</td>
</tr>
<tr>
<td>Integration Capability</td>
<td>[sn_devops_integration_capability]</td>
</tr>
<tr>
<td></td>
<td>Extends table Application File.</td>
</tr>
<tr>
<td>App to Plan</td>
<td>[sn_devops_m2m_app_plan]</td>
</tr>
<tr>
<td>Deployed Artifact to TaskExecution</td>
<td>[sn_devops_m2m_artifact_execution]</td>
</tr>
<tr>
<td>Artifact Version to Commit</td>
<td>[sb_devops_m2m_artifact_version_commit]</td>
</tr>
<tr>
<td>Artifact Version to Package</td>
<td>[sn_devops_m2m_artifact_version_package]</td>
</tr>
<tr>
<td>Branch To Commit</td>
<td>[sn_devops_m2m_branch_commit]</td>
</tr>
<tr>
<td>Run Commit</td>
<td>[sn_devops_m2m_commit_execution]</td>
</tr>
<tr>
<td>Work Item To Plan Version</td>
<td>[sn_devops_m2m_work_item_plan_version]</td>
</tr>
<tr>
<td>Orchestration Task</td>
<td>[sn_devops_orchestration_task]</td>
</tr>
<tr>
<td>Orchestration Task Definition</td>
<td>[sn_devops_orchestration_task_definition]</td>
</tr>
<tr>
<td>Name</td>
<td>Table</td>
</tr>
<tr>
<td>---------------------------</td>
<td>-------------------------------------</td>
</tr>
<tr>
<td>Package</td>
<td>[sn_devops_package]</td>
</tr>
<tr>
<td></td>
<td>Extends table Configuration Item.</td>
</tr>
<tr>
<td>Participant</td>
<td>[sn_devops_participant]</td>
</tr>
<tr>
<td>Pipeline</td>
<td>[sn_devops_pipeline]</td>
</tr>
<tr>
<td>Pipeline Execution</td>
<td>[sn_devops_pipeline_execution]</td>
</tr>
<tr>
<td>Plan</td>
<td>[sn_devops_plan]</td>
</tr>
<tr>
<td></td>
<td>Extends table Base Planning Item.</td>
</tr>
<tr>
<td>Plan Version</td>
<td>[sn_devops_plan_version]</td>
</tr>
<tr>
<td></td>
<td>Extends table Base Planning Item.</td>
</tr>
<tr>
<td>Repository</td>
<td>[sn_devops_repository]</td>
</tr>
<tr>
<td>Step</td>
<td>[sn_devops_step]</td>
</tr>
<tr>
<td>Step Execution</td>
<td>[sn_devops_step_execution]</td>
</tr>
<tr>
<td>Tag</td>
<td>[sn_devops_tag]</td>
</tr>
<tr>
<td>Task Execution</td>
<td>[sn_devops_task_execution]</td>
</tr>
<tr>
<td>Test Execution</td>
<td>[sn_devops_test_execution]</td>
</tr>
<tr>
<td>Test Result</td>
<td>[sn_devops_test_result]</td>
</tr>
<tr>
<td>Test Type</td>
<td>[sn_devops_test_type]</td>
</tr>
<tr>
<td>DevOps Tool</td>
<td>[sn_devops_tool]</td>
</tr>
<tr>
<td>Tool Action</td>
<td>[sn_devops_tool_action]</td>
</tr>
<tr>
<td>Tool Capability Mapping</td>
<td>[sn_devops_tool_capability_mapping]</td>
</tr>
<tr>
<td>DevOps Tool Integration</td>
<td>[sn_devops_tool_integration]</td>
</tr>
<tr>
<td></td>
<td>Extends table Application File.</td>
</tr>
<tr>
<td>Tool Type Capability</td>
<td>[sn_devops_tool_type_capability]</td>
</tr>
<tr>
<td>Work Item</td>
<td>[sn_devops_work_item]</td>
</tr>
<tr>
<td></td>
<td>Extends table Base Planning Item.</td>
</tr>
</tbody>
</table>