Setup Guide

DevOps Center

Beta, Spring ’22

@salesforcedocs

Last updated: April 21, 2022

© Copyright 2000–2022 Salesforce, Inc. All rights reserved. Salesforce is a registered trademark of Salesforce Inc., as are other names and marks. Other marks appearing herein may be trademarks of their respective owners.

Important: This feature is a Beta Service. Customer may opt to try such Beta Service in its sole discretion. Any use of the Beta Service is subject to the applicable Beta Services Terms provided at Agreements and Terms.

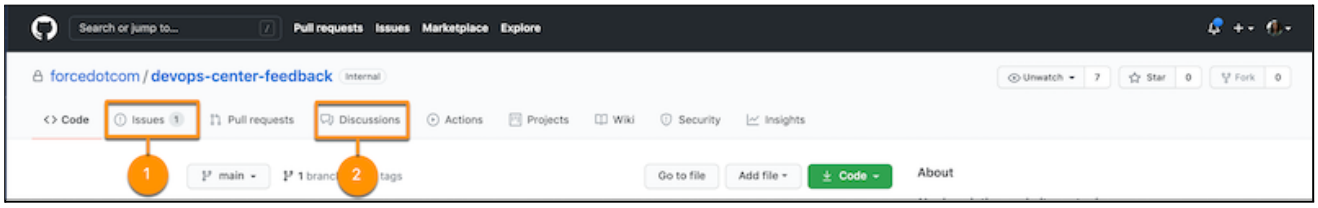
CONTENTS

Welcome To The DevOps Center Beta

Congratulations! Your team has been nominated to participate in the DevOps Center beta. We thank you for taking the time to use DevOps Center and provide feedback. Your input is instrumental in improving the product and user experience.

Ask questions and post feedback in the DevOps Center GitHub repo. See `readme.md` for all the details.

This private shared repository is the central location for sharing information and collecting feedback on the beta. Use the Issues tab (1) at the top of the repo to capture bugs, feature requests, and general feedback. Access to this repository is limited to participants of the beta and members of the Salesforce DevOps Center product team.



Share this form with any users on your team who are participating in this beta so that we can provide them with access to this repository.

See also:
DevOps Center Feedback repo: Known Issues

A Reminder About Beta Software

This product is a beta release. The software isn't yet feature complete. You'll be able to use the product to evaluate how it might work for your team's situation and environment, and to provide us with feedback on the product and your experience with it as we continue through our development process.

Updates to DevOps Center

DevOps Center is delivered as a managed package. We'll occasionally push updates to the managed package to deliver ongoing improvements to the product. Whenever possible, these updates won't disrupt you. If you need to take any action after an update, we'll reach out to let you know.

Install DevOps Center (Beta)

Install the DevOps Center package in a sandbox, Developer Edition org, or production org using the provided installation URL.

Before You Begin:

- Did you receive a Welcome email that your org has been activated to use DevOps Center?
- Do you have the appropriate user permissions to install and configure DevOps Center in the sandbox or production org?

If you have any issues during installation, open an issue in the repository or send an email to: sf-devopscenter-preview@salesforce.com.

See also:
DevOps Center Feedback repo: Known Issues
Uninstall DevOps Center

EDITIONS

Available in:

- Enterprise
- Performance
- Professional
- Unlimited
- Developer
- All sandbox types

USER PERMISSIONS

To install and configure DevOps Center:

- "Download AppExchange Packages"

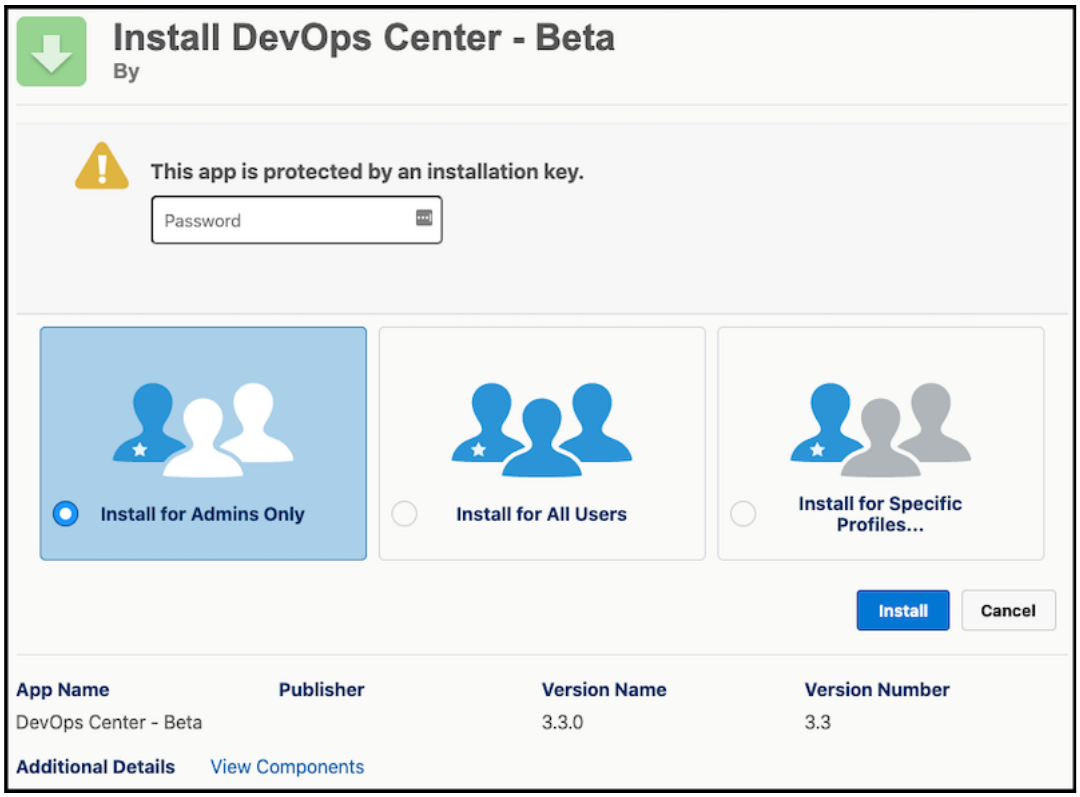
Install the DevOps Center Package

1. In your org, enter this URL:
   `https://<your-domain>.lightning.force.com/packaging/installPackage.apexp?p0=04t6g000008ffbtAAA`

   Example:

   Home page URL:
   `https://my-customization-7147.lightning.force.com/lightning/page/home`

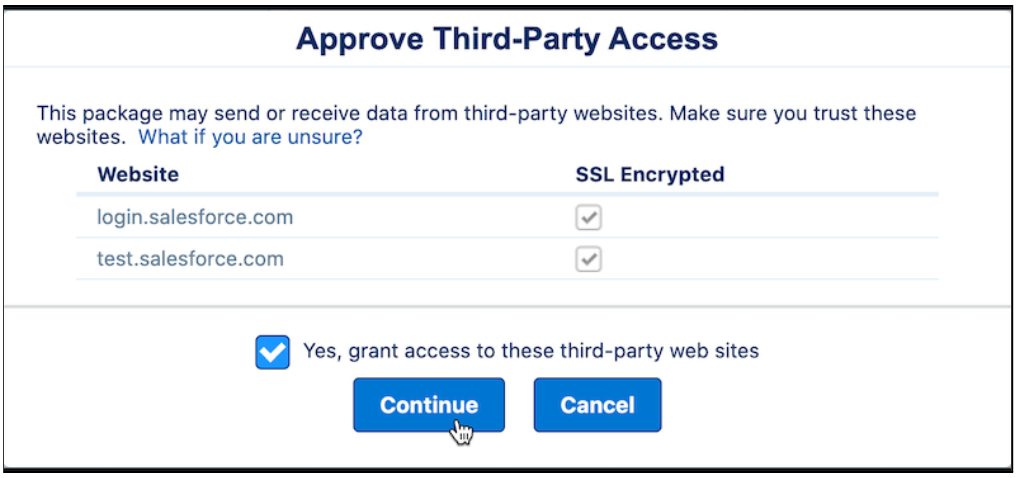
   Installation URL:
   `https://my-customization-7147.lightning.force.com/packaging/installPackage.apexp?p0=04t6g000008ffbtAAA`



Important: The installation succeeds only if the DevOps Center product team enabled your org with the appropriate permissions. If the installation fails, open an issue in the repository or send an email to: sf-devopscenter-preview@salesforce.com. If you see an error that you don't have the necessary authentication providers, and this is your first time installing DevOps Center, it's highly likely that we need to enable DevOps Center in your org. Please contact us – don't attempt to create them manually.

```
(DevOps_Center_GitHub) In field: authProvider - no AuthProvider named
DEVOPS_CENTER_GITHUB found DevOps_Center_GitHub: In field: authProvider - no
AuthProvider named DEVOPS_CENTER_GITHUB found
```

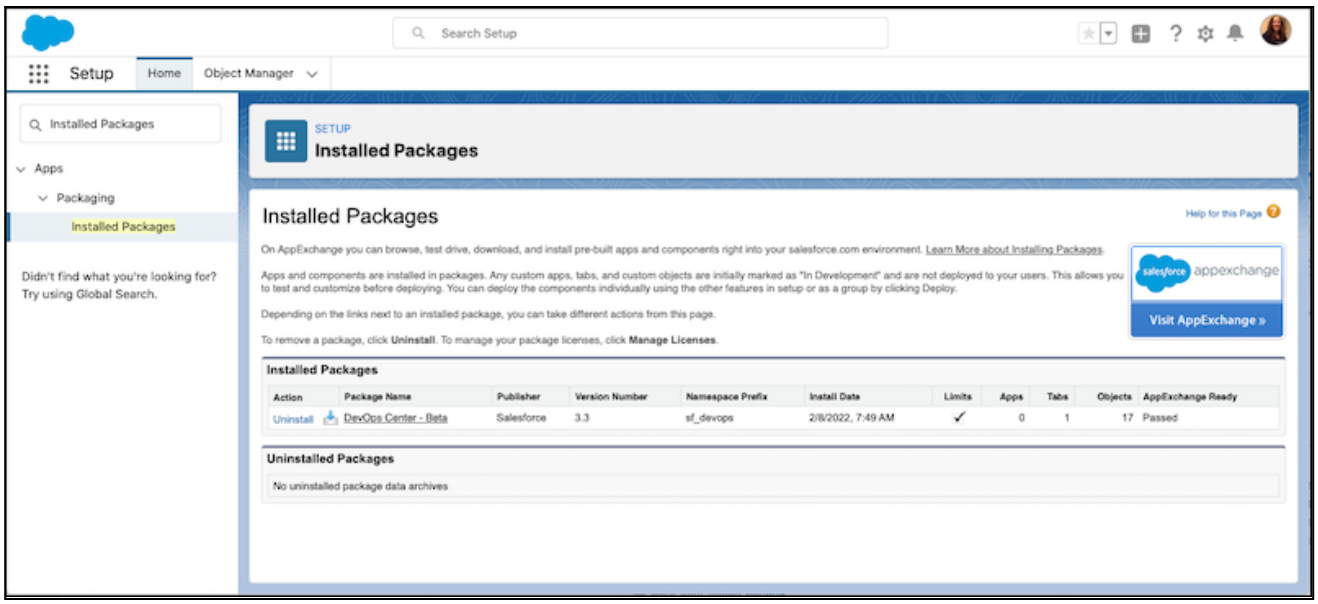
2. For the installation key, see the Welcome email.
3. Select **Install for Admins Only**.
4. Select the acknowledgement.
5. Click **Install**.
6. Approve third-party access.



Confirm Installation

When installation is complete, you receive a confirmation email. You can confirm the installation on the Installed Packages Setup page. If you get the message that the app is taking a long time to install, you get automatically redirected to this page after you click **Done**.

From Setup, enter `Installed Packages` in the Quick Find box, then select **Installed Packages**.

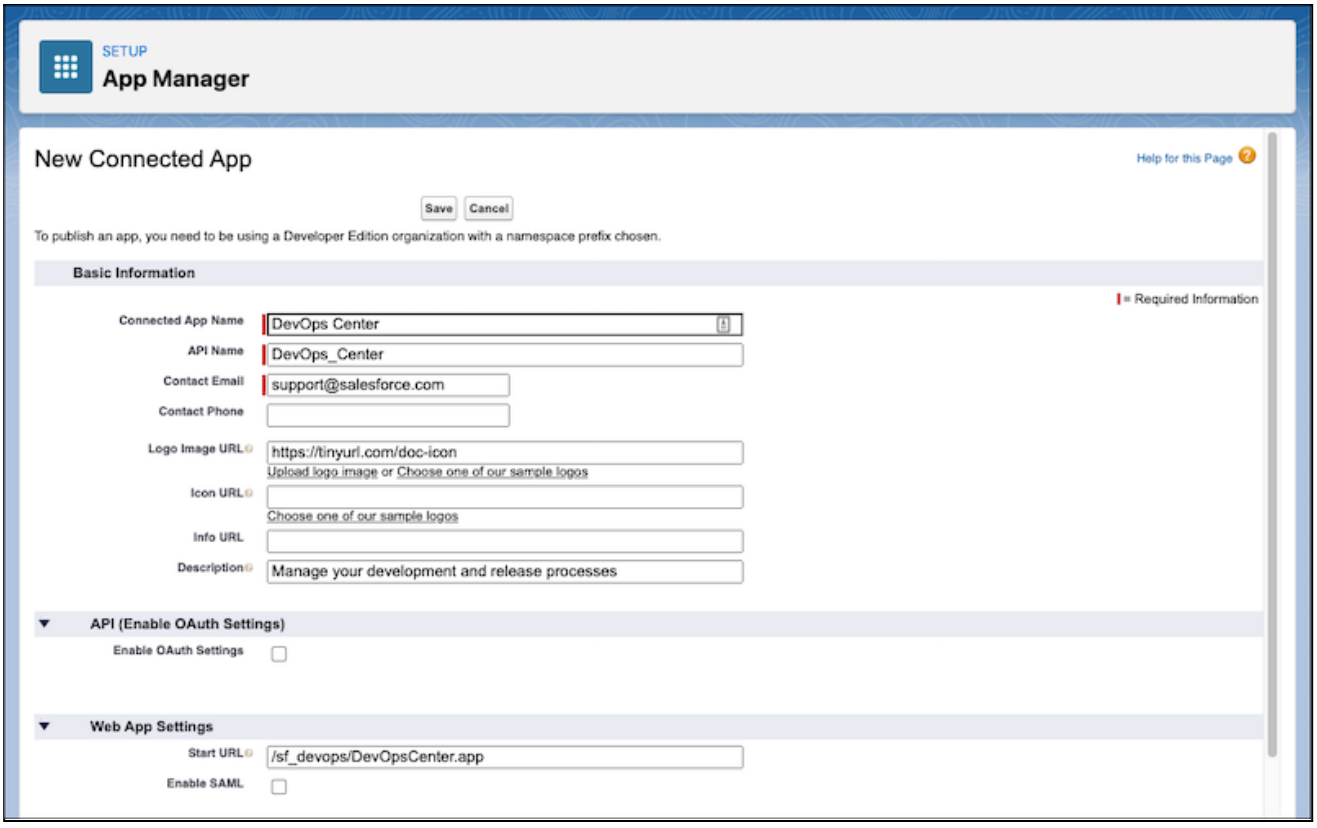


See also:
DevOps Center Feedback repo: Known Issues

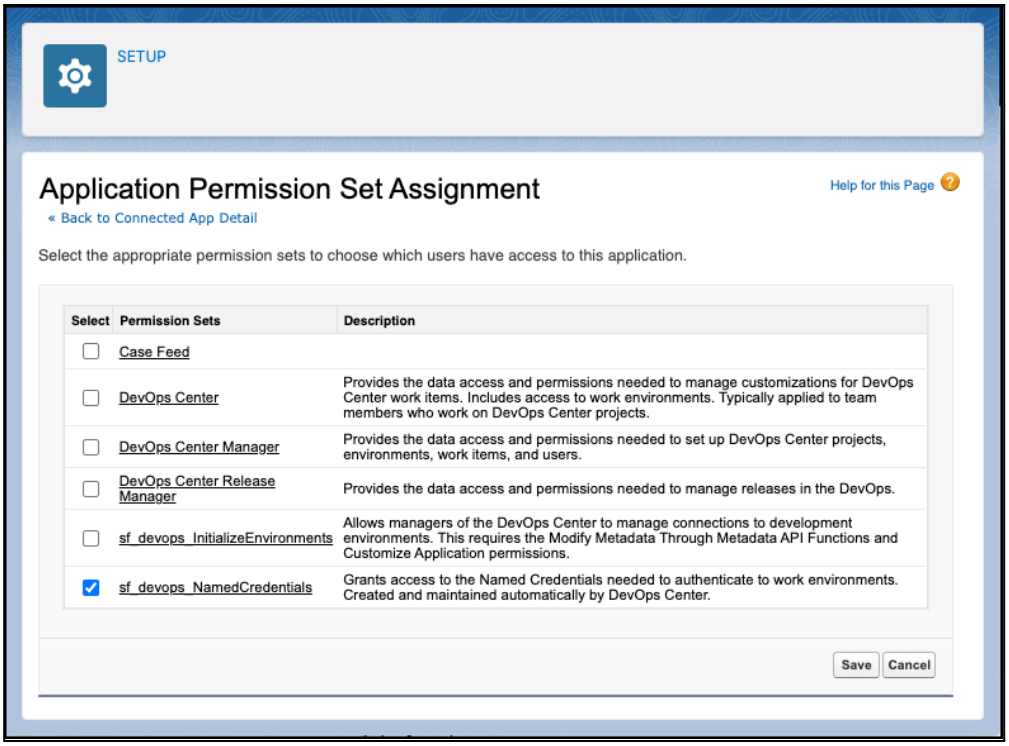
Create the Connected App

The connected app provides access to DevOps Center in the App Launcher. The app appears in App Launcher after users are assigned the appropriate DevOps Center permission sets.

1. From Setup, enter App Manager in the Quick Find box, then select **App Manager**.
2. Click **New Connected App**.
3. In Basic Information:
    a. **Connected App Name**: DevOps Center
    b. **API Name**: DevOps_Center
    c. **Contact Email**: support@salesforce.com
    d. **Logo Image URL**: https://tinyurl.com/doc-icon
    e. **Description**: Manage your development and release processes
4. In Web App Settings, enter the **Start URL**: /sf_devops/DevOpsCenter.app



5. Click **Save**.
6. In Manage Connected Apps, click **Manage**.
7. In the Permissions Sets section, click **Manage Permission Sets**.
8. Select **sf_devops_NamedCredentials**, then click **Save**.



Add Team Members as Users in the DevOps Center Org

Add any team members who aren't already users in the DevOps Center org. For each team member, specify the appropriate license and profile based on their role. The listed licenses and profiles are the minimum required to use DevOps Center for the associated roles. Users can also have a more fully featured license or profile that provides them access beyond the minimum requirements for DevOps Center. However, be aware of what access you are delegating to your users in your org.

Minimum Required Licenses and Profiles

Role	License	Profile
Project Manager, Release Manager (anyone who manages and add environments to DevOps Center)	Salesforce	Standard User
Team members	Salesforce Limited Access - Free	Limited Access User

Tip: This procedure generates an email inviting the new users into the org. But until you're finished setting up DevOps Center, there's not much for them to do in the org. We recommend that you let your team know that you're setting up DevOps Center and to wait until they hear from you before logging in.

1. Log in to the DevOps Center org.
2. From Setup, enter `Users` in the Quick Find box, then select **Users**.
3. Click **New User** or **Add Multiple Users**.
4. Select the appropriate license type and profile based on the user's role.
5. Select the **Generate passwords and notify user via email** checkbox.
6. Click **Save**.

See also:
Salesforce Help: View and Manage Users
Salesforce Help: Standard Profiles
Salesforce Help: User Licenses
Salesforce DX Developer Guide: Free Limited Access License

Assign the DevOps Center Permission Sets

Assign permission sets to everyone working on your project in DevOps Center. Consider who needs to change project-level settings (such as adding another work environment) and who needs access to only work items.

DevOps Center Permission Sets

Permission Set	Description
DevOps Center	The base permission set for DevOps Center. Provides the data access and permissions needed to manage customizations for DevOps Center work items. Ability to view all connected environments and pipelines. **Assign to:** all DevOps Center users, including team members also assigned the DevOps Center Manager permission set. The two permission sets don't overlap.
DevOps Center Manager	Provides the data access and permissions needed to set up DevOps Center projects, environments, and users. **Assign to**: team/project managers
DevOps Center Release Manager	Provides permissions to perform promotions through the pipeline. **Assign to:** release manager
sf_devops_InitializeEnvironments	Allows managers of DevOps Center projects to manage the connections to work environments. Includes the Modify Metadata Through Metadata API Functions and Customize Application user permissions, so the manager can create

Permission Set	Description
	NamedCredential records. **Assign to:** team/project managers
sf_devops_NamedCredentials	Grants access to the named credentials needed to authenticate to environments. Created and maintained automatically by DevOps Center. **Assign to:** all DevOps Center users

See also:
Salesforce Help: Permission Sets
Salesforce Help: Named Credentials

Assign These Permission Sets to All DevOps Center Users

1. From Setup, enter `Permission Sets` in the Quick Find box, then select **Permission Sets**.
2. Select the **DevOps Center** permission set.
3. Click **Manage Assignments** and then **Add Assignments**.
4. Select the checkboxes next to the names of the users you want assigned to the permission set, and click **Assign**.
5. Click **Done**.
6. Repeat the procedure to assign the **sf_devops_NamedCredentials** permission set to your team members.

Assign Additional Permission Sets to DevOps Center Managers

Add the **DevOps Center Manager** and **DevOps Center Release Manager** permission sets to team members who need permissions to configure projects, build pipelines, and promote changes through the pipeline.

Setup Workflow

As the administrator, you can add team members, set up projects, add environments, configure your pipeline, and create work items.

Here's a sneak-peak at the overall process:

1. Decide which method to use to create a GitHub project repository.
2. Identify and create your environments.
3. Open DevOps Center.
4. Create a project.
5. Connect to the release environment.
6. Add development environments.
7. Add your project team members as collaborators in the project's GitHub repository.
8. Add your project team members as users in the DevOps Center org that Salesforce created for you.
9. Create and assign project work items.
10. Assign the team members the permission sets that allow them to address project work items.
11. Bring your team members into the DevOps Center org.

GitHub Project Repositories

A GitHub repository (sometimes called a *repo* for short) stores project work files – code, text, images, and so on. Each DevOps Center project needs its own repository for storing project changes. While you're working on the project, the repository is the team's centralized source of truth to manage changes.

Note: DevOps Center uses the OAuth 2.0 open protocol to establish access to both your GitHub repository (hosted by GitHub) and your work environments (hosted by Salesforce). OAuth allows you to delegate a client application (DevOps Center) to access data from a protected resource (your project repository, for example) through the exchange of tokens, instead of exchanging security credentials. For details, see Authorize Apps with OAuth in Salesforce Help.

We've designed DevOps Center to eventually integrate with multiple third-party source control systems, such as Bitbucket, GitLab, and GitHub. For the beta, all participants use GitHub as the source control system for DevOps Center projects. All participants need their own *GitHub-hosted cloud-based GitHub.com* account to work in this version of DevOps Center. Enterprise or locally hosted versions of GitHub aren't currently supported at this time.

Create a GitHub Account

If you already have a GitHub account, great! If you don't have a GitHub account yet, it's easy (and free) to sign up for one.

New to GitHub or to Source Control?

We've designed DevOps Center to make it easy to take advantage of a source control system like GitHub even if you aren't yet familiar with it. If you want to learn more about Git and GitHub concepts and terminology as you dive in to using DevOps Center, look at the Git and GitHub Basics Trailhead module (estimated time: less than two hours), which covers why source control is so key to successful team collaboration and what to expect in a typical GitHub workflow.

Methods to Create GitHub Project Repositories

A DevOps Center project repository must contain a Salesforce DX project. You can create a DevOps Center project repository using any of these methods:

- When you create your project within DevOps Center, let it create a Github repository that uses the Salesforce DX project structure. Skip to Identify Your Environments.
- Use an existing Github repository.
- Create a new GitHub repository.

Note: If you plan to use an existing GitHub repository, it must contain an `sfdx-project.json` file in the root directory, a file that identifies the repo as a Salesforce DX project.

Create a New Repository

Creating a repository from our template ensures that the repository has the right structure and configuration to work with DevOps Center, that is, the structure and configuration of a Salesforce DX project.

Log in to GitHub and use the repository template at https://github.com/forcedotcom/dx-empty to create your project repository. Your project repository can be either public or private. You don't need to include all branches. For details about how to create a repository from a template, see the GitHub help.

Identify Your Environments

Have the login credentials handy for the environments you plan to use for your DevOps Center project. Before you proceed, make sure all the environments you need for this project are created.

First, be sure that Source Tracking for Sandboxes is enabled in any orgs from which you plan to create your developer sandboxes.

Important: If you have any existing Developer sandboxes that were created before source tracking was enabled, enable Source Tracking for Sandboxes in the production org, then refresh those sandboxes before proceeding. For beta, however, after source tracking is enabled, don't refresh those sandboxes again because DevOps Center loses track of the source tracking state. Instead, use the synchronize development environments feature to keep dev environments in sync with the source control repository.

Did you...

- Create all the necessary source-tracked Developer or Developer Pro sandboxes you need for the project.
- Create sandboxes (as needed) for pipeline stages, for example, integration, user acceptance testing, and staging.
- Gather the usernames and passwords for all environments, including the final release environment, such as production.
- Add team members as users to every environment that they require access to.
- Update the IP allow list for the DevOps Center app (if your security settings require it)

See also:
DevOps Center Quick Start Guide: Synchronize Your Development Environment

What's Special About Development Environments?

Development environments require source tracking so they can automatically track changes as they are made. Use Developer or Developer Pro sandboxes created from your own production org.

Ideally, everyone contributing customizations for a DevOps Center project is assigned their own Developer sandbox. See your Salesforce Admin about allocating sandboxes for your team's participation in this beta.

See also:
Salesforce DX Developer Guide: Track Changes Between Your Project and Org
DevOps Center Quick Start Guide: Are You Sharing Your Development Environment?
DevOps Center Quick Start Guide: Synchronize Your Development Environment

Update IP Allow List for DevOps Center App

If your security settings require it, add these IP addresses for all environments that have allow listing enabled.

1. Add these addresses for the DevOps Center app:

   ```
   52.43.33.118
   44.230.180.227
   44.225.154.211
   100.21.209.24
   ```

2. Add the Salesforce IP address ranges for the org's region. See Salesforce IP Addresses and Domains to Allow.

See also:
Salesforce Security Guide: Restrict Login IP Ranges for a Profile
Salesforce Help: Set Trusted IP Ranges for Your Organization
Plan Your Pipeline

Open DevOps Center

From the App Launcher, find and select **DevOps Center**. DevOps Center opens to the Projects page. You've just logged in, so there aren't any projects yet.

Create a Project

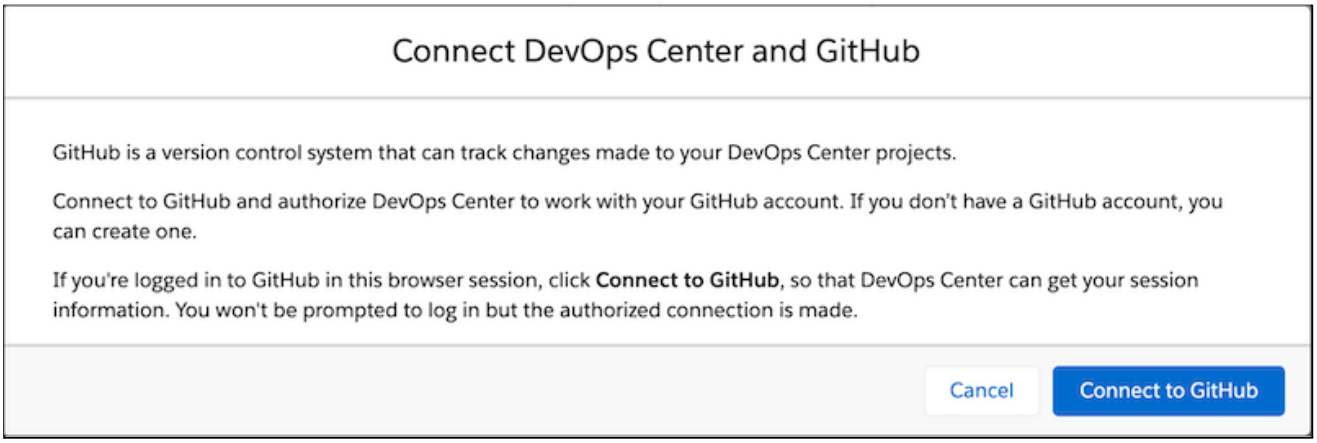
Your team's central arena for work in DevOps Center is the *project*. The purpose of a project is to help you and your team manage changes being developed for a particular application. A project encapsulates definitions and configurations of the many different things that managing a set of changes requires, including:

- Work items that define the changes to be made
- A pointer to the source control repository that stores changes made for the project
- Which work environments are used to make changes
- Environments used for pipelines stages, for example, integration, UAT, and staging
- A pipeline that defines how changes are deployed as they move from development to production

Projects in DevOps Center must use the Salesforce DX project structure, and be associated with a GitHub repository. The repository is used to store project changes.

1. From the DevOps Center Projects page, click **New Project**.

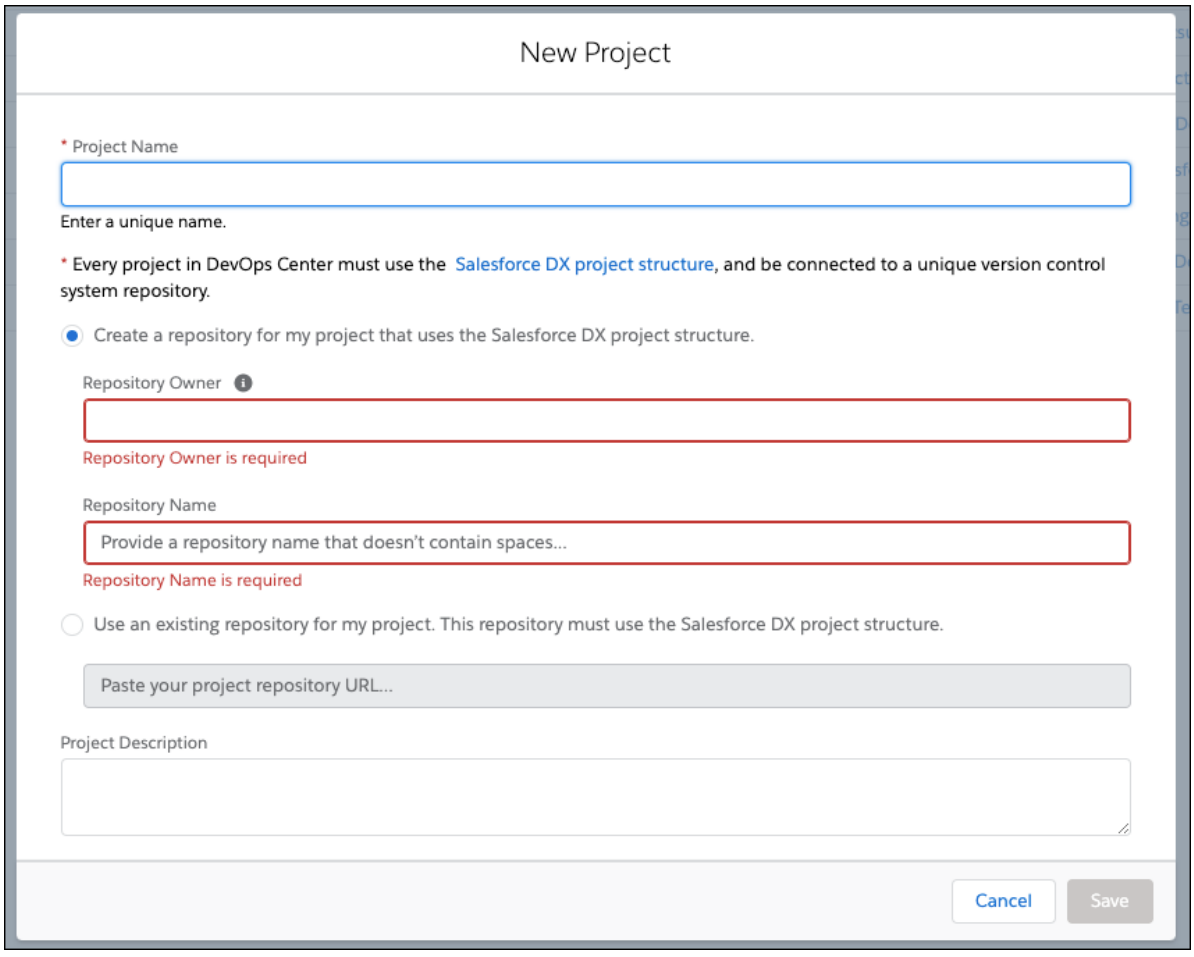
   The first time you create a project, you're prompted to log in to GitHub, so you can authorize DevOps Center to work with your GitHub account. After you go through the authentication process, DevOps Center can make changes on your behalf in your project's GitHub repository.



2. Click **Connect to GitHub**.

Is your GitHub repo owned by an individual or an organization?	How to proceed
Individual account	Authorize access so DevOps Center can make changes in GitHub on your behalf. After authentication, you're returned to the DevOps Center Projects page. Continue to step 3.
Organization account	In beta, GitHub repos owned by an organization aren't visible in DevOps Center without specifically providing access via OAuth. After you follow the instructions in If an Organization Owns the GitHub Repo, and the repo owner approves the request for access, you can create the project.

3. On the Projects page, click **New Project** (again).



4. Enter a unique name for your project.

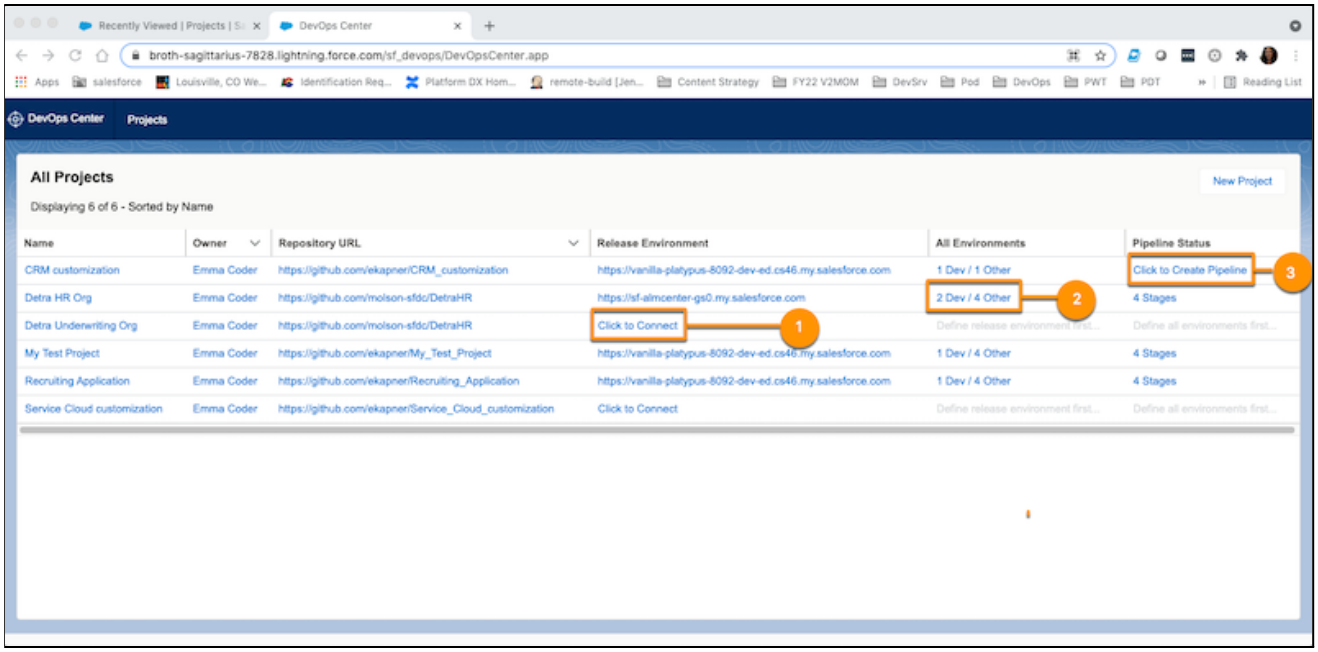
5. Create a repository or use an existing one.

    - If you create a repo, we base the repository name on the project name. However, you can change the name as long as it doesn't contain any spaces.
    - If you use an existing repository, enter the URL of the GitHub repository you want to use for the project. For example, https://github.com/mygithubusername/Myrepo.

6. (Optional) Enter a description to identify the purpose of the project.

7. Click **Save**.

Your project is created and added to the Projects page. The Projects list view helps guide you through the project configuration process. Different projects can be in different phases of project setup.



The project setup workflow is:

- Define and connect to the final release environment (1).
- Add development and other environments (2) needed for this project.
- Configure your pipeline (3).

DevOps Center provides the "hot link" only after you complete each step, so you don't accidentally jump ahead.

See also:
DevOps Center Quick Start Guide: Start on a Work item

If an Organization Owns the GitHub Repo

GitHub repos owned by an organization aren't visible in DevOps Center until an organization account owner provides access.

1. Authenticate to GitHub through DevOps Center.
2. In GitHub, provide access to both Salesforce Integration applications.

   One provides integration across your GitHub repo, DevOps Center, and development environments (scratch orgs and sandboxes). The other provides integration across your GitHub repo, DevOps Center. and production orgs.

   a. In your personal GitHub account, go to **Settings**.

   b. Click **Applications**, then select **Authorized OAuth Apps**.

   c. Click **Salesforce Integration Application**.

   d. Find the organization that owns your repo, then click **Request**.

   e. Repeat steps for the other Salesforce Integration Application.
3. After the GitHub repository owner approves your request for access, open the DevOps Center org.

   a. If you're in DevOps Center, click the Home icon to get to the org's home page.

   b. Click your profile icon, then select **Settings**.

   c. Click **Authentication Settings for External Systems**.

   d. Delete **DevOps Center GitHub**.
4. Start with step 1 again in Create a Project.

If the Organization Has an IP Allow List

If you have restricted access to the GitHub organization assets by configuring an allow list, add the IP addresses for the DevOps Center app.

1. Add these IP addresses to your GitHub organization's allow list:

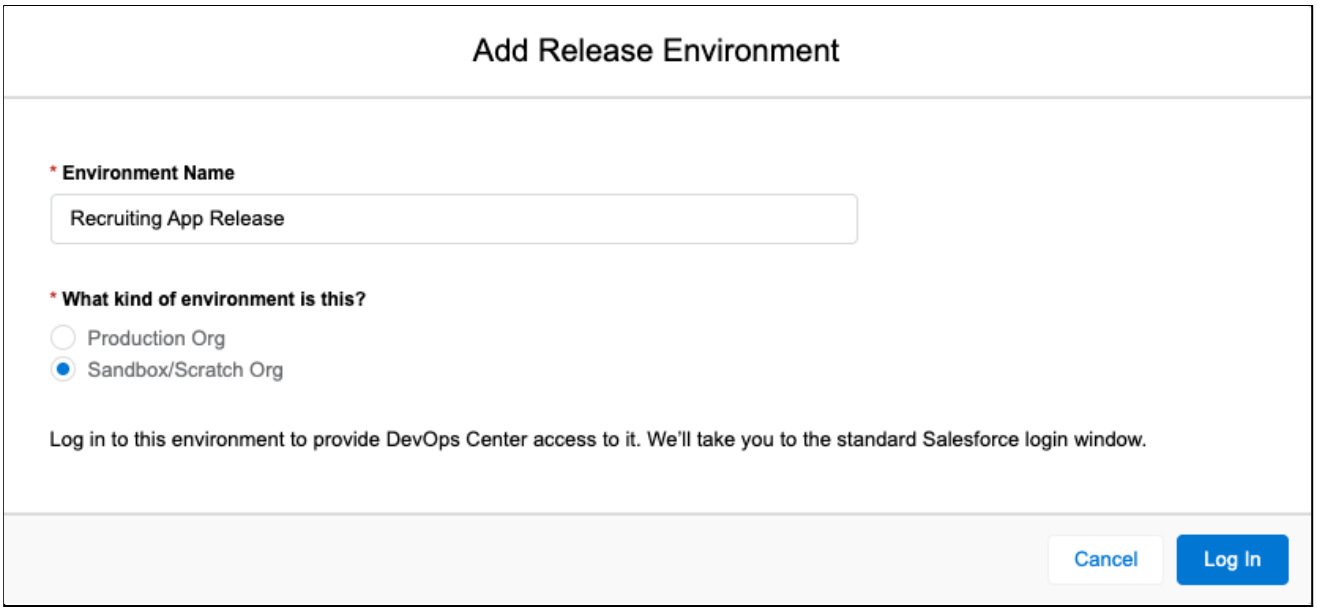
   ```
   52.43.33.118/32
   44.230.180.227/32
   44.225.154.211/32
   100.21.209.24/32
   ```
2. Add the DevOps Center Salesforce organization IP address ranges for your org's region. See Salesforce IP Addresses and Domains to Allow.

See also:
GitHub Docs: Managing allowed IP addresses for your organization

Connect to the Release Environment

At a minimum, projects require a release environment (often your production org) and one developer environment. Be sure all team members are users in this environment if they require access to it to perform deployments to it from within DevOps Center.

1. On the All Projects page, under Release Environment, click **Click to Connect**.
2. Provide a unique name for the production environment and indicate the environment type. By default, DevOps Center populates this field with the project name + Release.



3. Click **Log In**.
4. Log in to the environment.

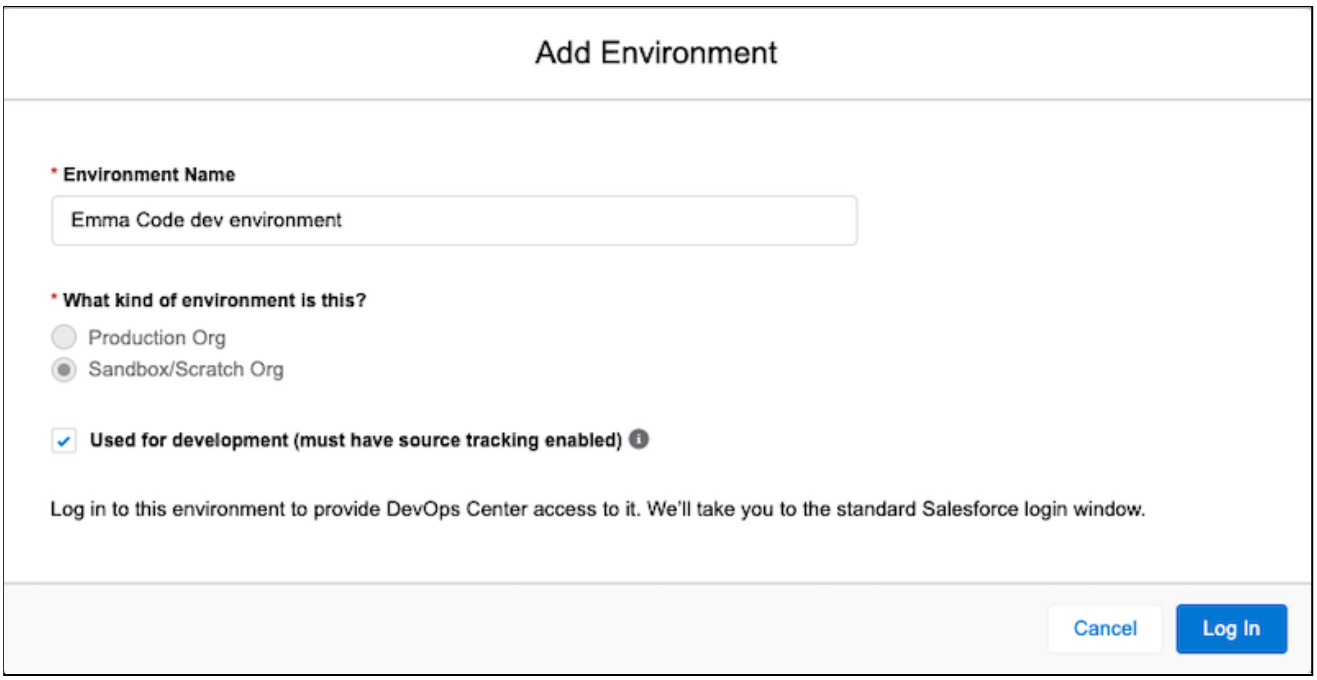
   After you authenticate successfully, allow DevOps Center to use an OAuth process to remember your credentials. The release environment URL is now listed on the Projects page. Now define the development environments for this project.

Add Development Environments

Add the source-tracked development environment that you created for each developer working on the project.

1. In the Project page, click the hot link text under All Environments.

2. In the Settings page, click **Add**.
3. Select **Sandbox/Scratch Org**.
4. Be sure to select the checkbox, **Used for Development**.



5. Click **Log In**.

   DevOps Center takes you to `test.salesforce.com` to log in. After you authenticate successfully, allow DevOps Center to use an OAuth process to remember your credentials.

6. Repeat steps 2–5 to add more development environments.

Now, when team members select or are assigned a work item, they can easily connect to their development environments from within the work item. Users are asked to authenticate to the environment with their own credentials to authorize their use from DevOps Center.

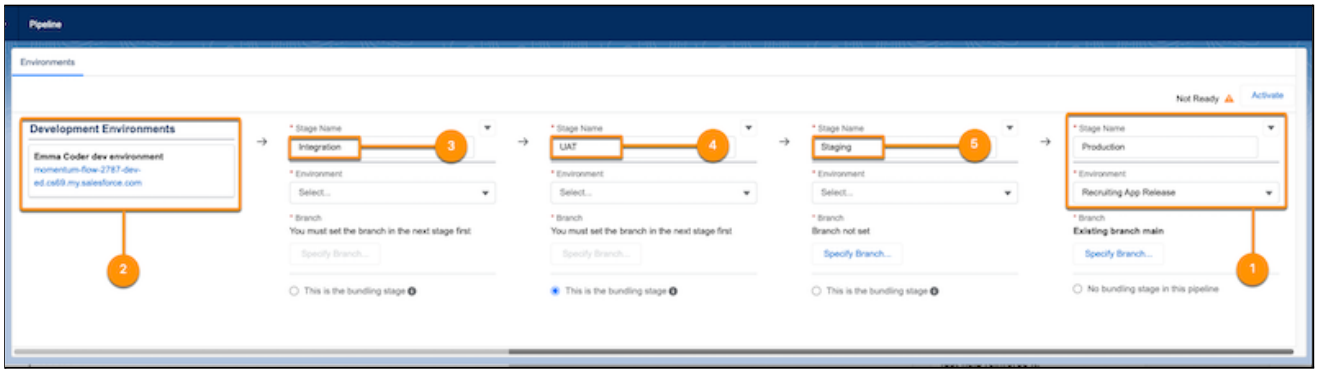
Next step: Determine your pipeline configuration (how many stages), then add the other environments associated with your pipeline stages.

Plan Your Pipeline

A *pipeline* defines the sequence of stages that work items progress as they go through the release lifecycle from development through to production (or some other final release stage). The pipeline consists of pipeline stages. Each pipeline stage corresponds to an environment (currently a Salesforce org), and a branch in the source control repository. The pipeline can't be modified after any changes have been promoted through it.

How Many Pipeline Stages Do I Need?

Your pipeline can contain any number of pipeline stages. At a minimum, a pipeline requires a *final release environment* (1) and *one development environment* (2). We recommend that you have at least one test stage besides your development and production stages.



The configuration of your pipeline is entirely up to you, and is based on the development and business processes that you have in place.

To assist you with building a robust pipeline, which typically includes 2–3 test stages, we provide a pipeline template. This pipeline template includes these recommended stages:

Integration
The first test stage (after the development stage) is an integration stage (3) where all changes from the various development environments come together for the first time and can be tested in an integrated environment. This stage is where you identify conflicts and resolve them before you move forward.

UAT
You can have one or more stages in between “integration” and “staging” (4) where you can perform additional testing, including by business stakeholders, often called user acceptance testing (UAT). We indicate this stage as the bundling stage, where you version a group of related changes and move them together through the pipeline.

Staging
This last test stage (before production) is used for final validation or “staging” (5) before you promote the changes to production.

Production (Release)
The final destination for your changes (1). The org for this stage is your live Salesforce instance.

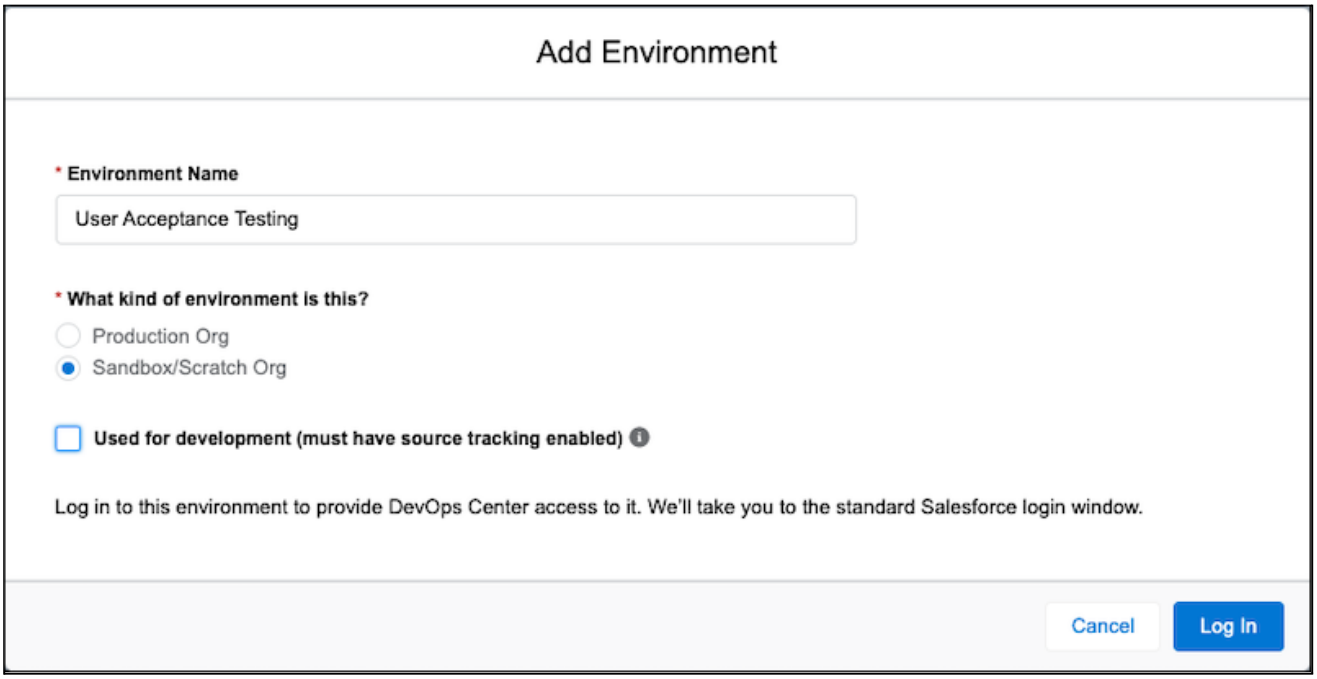
See also:
DevOps Center Quick Start Guide: Promote Work Items Through Your Pipeline
To Bundle or Not to Bundle, That’s a Great Question
Apex Developer Guide: Testing and Code Coverage

Add Other Pipeline Environments

After you determine the number of stages in your pipeline, you're ready to add the environments associated with each pipeline stage that occurs before the final release stage. The final release stage is often your production org, which is already connected. If team members require access to open the environments and perform deployments (promotions) to them using DevOps Center, be sure these team members are users in the environments.

Tip: You can add a new environment at any time. However, to expedite building your pipeline, we recommend defining all pipeline environments first before building the pipeline.

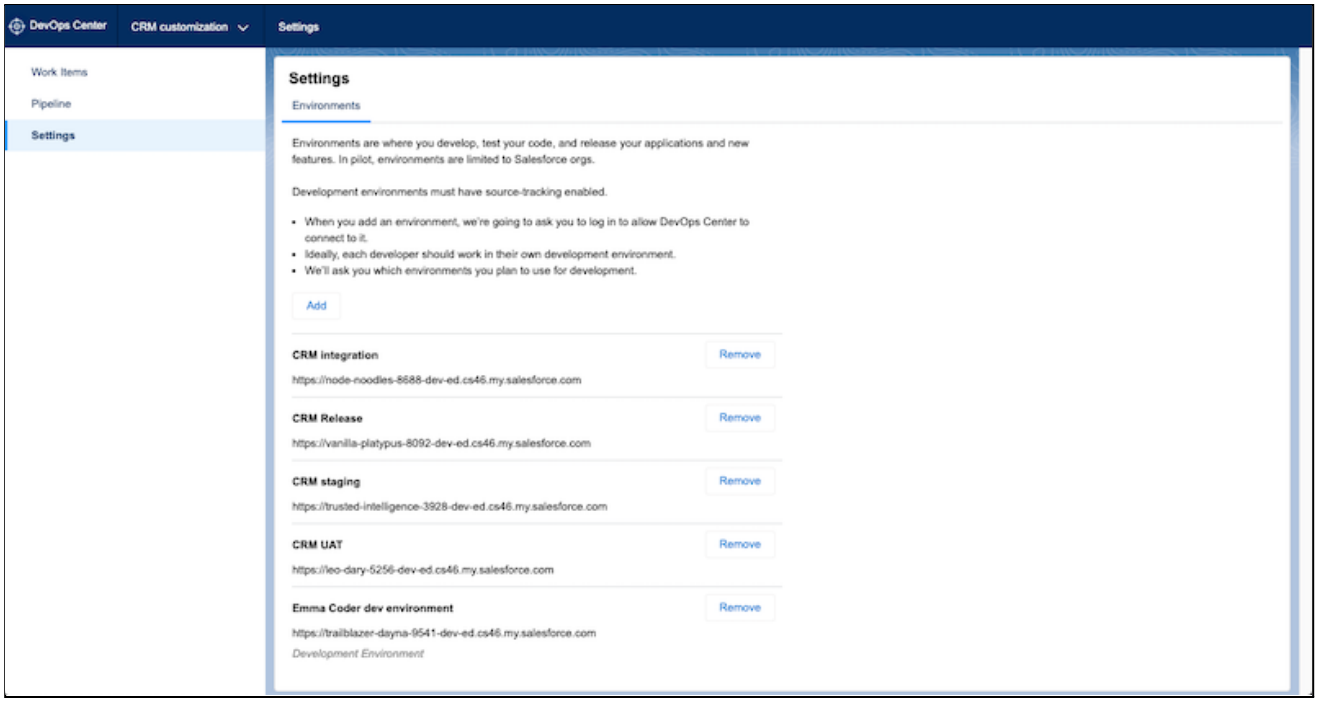
1. From the Setting page, click **Add**.



2. Select the environment type. Skip the **Used for development** checkbox.
3. Click **Log In**.

   DevOps Center takes you to the Salesforce login screen. After you authenticate successfully, allow DevOps Center to remember your credentials.
4. Repeat steps to add another pipeline environment.

After you've added all your pipeline environments, you see them listed. Now you're ready to build your pipeline.



Pipeline Configuration Options

Building and activating the pipeline are the last steps to complete configuration of your DevOps Center project.

Changes move through the pipeline when team members promote work items or work item bundles (a versioned group of changes that get promoted together). Upon promotion, changes are merged from the current stage branch (or feature branch) to the next stage branch, and then are deployed to the next stage org.

You can configure your pipeline in one of two ways:

- Allow team members to move work items individually through the entire pipeline.
- Allow team members to move work items individually in early stages of the pipeline, and as a versioned group of changes (work item bundle) in later stages. Keep reading to learn all about the benefits of work item bundles.

Build Your Pipeline

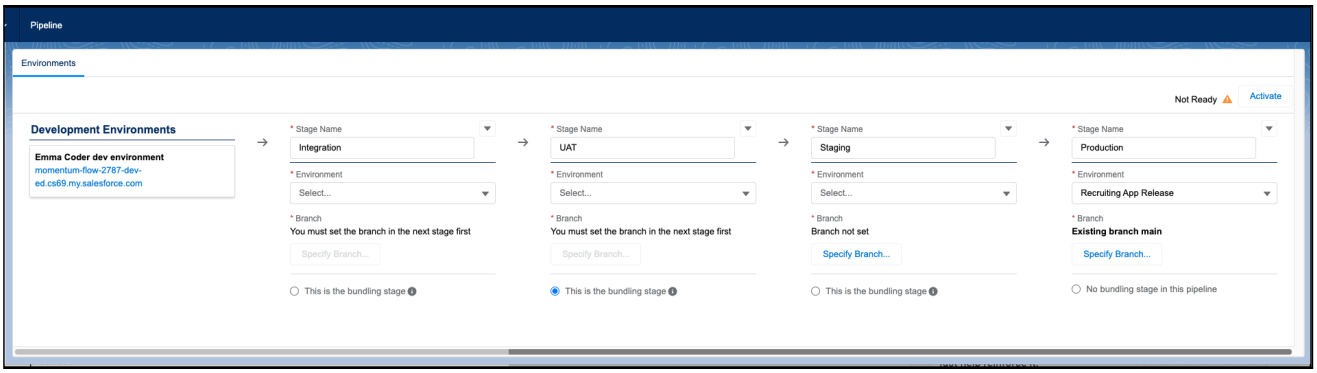
After you define development environments and a release (production) org, you have a basic pipeline. However, it's not recommended that you deploy directly to production. You can use our template to build the pipeline, you can add more stages to the pipeline template, or you can build your own.

While you're working on building the pipeline, your changes persist but don't get saved to the project until you activate the pipeline.

Prerequisite: Complete the creation of the environments associated with each pipeline stage.

1. Access the Pipeline page.
    - From within the project, click **Pipeline**.
    - From the All Projects page, click **Click to Create Pipeline**.

    In the Pipeline page, you see the pre-defined pipeline template.



2. Before you begin, it's important to understand how to properly specify the branch for each pipeline stage. We recommend that you don't share branches across projects.

3. Complete your pipeline configuration using the preferred method.
    - If you want to use the template to build your pipeline, see Build Your Pipeline Using the Template.
    - If you want to build your own pipeline configuration, skip to Build Your Own Pipeline.

What You Need to Know About Branches

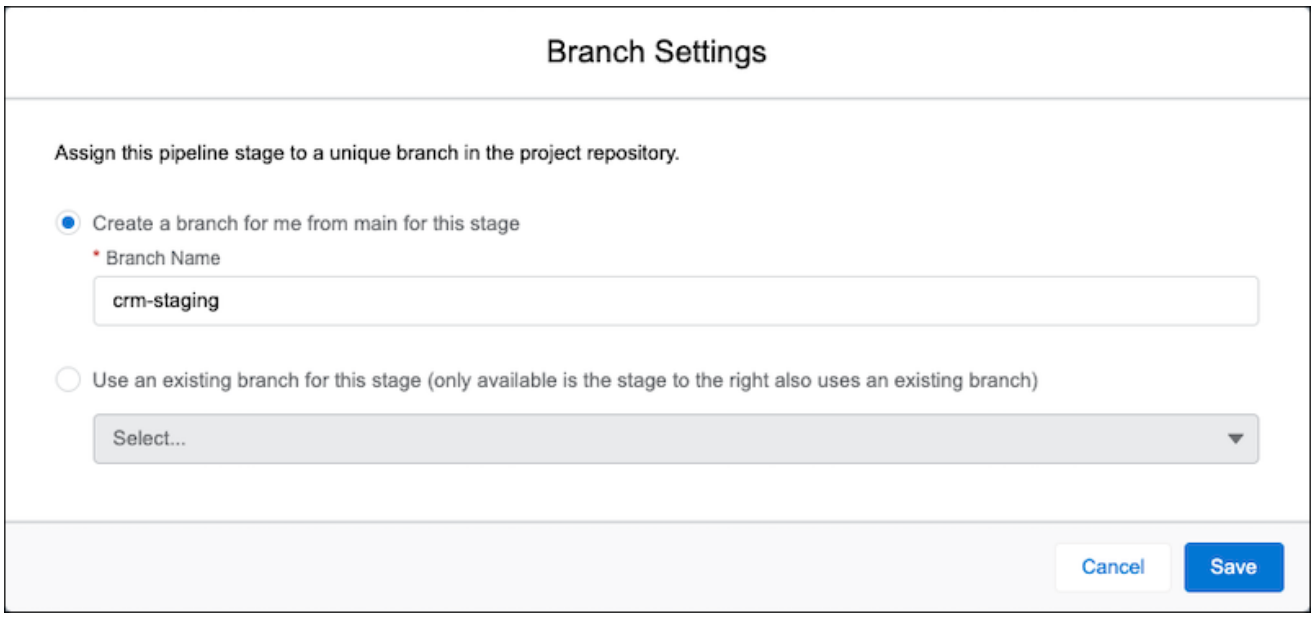
You can specify either an existing branch in the repository, or allow DevOps Center to create one for you. If you allow DevOps Center to create it for you, indicate a unique alphanumeric string. We also recommend that branch names use:

- All lowercase letters
- 60 characters or fewer
- Hyphens or underscores as separators (no spaces)
- Both letters and numbers, but not all numbers

Important: We recommend that you don't share branches across projects.

The branch must be sourced from the next stage branch (right to left). For example, let's use our template pipeline structure to clarify what we mean. If your release environment's branch is `main`, the branch for the pipeline stage to the left of it, `staging`, must be created from `main`. The branch for the pipeline stage to the left of Staging, called `uat`, must be created from the `staging`.

We handle this process for you if DevOps Center creates the branch.

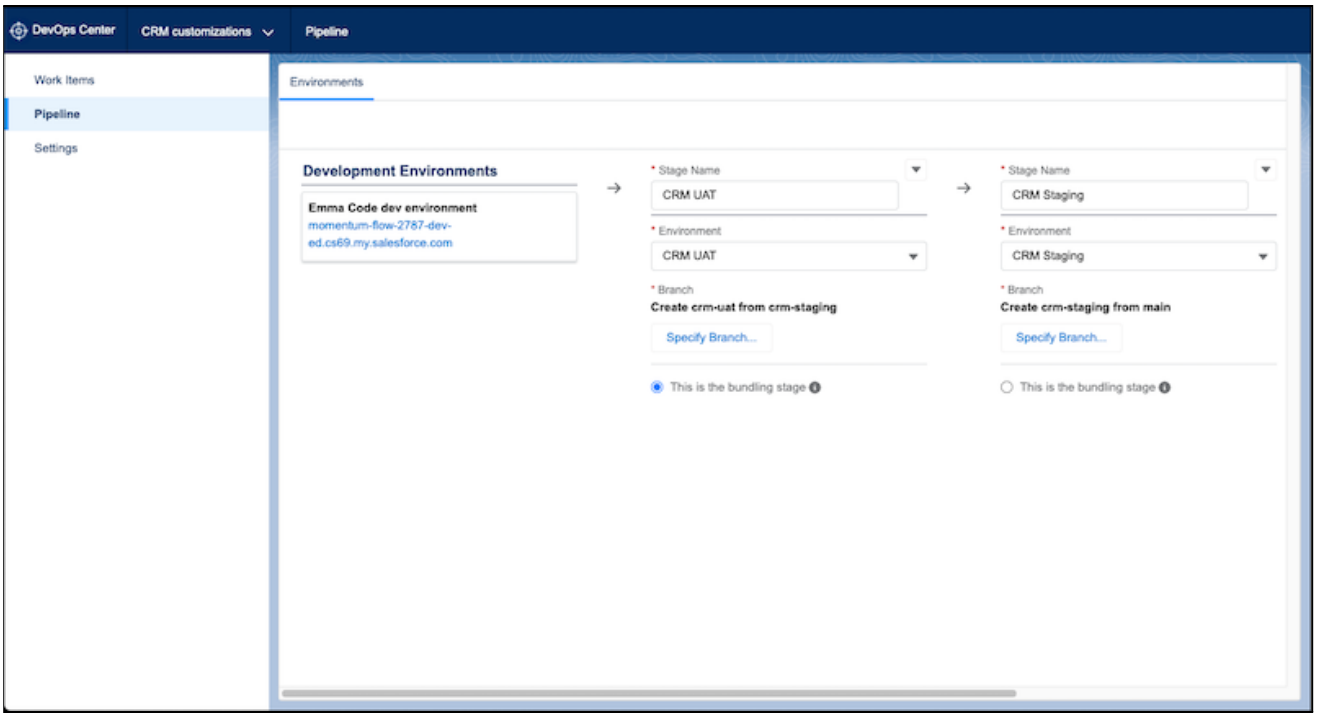


Build Your Pipeline Using the Template

When you specified the release target (production) environment, it automatically became the last pipeline stage. DevOps Center also associates it with the default branch in the source control repository, which is `main` in GitHub. We don't recommend that you edit the branch for this final pipeline stage.

1. From the Pipeline page, start with the Staging stage.
2. Select the environment associated with this stage. If you don't see the environment in the dropdown list, go to Settings and add it there first.
3. Next, specify the branch associated with this stage.
4. Indicate which stage is the bundling stage, the stage where changes are grouped together, versioned, and promoted together in downstream stages.

   In the template, the UAT stage is the bundling stage. The items you promote from integration are released together in a work item bundle in the next stage, UAT, and for all future stages.



See To Bundle or Not to Bundle, That's a Great Question for why we recommended a bundling stage.

5. Repeat this process for the rest of the stages.
6. Activate the pipeline.

Build Your Own Pipeline

You can quickly build your own pipeline by modifying the existing template pipeline.

- From the Stage dropdown, move stages left or right, or remove a stage.
- Change the stage name.

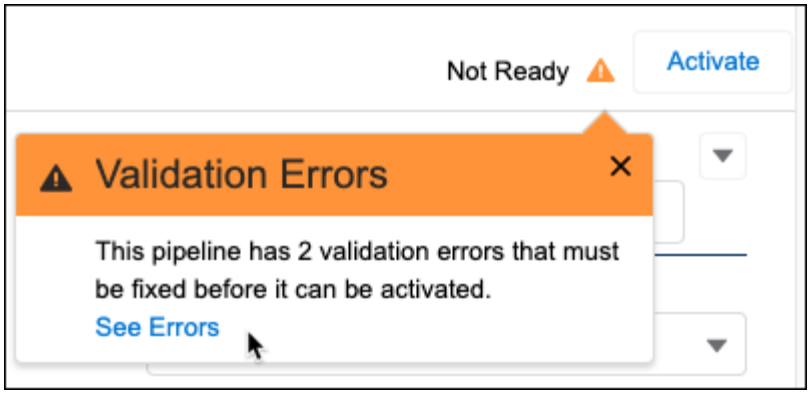
1. Starting with the last stage before your release stage, select the environment associated with this stage. If you don't see the environment you want to select, go to Settings and add it there first.
2. Next, specify the branch associated with this stage.
3. Repeat for the rest of the stages.
4. Indicate which stage is the bundling stage, the stage where changes are grouped together, versioned, and promoted together in downstream stages. See To Bundle or Not to Bundle, That's a Great Question for why we recommended a bundling stage.
5. Activate the pipeline.

Activate The Pipeline

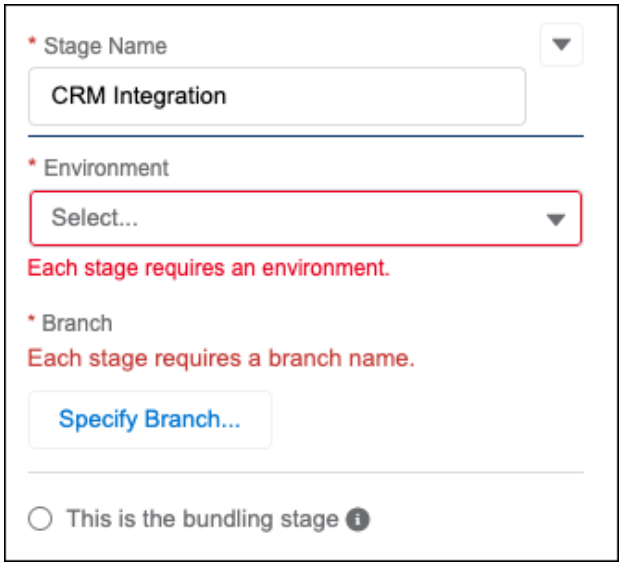
A pipeline is ready for your team to use after you activate it.

- You and your team can create work items after a pipeline is activated.
- You can't inactivate a pipeline after changes have been promoted through it.
- You can edit only inactive pipelines. If you haven't yet promoted changes, you can deactivate it, make the changes, and then reactivate it.

You can activate the pipeline if the status is `Ready to activate`. If there are validation errors with the pipeline, a `Not Ready` warning appears next to the Activate button with a link to see the errors to resolve.

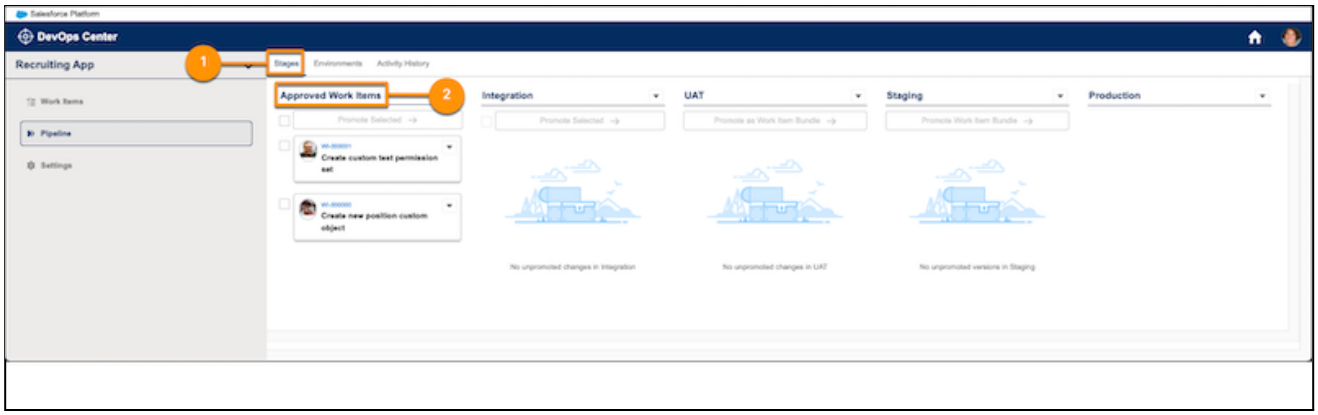


The errors appear in context, so you know exactly what to address.



What's The Difference Between the Stages and Environments Tabs?

After the pipeline is activated, the Stages tab (1) appears as a first tab on the Pipeline page.



The Stages tab is where you promote work items through the pipeline. Work items appear in the Approved Work Items column (2) after they have been marked `Ready to Promote`. See Create and Assign Project Work Items for more information.

In the Environments tab, you can:

- Continue to configure the pipeline stages, if the pipeline is not active.
- Deactivate your pipeline (1), if you haven't promoted any work items. After a work item has been promoted, you can no longer deactivate it.
- View the version of the branch and the number of work items not yet deployed to the stage's environment. In the illustration, the version for each pipeline is "na" because nothing has been promoted yet.

After you activate the pipeline, in the Environments tab, you can:

- Check to see if your development environments are in sync with the next pipeline stage, and initiate a synchronization process if they aren't (2).
- See the configuration of each pipeline stage (3), including the environment and branch associated with the stage, and the latest bundle version installed to the stage.
- Access environments directly with a click (4).

To Bundle or Not to Bundle, That's a Great Question

The process and mechanism for promoting changes from one stage to the next can vary as you move from "left to right" in the pipeline. In the earlier (left) stages of the pipeline, you often want more flexibility to promote individual work items from one stage to the next. As you move to the later (right) stages of the pipeline, it's often desirable to have more predictability and ability to version the sets of changes that are promoted and ultimately released.

DevOps Center allows you to define what this overall model of promotion looks like. The point in the pipeline where you transition from the more flexible/individually-selectable work item promotion to the more predictable/versioned promotion is referred to as the *bundling stage*. The bundling stage is where changes come together to be bundled into a *work item bundle* that can be versioned and promoted as a unit through the subsequent stages. When changes are promoted from the bundling stage to the next stage, all work items that have not yet been promoted are included in the versioned work item bundle and promoted as a unit. This versioned bundle continues to be promoted as a consistent unit through subsequent stages when you perform a promotion.

The bundling stage is defined when you configure the pipeline. All stages to the left of this stage allow for individual work item promotion, and all stages to the right of this stage allow for versioned work item bundle promotion.

Work Item Bundles Reduce Merge Conflicts

In the stages to the left of the bundling stage, you have increased flexibility regarding which work items can be promoted and when. However, this flexibility comes with the tradeoff of increased risk of conflicts and unanticipated behavior because the combination of changes in each stage can be inconsistent. The versioned work item bundle that is created in the bundling stage and promoted through subsequent stages provides improved consistency because the changes contained in it have been merged into a unit that you can promote consistently and predictably from stage to stage.

See also:
DevOps Center Quick Start Guide: Conflict Detection and Resolution

Set Up Team Members in the DevOps Center Org

When your project is connected to its GitHub repository and you've set up the environments and pipeline, it's time to start adding the people on your project team to DevOps Center. Add team members as collaborators in your GitHub repository, create user accounts for them in the DevOps Center org, and assign them the permission sets they need to work in DevOps Center.

Add Team Members as Repository Collaborators

In GitHub, a *collaborator* is someone who has been granted write access to the project repository. Add everyone who creates customizations or code for your project to the project repository as a collaborator, so they have access to the repository through DevOps Center.

To add a team member as a collaborator in the project repository:

1. Ask each of your team members to create a GitHub account (if they don't have one already) and to send you their GitHub username.
2. Log in to GitHub and update the project repository settings for access to invite the team member as a collaborator.

   GitHub sends an email to each team member you invited, asking them to accept the invitation.

   Tip: Follow up with your team members to make sure they accept the invitation from GitHub to avoid access problems when the team members begin using DevOps Center.

Create and Assign Project Work Items

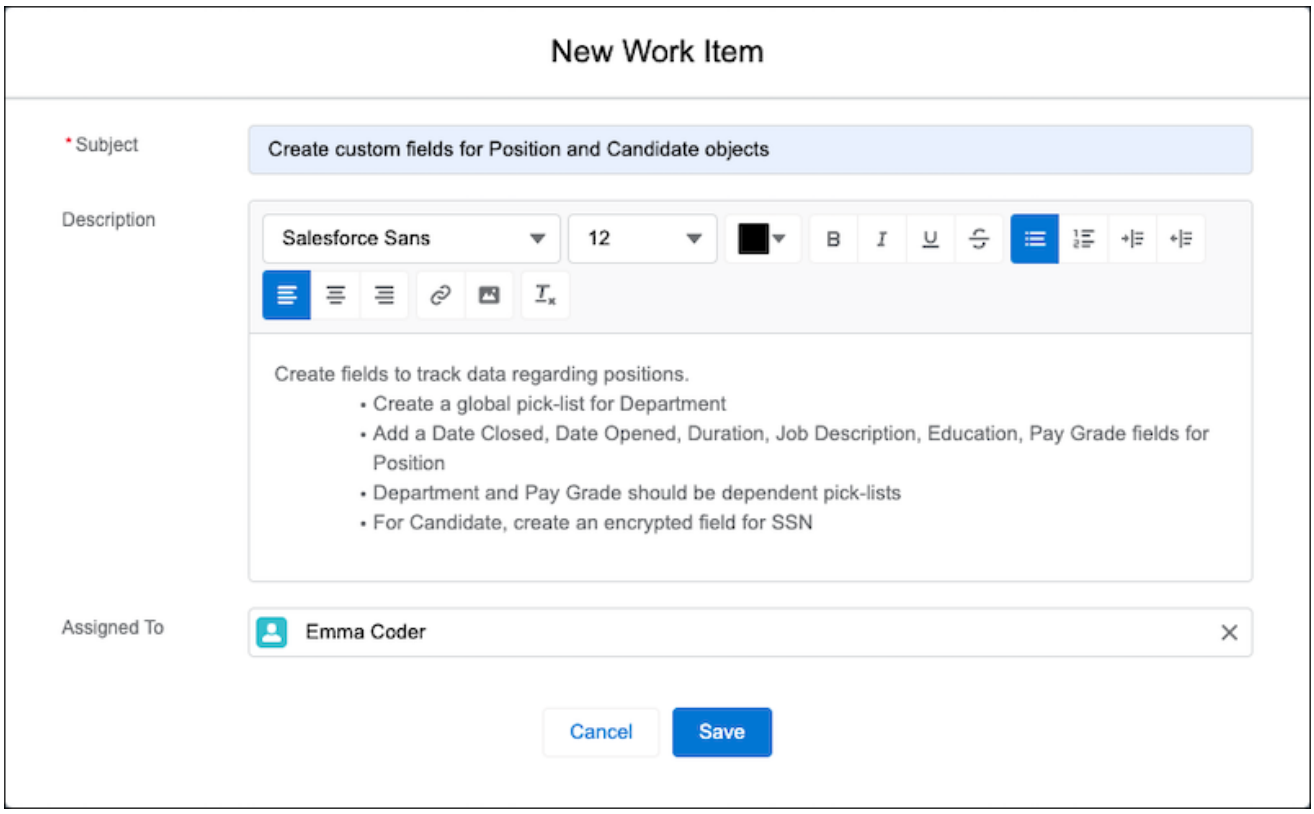
Create work items so that when your team members open DevOps Center for the first time, project work is already identified and assigned to them.

In DevOps Center, a team uses work items to track the progress of changes created to achieve a specific objective, such as enabling a user story or addressing a bug. Work items help a team manage a release by making it easier to identify the status and manage the progress of related changes.

1. From the Projects page, click the name of the project for which you're creating work items.

Tip: Did you already configure your pipeline? If not, DevOps Center walks you through specifying a release environment first before you can continue.

2. From the Work Items tab, click **New Work Item**.
3. Specify the objective or the problem to be addressed in the Subject field.
4. If more details would be helpful to the assignee, use the Description field to provide additional information. We use the first 255 characters of the description to help identify changes for this work item in GitHub.
5. (Optional) Assign the work item to a team member.



6. Click **Save**.

   The work item is displayed in the Work Items tab.

7. Repeat this procedure as needed to track and assign project work. Both you and your team members can create additional work items as the project progresses.

See also:
DevOps Center Quick Start Guide: Start on a Work Item

Next: Bring Team Members into DevOps Center

Now that you've set up a DevOps Center project and set up your team members to work in DevOps Center, it's time to bring your team members into the DevOps Center org. Refer them to the *DevOps Center Quick Start Guide,* where they can learn how to use the main features of DevOps Center, including how to manage project work items, review changes they've made in a work environment, commit those changes to the project repository, and promote the work items through the pipeline.

See also:
DevOps Center Quick Start Guide: Get Set Up with GitHub and the Repository
DevOps Center Quick Start Guide: Basic Workflow

Uninstall DevOps Center

You can uninstall DevOps Center, if necessary. It's important to follow these tasks in order to properly uninstall the DevOps Center package. You'll need to remove some configuration dependencies before you can successfully uninstall the package. Uninstalling DevOps Center also removes all associated data including, but not limited to, work items, pipeline configuration, and activity history.

Remove DevOps Center Permission Set Assignments

First, remove these DevOps Center permission set assignments from all DevOps Center users.

- DevOps Center
- DevOps Center Manager
- DevOps Center Release Manager
- sf_devops_InitializeEnvironments
- sf_devops_NamedCredentials

See also:
Salesforce Security Guide: Remove User Assignments from a Permission Set

Delete DevOps Center Permission Sets

Next, delete these permission sets.

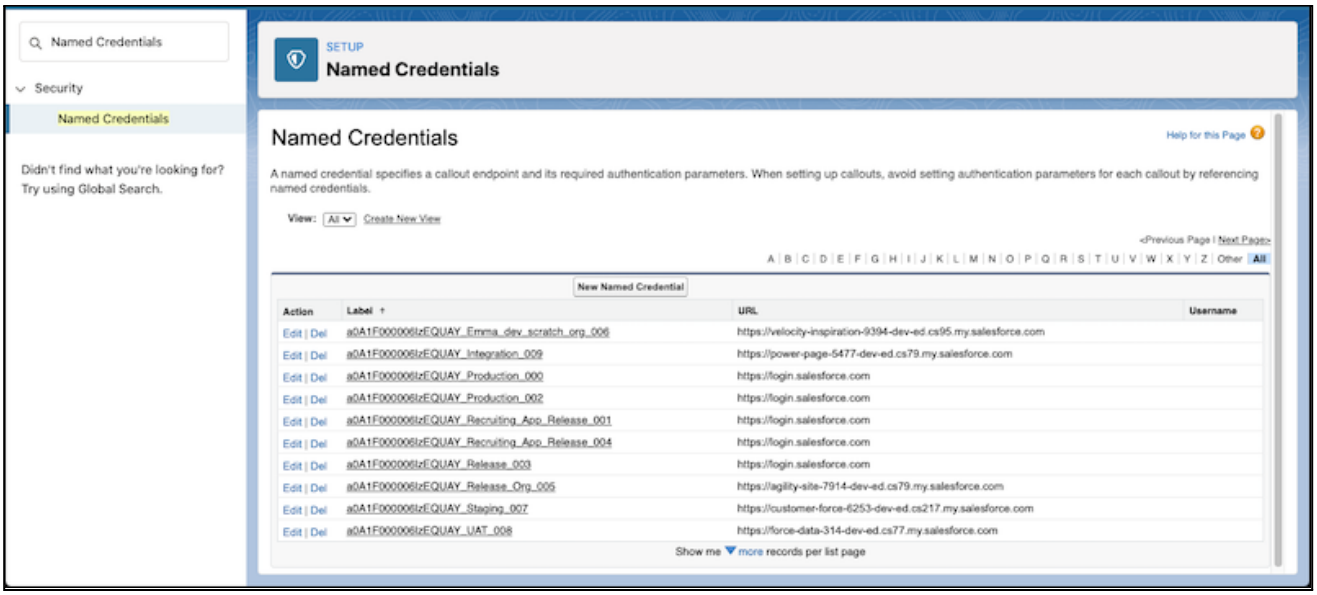
1. From Setup, enter `Permission Sets` in the Quick Find box, then select **Permission Sets**.
2. Delete these two permission sets:
    - sf_devops_InitializeEnvironments
    - sf_devops_NamedCredentials

Uninstall the DevOps Center Package

1. From Setup, enter `Installed Packages` in the Quick Find box, then select **Installed Packages**.
2. Click **Uninstall**, then scroll to the bottom of the page to choose whether to save and export a copy of the package's data.
3. Select **Yes, I want to uninstall this package and permanently delete all associated components**.
4. Click **Uninstall**.

Delete DevOps Center Named Credentials

1. From Setup, enter `Named Credentials` in the Quick Find box, then select **Named Credentials**.

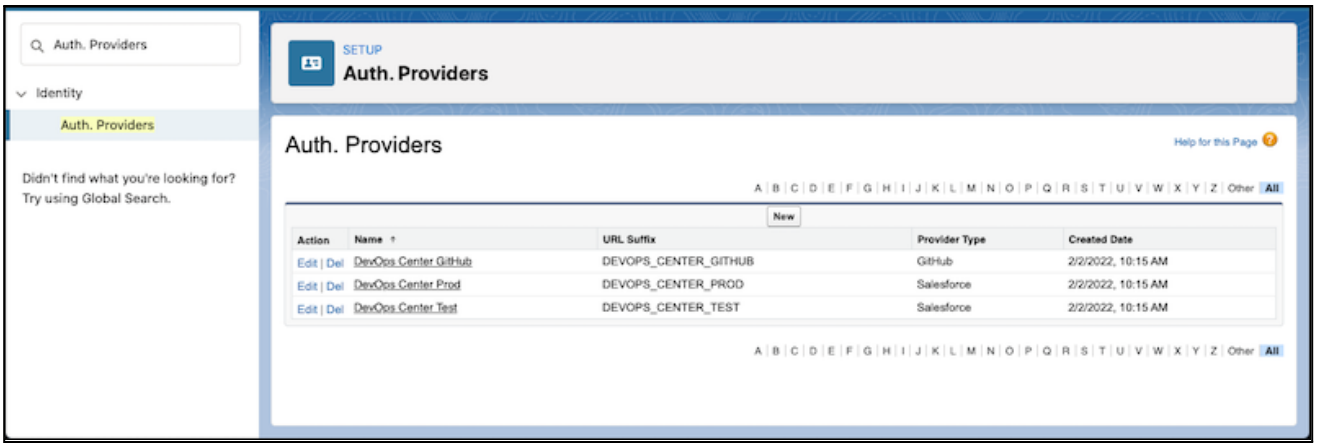


2. Delete any named credentials associated with DevOps Center environments. Look for the format `<ID>_<environment-name>_<number>`.

Delete DevOps Center Authentication Providers

Warning: Only delete the DevOps Center auth providers if you're done with your evaluation of the beta and want to remove all installation artifacts from your org. If you plan to reinstall DevOps Center, DO NOT delete the DevOps Center auth providers and skip this section.

1. From Setup, enter `Auth. Providers` in the Quick Find box, then select **Auth. Providers**.



2. Delete the DevOps center authentication providers.

Delete DevOps Center Connected App

1. From Setup, enter App `Manager` in the Quick Find box, then select **App Manager**.
2. From the drop-down menu for DevOps Center, select **View**.
3. Click **Delete**.
4. Confirm that you want to delete the connected app.

Troubleshooting

Here are some tips if you encounter these errors.

Error: "(DevOps_Center_GitHub) In field: authProvider - no AuthProvider named DEVOPS_CENTER_GITHUB found DevOps_Center_GitHub: In field: authProvider - no AuthProvider named DEVOPS_CENTER_GITHUB found"

Cause: It's highly likely that we need to enable DevOps Center in your org.

Action: Open an issue in the repository or send an email to: `sf-devopscenter-preview@salesforce.com`. Please contact us – don't attempt to create the auth providers manually.

Error: "The request was invalid. Resource protected by organization SAML enforcement."

Cause: GitHub repos owned by an organization aren't visible in DevOps Center until an organization account owner provides access.

Action: See If an Organization Owns the GitHub Repo for instructions on working around this issue.

Known Issues

To provide you with timely information on how to work around any known issues, we've documented them in the DevOps Center Feedback repo. See KNOWNISSUES.md.