



Quick Start Guide

DevOps Center

Beta, Spring '22



@salesforcedocs

Last updated: April 6, 2022

© Copyright 2000–2022 salesforce.com, inc. All rights reserved. Salesforce is a registered trademark of salesforce.com, 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](#)

[DevOps Center Basics](#)

[DevOps Center Projects](#)

[One Collaboration Tool to Support a Variety of Roles](#)

[What's New in Beta?](#)

[GitHub/DevOps Center Interoperability \(Hybrid Team Flows\)](#)

[Development Environment Synchronization](#)

[Manually Add Any Metadata](#)

[Never Status for Work Items](#)

[Activity History](#)

[Work Item List View Sorting](#)

[Improved Navigation Through Our Application Header](#)

[Get Set Up with GitHub and the Repository](#)

[Basic Workflow](#)

[Open DevOps Center](#)

[Open a Project Work Item](#)

[Start on a Work Item](#)

[View and Pull Changes from the Development Environment](#)

[View Changes](#)

[Pull Changes from the Development Environment](#)

[Commit Any Metadata Component Manually](#)

[Commit Changes to the Project Repository](#)

[View Work Item Events in Activity History](#)

[Change Work Item Status to Never](#)

[Synchronize Your Development Environment](#)

[Are You Sharing a Development Environment?](#)

[Share Your Changes for Team Member Review](#)

[Review Changes in GitHub](#)

[Indicate That Work Items Are Ready to Promote](#)

[Promote Work Items Through Your Pipeline](#)

[GitHub Mergeability](#)

[Promotion Options](#)

[Apex Test Options](#)

[Promote Individual Work Items](#)
[Promote as Work Item Bundle](#)
[Promote Work Item Bundles](#)
[Promote Changes Merged Outside of DevOps Center](#)

[View Promotion Status](#)

[View Deployment Status to Dig Deeper](#)

[View Pipeline Events in Activity History](#)

[Conflict Detection and Resolution](#)

[Resolve Merge Conflicts in GitHub](#)

[If the Promotion Fails](#)

[Review and Address Conflicts in GitHub](#)

[Manage Work Items](#)

[Work Item Statuses](#)

[Create Work Items](#)

[Edit Work Item Details](#)

[Troubleshooting](#)

[Error During Pull: "There was an error. Please reload the page or contact your system administrator if the problem persists."](#)

[Error: "The request was invalid. Resource protected by organization SAML enforcement."](#)

[Error: "The request was invalid. URL does not reference a valid SFDX project. projectId"](#)

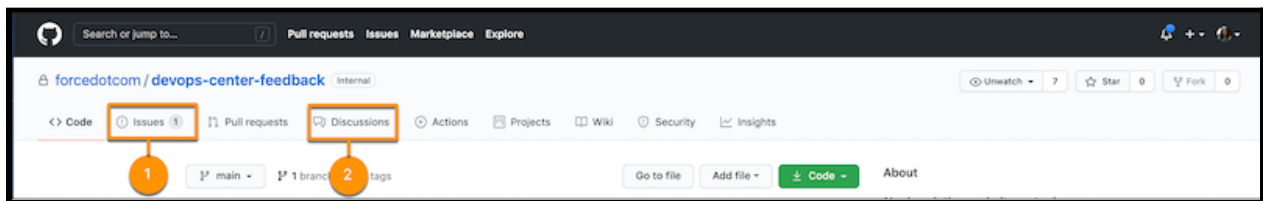
[Known Issues](#)

Welcome to 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. Reach out to your team manager if you don't have access to this repo or fill out [this form](#).

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 beta participants and members of the Salesforce DevOps Center product team.



See also:

DevOps Center Feedback repo: [Known Issues](#)

DevOps Center Basics

Your team manager has set up DevOps Center according to the steps in the *DevOps Center Setup Guide*. We assume that you've been notified by the team manager that the DevOps Center org is ready for you to log in.

DevOps Center Projects

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 collaborate and 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 pipeline stages, for example, integration, UAT, and staging
- A pipeline that defines how changes are deployed as they move from development to production

One Collaboration Tool to Support a Variety of Roles

Working on Salesforce customizations requires a variety of tasks by teams large and small. The degree of role specialization varies, and team members frequently have more than one role over the course of a release. The team roles we recommend include:

- Team Manager/ Project Manager
- Org Admin
- Low-code Developer
- Pro-code Developer
- Release Manager
- Business Owner
- Environments Manager
- Quality Assurance Specialist

To ensure that team members have access to the features they require to do their work, we provide several DevOps Center permission sets. Your Salesforce admin (or team manager) can use these perm sets to assign you appropriate permissions to access DevOps Center functionality.

If you perform multiple roles and don't have access to functionality required to do your job, talk to your team manager.

For more information about DevOps Center permission sets, see the *DevOps Center Setup Guide*.

What's New in Beta?

We added these new features and usability improvements in the beta release beyond what was previously provided in the pilot.

GitHub/DevOps Center Interoperability (Hybrid Team Flows)

We support more interactions both inside and outside of DevOps Center for team members to review and merge changes directly in the source control system. The following hybrid team use cases are supported with this beta release.

- A developer *merges* a pull request directly in GitHub. When the pull request is merged, it's effectively "half-promoted" because it has been merged but not yet deployed into the org associated with that repository branch. DevOps Center indicates this state and guides the user to complete the promotion in DevOps Center.
- Mergeability, as determined by GitHub, is honored by DevOps Center.
 - You can indicate that a work item is "Ready to Promote" only when the changes are deemed mergeable. Mergeability is determined by GitHub and is based on either merge conflicts or custom [Mergeability Rules in GitHub](#). If GitHub identifies merge conflicts with the changes, then the changes can't be promoted until the conflicts are resolved.

- Similarly, if your repo uses custom mergeability rules that haven't been met for the changes, then the changes can't be promoted until those rules are met.

Use cases not yet supported in beta:

- A developer *commits* changes to the GitHub repository from outside DevOps Center, and those changes are reflected in DevOps Center. This use case would allow a developer to contribute changes to the project, from entirely outside of DevOps Center.
- A user or CI/CD system *deploys* changes from outside of DevOps Center and the deployment is appropriately reflected in DevOps Center.

See also:

[Promote a Work Item Merged Outside of DevOps Center](#)

Development Environment Synchronization

DevOps Center makes it easier to keep your development environments up to date with the latest source of truth. DevOps Center tracks the differences between each development environment and the first integrated stage of the pipeline. This synchronization process ensures that you're developing against the latest integrated source of truth, including changes from other teammates.

Keeping your development environment synchronized is a best practice for avoiding merge errors and code conflicts, and preventing downstream release headaches.

See also:

[Synchronize Your Development Environment](#)

Manually Add Any Metadata

You can now select and add any metadata source files to your commit, not just the ones that DevOps Center determines have changed.

Known Limitation: We support standard metadata types, not decomposed Salesforce DX source format. To learn more about DX source format, see [Salesforce DX Project Structure and Source Format](#) in the *Salesforce DX Developer Guide*.

See also:

[Commit Any Metadata Component Manually](#)

Never Status for Work Items

You can now change a work item status to Never, which is useful in the following situations:

- You have a work item that you no longer plan to implement.
- You attempted to promote a work item and it failed due to a “bad” metadata file in the work item. You want to remove the “bad” file from the work item.
- You want to basically start over on a work item.

Not yet supported in beta: You can’t revert a commit to correct an issue on the original work item. Right now, your only option is to abandon the work item by changing its status to Never, and then create another work item.

See also:

[Change Work Item Status to Never](#)

Activity History

DevOps Center now shows a comprehensive history of all key events, including promotions, commits, synchronizations, work item status changes, and errors (failures). Activity History is available on each Work Item and on the pipeline. This feature allows you to maintain and view a record of events and associated details for auditing, trouble-shooting, and general visibility purposes.

Work Item List View Sorting

You can now sort the Work Item list view by any of the visible columns.

Improved Navigation Through Our Application Header

DevOps Center contains a header that provides access to the org’s default home page and active user information. You can easily navigate out of the DevOps Center application to the rest of your org, and have visibility into the active logged-in user with the ability to log out.

Get Set Up with GitHub and the Repository

You *must* have a cloud-based, GitHub-hosted, github.com account to use DevOps Center. You can create a new (free) account or use an existing account. We aren’t supporting any kind of on-premise, enterprise, locally hosted GitHub, or any other Git-based or other source control provider system in this release.

1. 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](#).
2. Send your GitHub username to the team manager or Salesforce admin who’s setting up DevOps Center so they can add you as a collaborator in the project repository.

3. Accept the invitation to collaborate in the GitHub repository.

If you haven't received an email invitation, contact the person who set up DevOps Center.

If you're new to GitHub concepts, we recommend working through the [Git and GitHub Basics](#) module and the [Work with the GitHub Workflow](#) unit in Trailhead.

Basic Workflow

1. [Open DevOps Center.](#)
2. [Start on a work item.](#)
3. [Pull changes from the development environment.](#)
4. [Commit changes to the project repository.](#)
5. [Share your changes for team member review.](#)
6. [Approve work items.](#)
7. [Promote work items](#)

Open DevOps Center

1. Log in to the org where DevOps Center is installed.

Contact your team manager for org information or user credentials if you don't have that information.

2. From the App Launcher, find and select **DevOps Center**.

DevOps Center opens to the Projects page, which lists current projects. Your team manager created at least one project as part of the setup process.

Open a Project Work Item

A team uses *work items* to track the progress of changes created to achieve a specific objective or task, such as enabling a user story, creating a feature, or addressing a bug. Managing releases through work items makes it easier to track the progress of the overall release and identify areas of concern.

When you open a project for the first time, you could see one or more work items assigned to you.

Before you start: Be sure you've accepted the emailed invitation from GitHub to collaborate. You can also log in to GitHub, open the repository, and accept the invite from there.

1. From the Projects page, click the name of the project you're working on.

The project opens to its Work Items page. If your team manager created work items, you see them listed here. Both you and other team members can create additional work items as the project progresses.

ID	Subject	Assigned To	Status	Date Created
WI-000024	Create a Position custom object and tab	Emma Coder	In Progress	4/16/2021, 04:53 PM
WI-000025	Create custom fields for Position and Candidate objects		New	4/16/2021, 04:55 PM
WI-000026	Link to external site for job posting records		New	4/16/2021, 04:58 PM
WI-000027	Tech debt cleanup	Emma Coder	In Progress	4/19/2021, 08:46 AM

2. Click the ID of a work item.

- If you're currently logged in to GitHub, DevOps Center opens the work item. Skip to [Start on a Work Item](#).
- If you're not currently logged in to GitHub, log in to GitHub and authorize access to DevOps Center. Click **Connect to GitHub**.

Connect DevOps Center and GitHub

GitHub is a version control system that can track changes made to your DevOps Center projects.

Connect to GitHub and authorize DevOps Center to work with your GitHub account. If you don't have a GitHub account, you can create one.

If you're logged in to GitHub in this browser session, click **Connect to GitHub**, so that DevOps Center can get your session information. You won't be prompted to log in but the authorized connection is made.


CancelConnect to GitHub

If the authorization is successful, you're returned to the work item. DevOps Center can make changes on your behalf in the project's GitHub repository.

Start on a Work Item

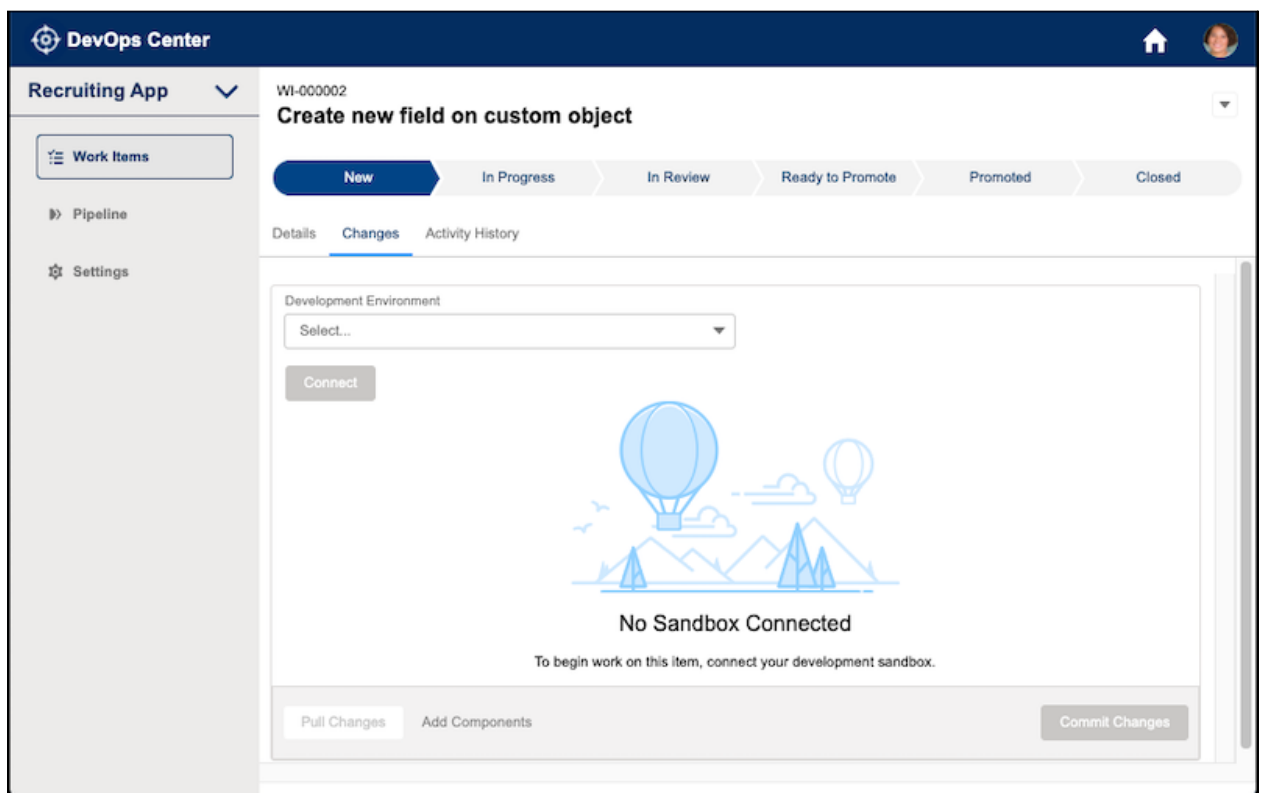
When you're ready to begin work, connect to the environment you want to use for this work item. For background information on work items and work item statuses, see [Manage Work Items](#).

The development environments are Developer or Developer Pro sandboxes with the source tracking in sandboxes feature enabled. The environments were added to the project during setup to make them available for team members. During setup, the team manager gave each environment a nickname to make them easier to recognize in this menu.

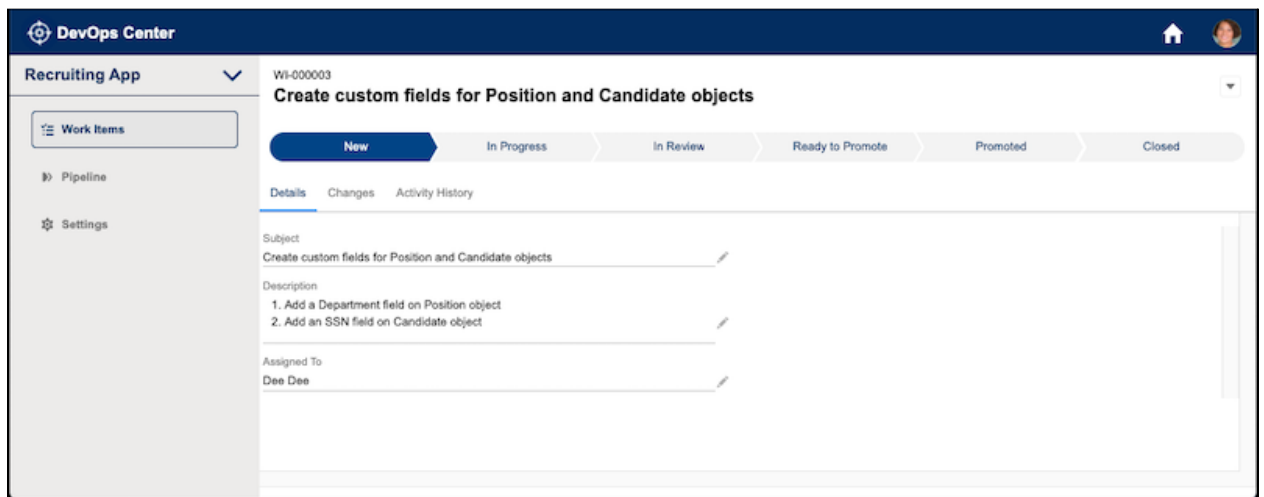
 **Note:** Go to the Settings page in DevOps Center to see the nickname and URL for each environment to confirm you're connecting to the correct environment.

A work item has a progress bar that shows which stage of the development process it's in. The work item starts in the New stage, meaning that you haven't made any changes in support of the bug or user story that the work item describes. (We'll go over [work item stages](#) in more detail a little later.)

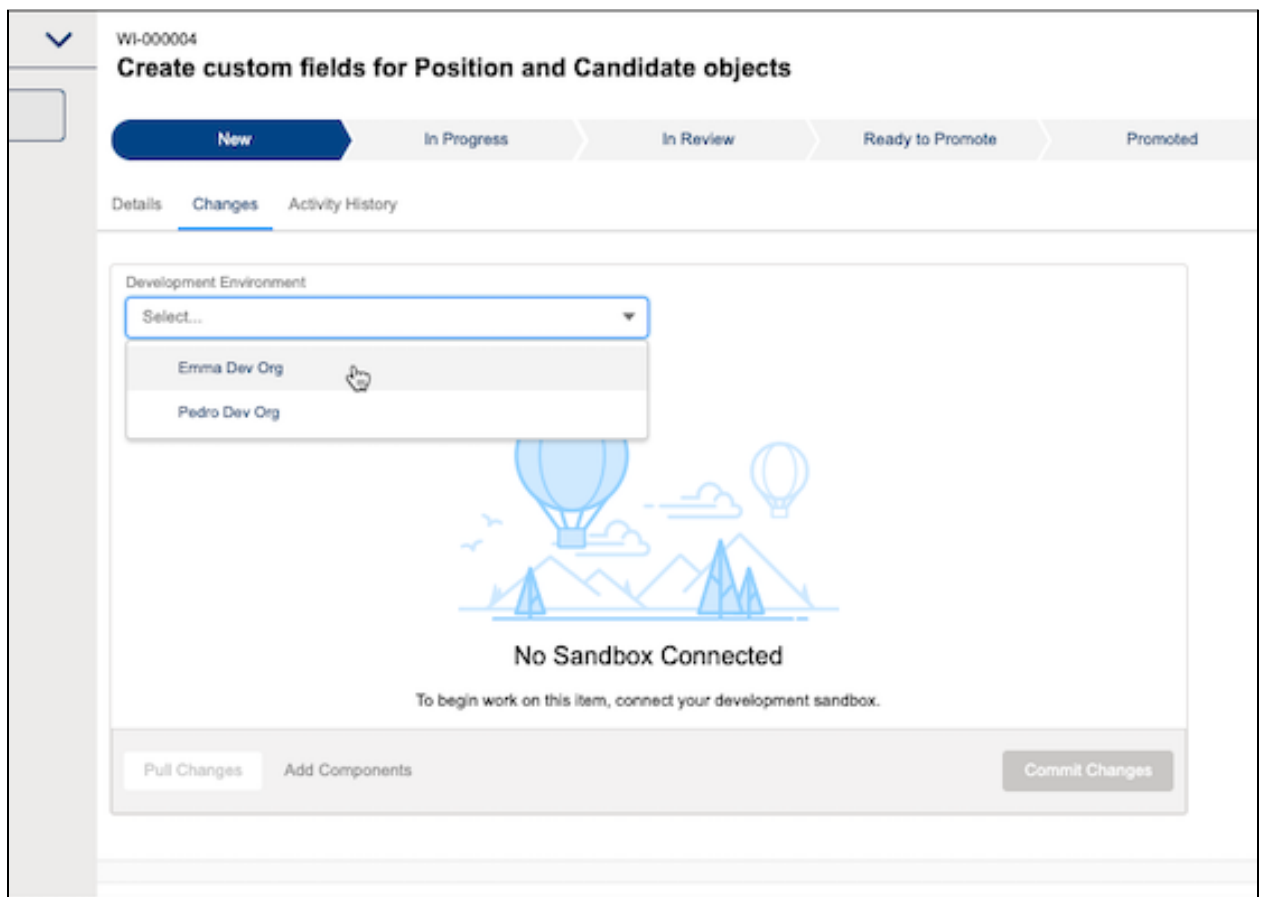
1. In Work Items, click the ID of a work item to open it.



2. Click the Details tab to view the description for the work item scope, or to assign it to yourself.

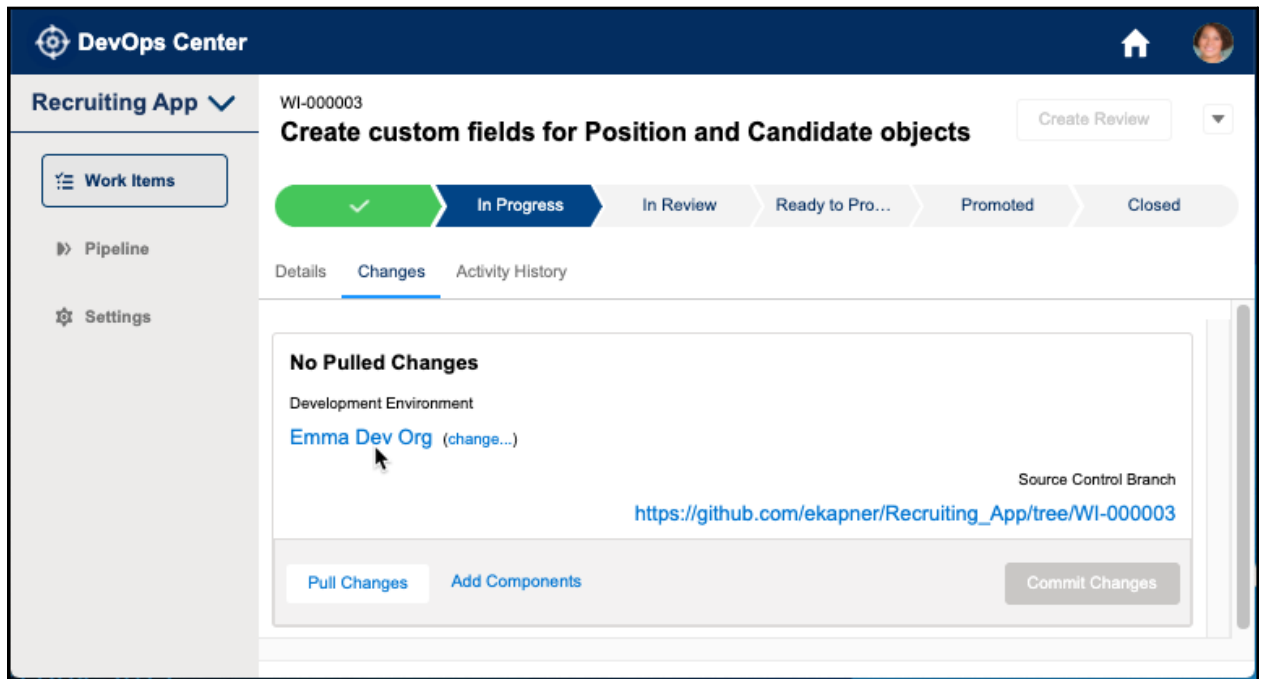


3. Back in the Changes tab, select the development environment to use for this work item.



4. Click **Connect** to log in. After you connect to a development environment, the work item moves to the In Progress status. A branch is also automatically created in the source control repository to manage the changes for this work item.

5. Click the link for the Development Environment name to open the development environment and begin making your changes.



View and Pull Changes from the Development Environment

The work item Changes tab is where you can see the changes you made in the development environment. As you make changes in the development environment, the source-tracking feature keeps a record of the component files that have been added, deleted, or changed.

After making and testing your changes, pull them from your development environment, so you can see the changes before committing them to the repository for review.

View Changes

The Changes tab lists how many files are available to pull (new changes from the dev environment), and how many files are already pulled but not yet committed in the source control repo (1).

Development Environment					
rDevelopment					
6 files available to pull 7 files pulled					
<input type="checkbox"/> File Name	Metadat...	Operation	Last Mod...	Last Mo...	
<input type="checkbox"/> Admin	Profile	CHANGE	Emma Coder	May 11, 2021	
<input type="checkbox"/> Custom: Sales Profile	Profile	CHANGE	Emma Coder	May 11, 2021	
<input type="checkbox"/> Custom: Marketing Profile	Profile	CHANGE	Emma Coder	May 11, 2021	
<input type="checkbox"/> Custom: Support Profile	Profile	CHANGE	Emma Coder	May 11, 2021	
<input type="checkbox"/> Recruiter__c-Recruiter Layout	Layout	ADD	Emma Coder	May 10, 2021	
<input type="checkbox"/> Candidate__c.Social_Security_Number__c	CustomField	REMOVE	Emma Coder	May 11, 2021	
<input type="checkbox"/> Candidate__c-Candidate Layout	Layout	CHANGE	Emma Coder	May 11, 2021	



Warning: DevOps Center doesn't currently support committing files marked as REMOVE (2). In some cases, the commit fails (if it's the only file being committed), or is silently ignored if it's part of a larger commit. Either way, the file isn't removed from downstream pipeline stages.

Pull Changes from the Development Environment

1. Click **Pull Changes**.

A list of the added, changed, and removed component files is generated from the source tracking in your development environment.

2. Select the changes to commit to the source control repository.

You can click the headings in the list of files to sort the list. If the list shows a file that you don't want to commit, leave its checkbox unselected. For example, you can exclude a Profile component file if it isn't part of your intended changes for the work item. Simply don't select it to leave it out of your commit.



Note: When you don't include files, they aren't included in your commit, but still exist in the development environment. Any changes that you pull, but don't commit, are indicated as pulled in all work items that are using this development environment.

Commit Any Metadata Component Manually

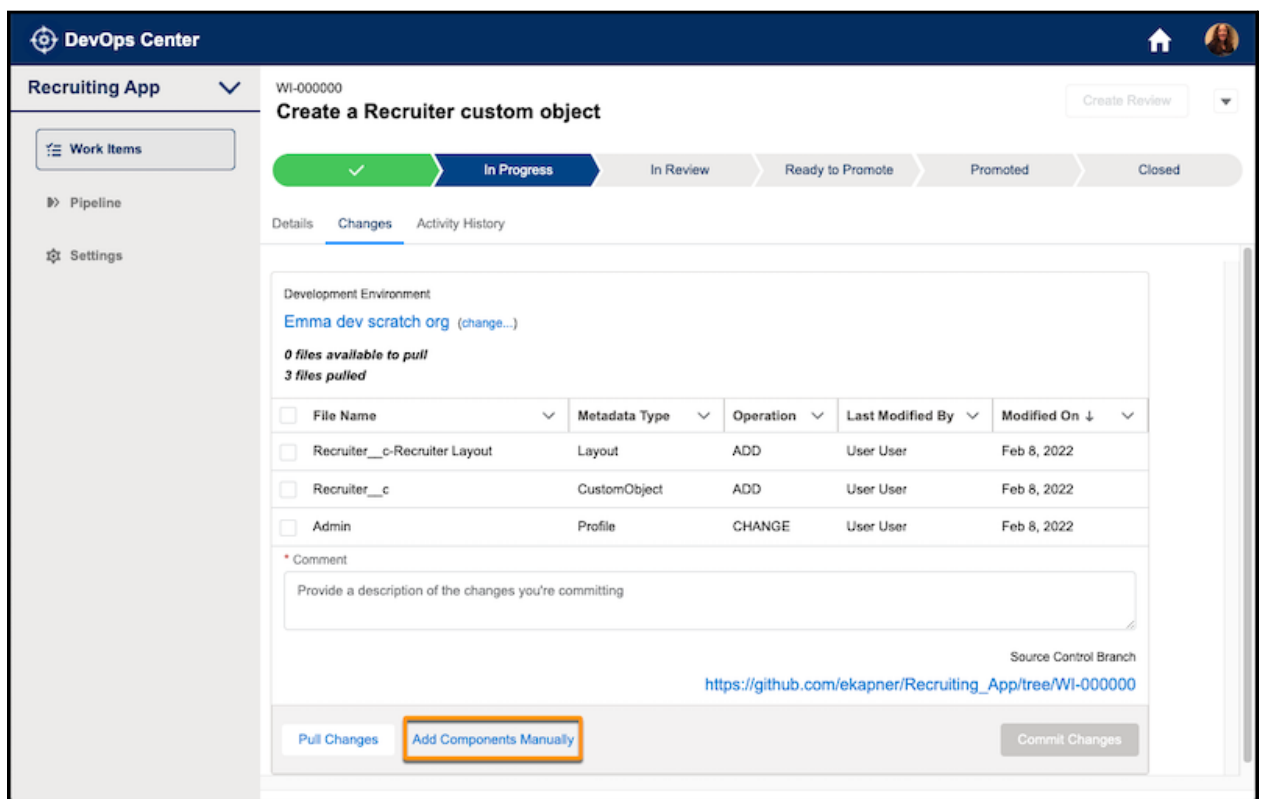
A key benefit of DevOps Center is centered around the idea that we track changes so you can focus on development tasks. However, for those times when you require flexibility to include more

metadata than what was pulled automatically, you can select and commit additional metadata source files, not just the ones that DevOps Center determines have changed. Some use cases include:

- Source tracking fails to identify that a file has changed.
- Your development environment is shared, and another teammate has already committed a file that you want to also include in your work item commit. If the component hasn't changed since the other teammate committed it, it doesn't appear in your changes list automatically.
- You want to commit files that were changed before source tracking was enabled in the development environment.

If you manually add a component that's already in the Changes list, the Operation type continues to reflect the original Operation type (CHANGE or ADD).

1. On the Work Item Changes tab, click **Add Components Manually**.



2. Select a component type from the list.
3. (Optional) Narrow the results by entering alphanumeric characters in the file name field, or selecting who last modified the component from the menu.
4. Select which components to add.

Known Limitation: We support standard metadata types, not decomposed Salesforce DX source format. For example, you can't select an individual custom field, but rather, you must

select the entire custom object. To learn more about DX source format, see [Salesforce DX Project Structure and Source Format](#) in the *Salesforce DX Developer Guide*.

Add Components

Component Type

CustomObject

File Name

Search Name...

Last Modified By

Emma Coder

<input type="checkbox"/> File Name	Metadata Type	Last Modified By	Modified On
<input checked="" type="checkbox"/> Candidate__c	CustomObject	Emma Coder	Jan 11, 2022
<input checked="" type="checkbox"/> Job_Position__c	CustomObject	Emma Coder	Jan 6, 2022
<input type="checkbox"/> Individual	CustomObject	Emma Coder	Dec 31, 1969
<input type="checkbox"/> QuickText	CustomObject	Emma Coder	Dec 31, 1969
<input type="checkbox"/> SocialPost	CustomObject	Emma Coder	Dec 31, 1969
<input type="checkbox"/> Location	CustomObject	Emma Coder	Dec 31, 1969
<input type="checkbox"/> CommSubscriptionChannelType	CustomObject	Emma Coder	Dec 31, 1969

Cancel

Add

5. Click **Add**.

After it's added, the component appears in the Changes list with an Operation type of Manual.

WI-000003

Create a custom object for Candidate and add two custom fields

Create Review

☒ In Progress
 ☐ In Review
 ☐ Ready to Promote
 ☐ Promoted
 ☐ Closed

Details **Changes** Activity History

<input type="checkbox"/>	Candidate__c.Department__c	CustomField	ADD	Emma Coder	Jan 11, 2022
<input type="checkbox"/>	Candidate__c-Candidate Layout	Layout	CHANGE	Emma Coder	Jan 11, 2022
<input type="checkbox"/>	Candidate__c.Social_Security_Number...	CustomField	ADD	Emma Coder	Jan 11, 2022
<input type="checkbox"/>	Candidate__c	CustomObject	ADD	Emma Coder	Jan 11, 2022
<input type="checkbox"/>	Trial Customer Portal User	Profile	CHANGE	Emma Coder	Jan 11, 2022
<input type="checkbox"/>	Job_Position__c-Job Position Layout	Layout	MANUAL	Emma Coder	Jan 10, 2022
<input type="checkbox"/>	Job_Position__c	CustomObject	MANUAL	Emma Coder	Jan 6, 2022

* Comment

Provide a description of the changes you're committing

Source Control Branch

https://github.com/ekapner/Recruiting_App/tree/WI-000003

Commit Changes to the Project Repository

After you're satisfied with the contents of the change request, commit the changes to the project repository. When you commit changes, you store them in the project repository branch in GitHub that was created when the work item moved to the In Progress stage. The branch is named after the work item, so it's clear where the changes it contains came from. For example, the branch for a work item WI-12345 looks something like:

<https://github.com/<account>/<repo-name>/tree/WI-12345>

1. In the Changes tab, verify that you selected all the components you plan to commit.
2. Add a comment about this particular commit.

Include what's distinctive about the changes, so you can identify an individual commit by more than just its timestamp. A single work item can have multiple commits. You can continue to make changes in your development environment and commit a new group of changes for the same work item.

3. Click **Commit Changes**.

The changes are committed to the work item's project repository branch.

After you commit changes to the repository, the source tracking for these components in your development environment is reset. This reset ensures that changes that are already committed don't keep showing up when you pull changes from the development environment.

4. (Optional) To see the branch in GitHub, click the Source Control Branch URL.

The Code page for the branch is shown. Explore the changes you recently committed in the source control repository.

View Work Item Events in Activity History

You can view the Activity History for each work item, which provides a comprehensive history of all key events, including promotions, commits, work item status changes, warnings, and errors (failures). This historical view enables you to maintain and view a record of events and associated details for auditing, troubleshooting, and general visibility purposes.

The screenshot displays the 'Recruiting App' interface. On the left is a sidebar with 'Work Items', 'Pipeline', and 'Settings'. The main area shows a work item titled 'WI-000000' with the subtitle 'Create a custom object called Referred by University'. A progress bar at the top indicates the status 'In Progress'. Below the progress bar are tabs for 'Details', 'Changes', and 'Activity History'. The 'Activity History' tab is active, showing a list of events under the heading 'Today'. The events include a 'Commit' (09:47:38 AM, Emma Coder, 2 files committed), a 'Comment' (New custom object for Referred by University checkbox), and a table of file changes. The table has columns for File Name, Metadata Type, Operation, Last Modified By, and Modified On. It lists two files: 'Referred_by_University___c-Referred by University Layout' (Layout, ADD, User User, Mar 9, 2022) and 'Referred_by_University___c' (CustomObject, ADD, User User, Mar 9, 2022). Below the table are three more events: '09:35:14 AM Dev Environment Conn...' (Work item connected to Emma scratch org), '09:35:12 AM Status Change' (Work item status changed from New to In Progress), and '09:33:50 AM Assignment' (Work item assigned to Emma Coder). On the right side of the activity history, there are filters for 'User' (Any), 'Activity Type' (Any), and 'Date Range' (Last Week).

File Name	Metadata Type	Operation	Last Modified By	Modified On
Referred_by_University___c-Referred by University Layout	Layout	ADD	User User	Mar 9, 2022
Referred_by_University___c	CustomObject	ADD	User User	Mar 9, 2022

Change Work Item Status to Never

Sometimes plans change. Sometimes you begin down one path and must start over. Sometimes a change conflicts with another change and the best strategy is to remove the files from the commit to allow another work item's promotion to succeed. In these cases, you can change the status of a work item to Never.

- You can change a work item's status to Never status before it's been promoted into a pipeline stage.
- Changing the status to Never is non-reversible.

When you change the status to Never, the work item becomes inactive, and the committed component files are returned to the list of available changes. You can recommit any of these files to another work item.

From the work item menu, select **Change Status to Never**, then click **Confirm**.

WI-000003

Create a custom object for Candidate and add two custom fields

Create Review

Change Status to Never...

Closed

Details Changes Activity History

<input type="checkbox"/>	Candidate__c.Department__c	CustomField	ADD	Emma Coder	Jan 11, 2022
<input type="checkbox"/>	Candidate__c-Candidate Layout	Layout	CHANGE	Emma Coder	Jan 11, 2022
<input type="checkbox"/>	Candidate__c.Social_Security_Number...	CustomField	ADD	Emma Coder	Jan 11, 2022
<input type="checkbox"/>	Candidate__c	CustomObject	ADD	Emma Coder	Jan 11, 2022
<input type="checkbox"/>	Trial Customer Portal User	Profile	CHANGE	Emma Coder	Jan 11, 2022
<input type="checkbox"/>	Job_Position__c-Job Position Layout	Layout	MANUAL	Emma Coder	Jan 10, 2022
<input type="checkbox"/>	Job_Position__c	CustomObject	MANUAL	Emma Coder	Jan 6, 2022

Synchronize Your Development Environment

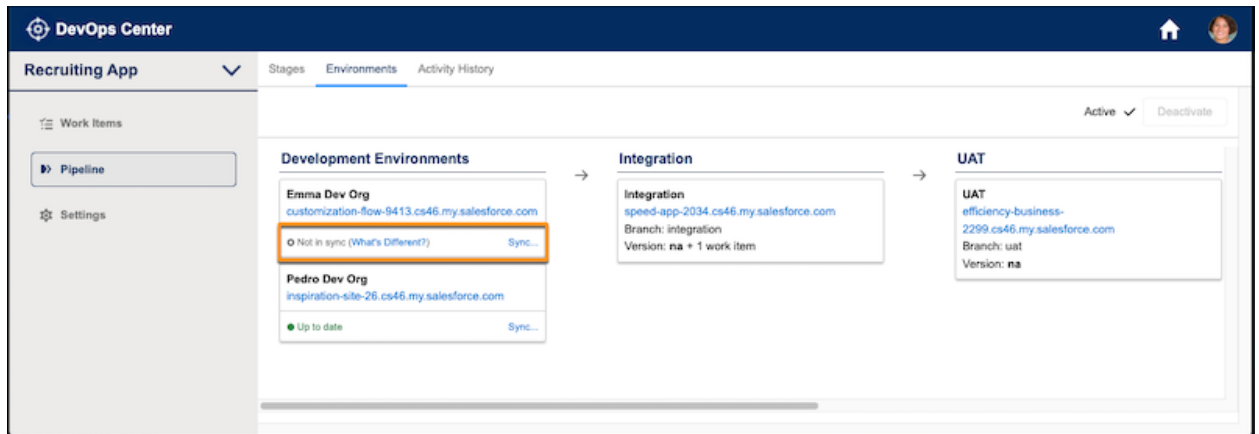
If you're working on a team with multiple developers, it's likely that your dev environment is going to become out of sync with the source control repository. DevOps Center tracks the differences between each development environment and the first integrated stage of the pipeline. When a development environment is connected to a work item, you're notified if it's not up-to-date and given the option to synchronize it before starting new work. You can also choose to synchronize it at any time.

This synchronization process ensures that you're developing against the latest integrated source of truth, including changes from other teammates. It also ensures that your changes are compatible with other changes in the pipeline and reduces the possibility of conflicts when promoting changes.

From the Environments tab, you can see whether each development environment is in sync or not, see the differences if it's not in sync, and optionally synchronize. The synchronization process runs a deployment of the changes from the first pipeline stage branch into the development environment. For this reason, you are presented with deployment options when the synchronization process is initiated.

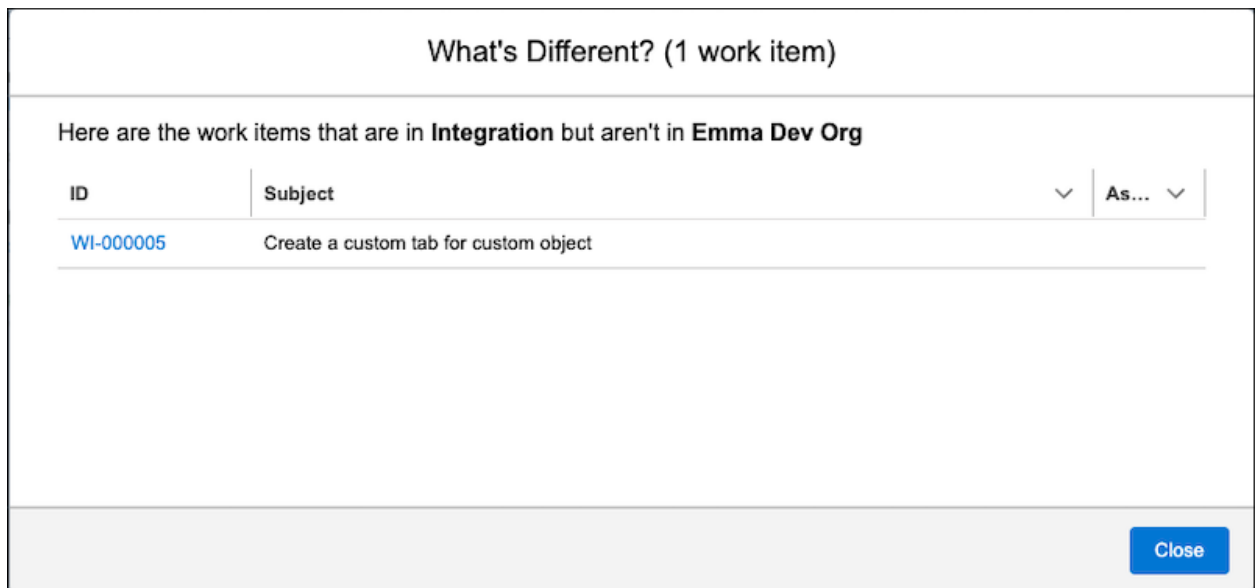
You can synchronize a development environment if it doesn't contain any In Progress work items.

1. From Pipeline, open the Environments tab.



The status for the development environment, Emma Dev Org, indicates that it's out-of-sync.

2. (Optional) To see the differences between the dev environment and the next pipeline stage's branch, click **What's Different**.



3. Click **Sync**.
4. Select the synchronization options.
 - **Changes Not in Dev Environment** (recommended) – deploy only the changed metadata (files) from the next stage's branch to your development environment.
 - **All** – deploy everything in the next stage's branch to the development environment. Select this option if your deployments are failing due to dependent or missing

metadata that doesn't exist in the What's Different list but does exist in the next stage's branch.

The synchronization process runs a deployment of the changes from the first pipeline stage branch into the development environment. For this reason, you're presented with deployment options when the synchronization process is initiated.

5. Click **Start Sync**.

See also:

[Apex Test Options](#)

Are You Sharing a Development Environment?

We recommend that all developers have their own development environment. Here are some considerations if you are sharing a development environment.

Changes are associated in DevOps Center with the *development environment* within a project. A single development environment can be connected to multiple work items.

When you pull changes, the changed components are displayed in the changes list, but no files have been retrieved out of the development environment yet. The file isn't actually retrieved from the development environment until it's committed.

After a component is committed, it no longer appears in the changes list for any work item that is connected to that development environment, until the component is changed again in the development environment.

If multiple developers are using one development environment (within the same project), actions by one developer can have ramifications on the other. For example:

- If two developers are making changes to the same component, after one developer commits the component, it no longer shows up in the other developer's change list (until the component is changed again in the development environment and pulled again). This disconnect can result in not all the needed files being included in a work item's commit and feature branch.
- If someone else has made a change to the component since you pulled it, the committed file is included with those latest changes. Consequently, the file you're committing could be different than what you expect.
- When you pull changes, you see all the files changed by anyone who is making changes in the development environment, since the files' last commit.
- If multiple developers are making changes in a shared development environment, the Last Modified By column in the changes list indicates who made the *last* change. So if you're using that field to identify changes you've made, you could miss some if someone else has modified it more recently than you.

Share Your Changes for Team Member Review

When changes for an In Progress work item are committed and ready to share with reviewers, creating a Review opens a change request, called a *pull request* in GitHub.

In GitHub, a pull request is how you get your proposed changes reviewed and approved before they're merged with the rest of the code. The pull request pulls your changes from the work item branch so that they can be merged into the next branch in your pipeline. Before that happens, however, a team member reviews the work item changes. The pull request presentation format makes reviews and online discussion easier, because it highlights differences between the work item branch and the main branch.

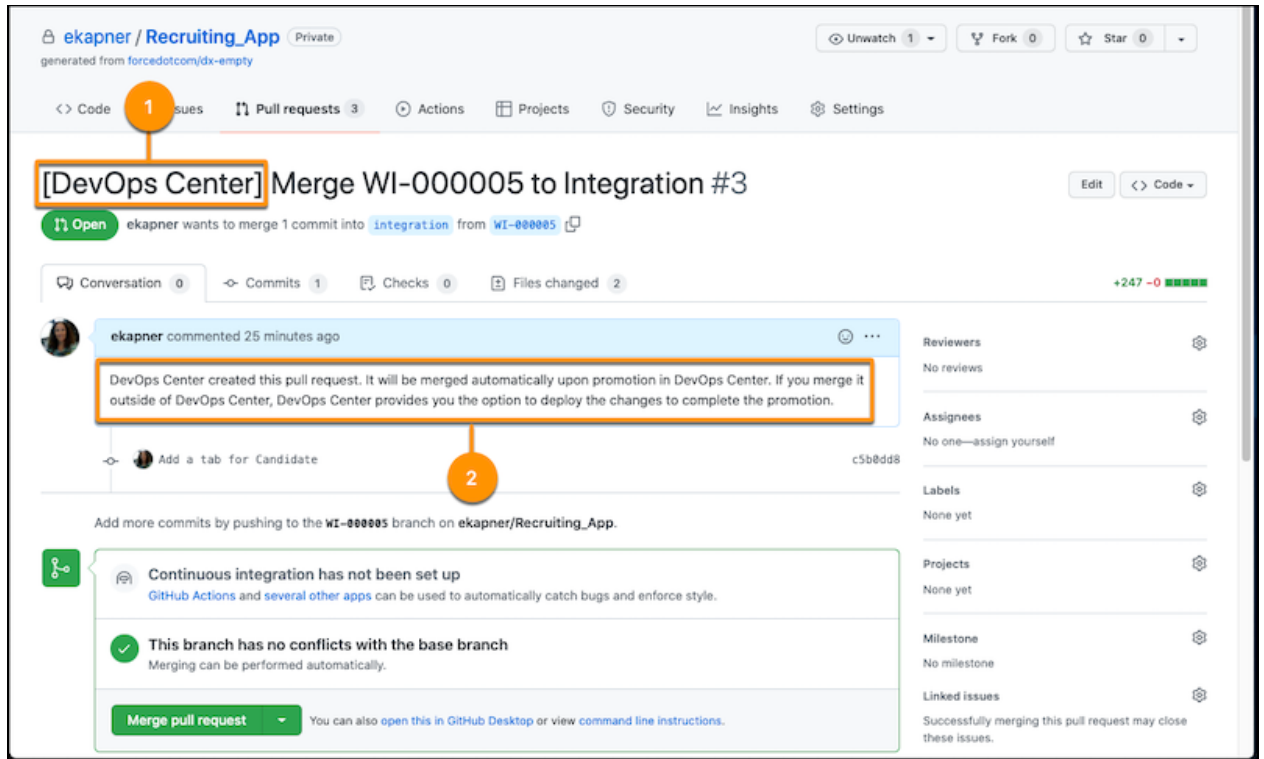
Review Changes in GitHub

1. From the work item, click **Create Review** to move the work item to In Review.

DevOps Center creates a change request and automatically opens a pull request for the changes in your work item branch.

2. (Optional) Click **View Change Request** to open a browser tab showing the pull request in the source control repository, where reviewers can see the file changes.

In GitHub, you can identify that the change request was created by DevOps Center by the subject line (1) and auto-generated comment (2).



3. Team members can add comments and have discussion in the source control pull request about the changes.
4. Go back to DevOps Center. What you do next depends on whether you merged the pull request in GitHub.
 - If you didn't merge the pull request, [indicate that the work item is ready to promote](#).
 - If you merged the pull request, the work item status is updated to Ready to Promote. The next step is to [complete the promotion](#).

Indicate That Work Items Are Ready to Promote

After the work item is reviewed, a team member approves the work item to be ready to promote. Work items aren't available for promotion through the pipeline until they are in the Ready to Promote status.

You can click the toggle to change the status to be not Ready to Promote until the work item has been promoted.

If you merged the pull request in GitHub, the work item is automatically moved into the Ready to Promote status and can't be changed back. Because the change is now in the next stage's branch, your only option is to complete the promotion.

Promote Work Items Through Your Pipeline

You promote work items through a pipeline, which defines the sequence of stages that work items progress as they go through the release lifecycle from development to production (or some other final stage). You can have any number of pipeline stages. Your team manager built the pipeline when configuring DevOps Center.

Each pipeline stage corresponds to an environment (currently a Salesforce org), and a branch in the source control repository. Depending on how your pipeline is configured, changes move through the pipeline when individual work items or a grouped set of work items (work item bundle) is promoted. Upon promotion, changes are merged from the current stage branch to the next stage branch, and then are deployed to the next stage org.

At a minimum, we recommend that your pipeline has one test stage and a production stage. However, it's common to have two to three test stages, often called something like integration, UAT (user acceptance testing), and staging.

Your team manager can configure your pipeline in one of two ways:

- Allow you to move work items individually through the entire pipeline.
- Allow you to move work items individually in early stages of the pipeline, and as a work item bundle in later stages. 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. When changes are promoted from the bundling stage to the next stage, all work items that haven't yet been promoted are included in the versioned work item bundle and promoted as a unit. This versioned work item bundle continues to be promoted as a consistent unit through subsequent stages when you perform a promotion.

Your pipeline configuration determines how many stages allow you to promote individual work items. Stages that allow for individual work item promotion have the Promote Selected button at the top, while stages that allow for versioned bundle promotion have the Promote Work Item Bundle or Promote Work Item Bundles button at the top. The bundling stage, where the versioned bundle is created, has the Promote as Work Item Bundle button at the top.

See also:

[Conflict Detection and Resolution](#)

DevOps Center Setup Guide: Configure Your Pipeline

GitHub Mergeability

Promotions can be blocked if there's a conflict or if a mergeability rule isn't met. DevOps Center honors mergeability, as determined by GitHub.

- Work items can be marked Ready to Promote only when the changes are deemed mergeable. GitHub determines mergeability either by merge conflicts or custom mergeability

rules. If GitHub identifies merge conflicts with the changes, then the changes can't be promoted until the conflicts are resolved.

- Similarly, if custom mergeability rules haven't been met for the changes, then the changes can't be promoted until those rules are met.

See the GitHub documentation for more information regarding the [mergeability of pull requests](#).

Promotion Options

When a promotion occurs, DevOps Center merges the branch from the originating stage into the target stage's branch. It then deploys the metadata from the target stage's branch to the target stage's org.

When you promote work items through the pipeline, you can select whether to deploy only the items that changed or all the changes in the target stage's branch.

- **Changes not in the <stage-name>'s branch** (recommended) – Deploy only the files from that target stage's branch that aren't yet in the target stage's org. In effect, you're deploying only the files (changes) that recently came from the originating stage as part of this promotion. This behavior is the default, and is more efficient because we're deploying fewer files. In most circumstances, selecting this option is sufficient.
- **All metadata in the <stage-name>'s branch** – Deploys all files in the target stage's branch to the target stage's org, after the branch has been merged. Select this option if your deployments are failing due to dependent or missing metadata that doesn't exist in the defined set of changes yet exists in the target stage branch. This option is useful when you or a colleague has committed and merged changes to the branch outside of DevOps Center but DevOps Center doesn't know about them.

Apex Test Options

When you promote work items through the pipeline, decide which Apex tests to run based on pipeline stage and to which environments you're deploying changes.

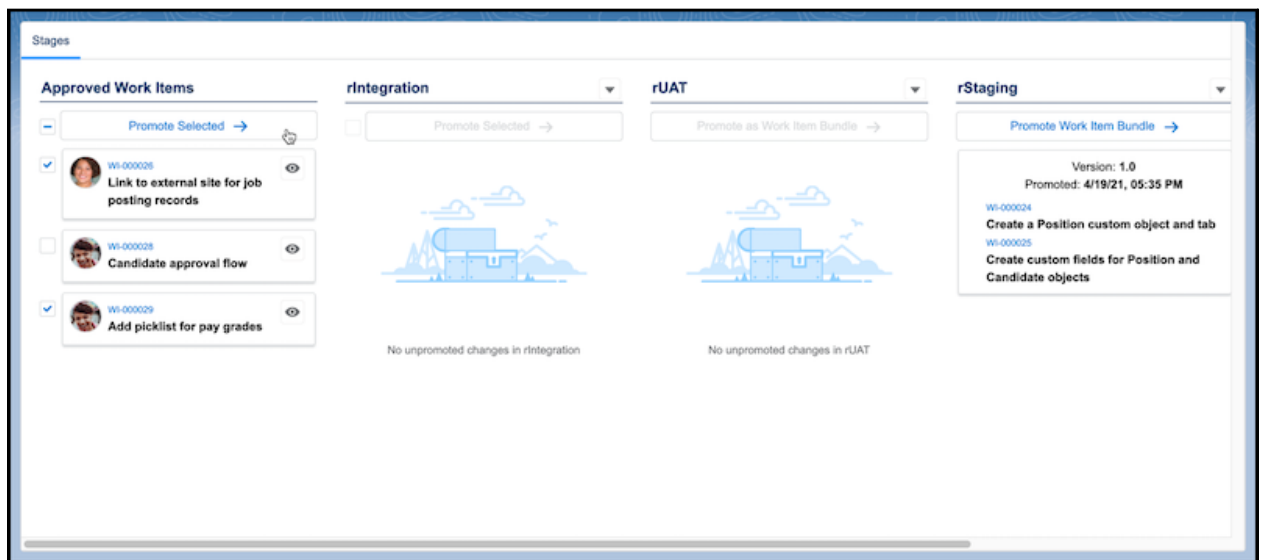
- **Default** – In sandboxes and scratch orgs, no tests are run. In production, all local tests are run if your changes contain Apex classes or triggers. Excludes tests included in installed packages.
- **Run local tests** – All local tests are run, excluding tests included in installed packages.
- **Run all tests** – All tests in your organization are run, including tests from installed packages.
- **Run specified tests** – Only the tests that you specify are run. Provide the names of test classes in a comma-separated list (no spaces between entries).
Example: `TestDeleteData,TestDataFactory.createTestRecords`

See also:

Apex Developer Guide: [Testing and Code Coverage](#)

Promote Individual Work Items

1. Click **Pipeline**.
2. Under Approved Work Items, select the work items to promote to the next stage.
3. Click **Promote Selected**.

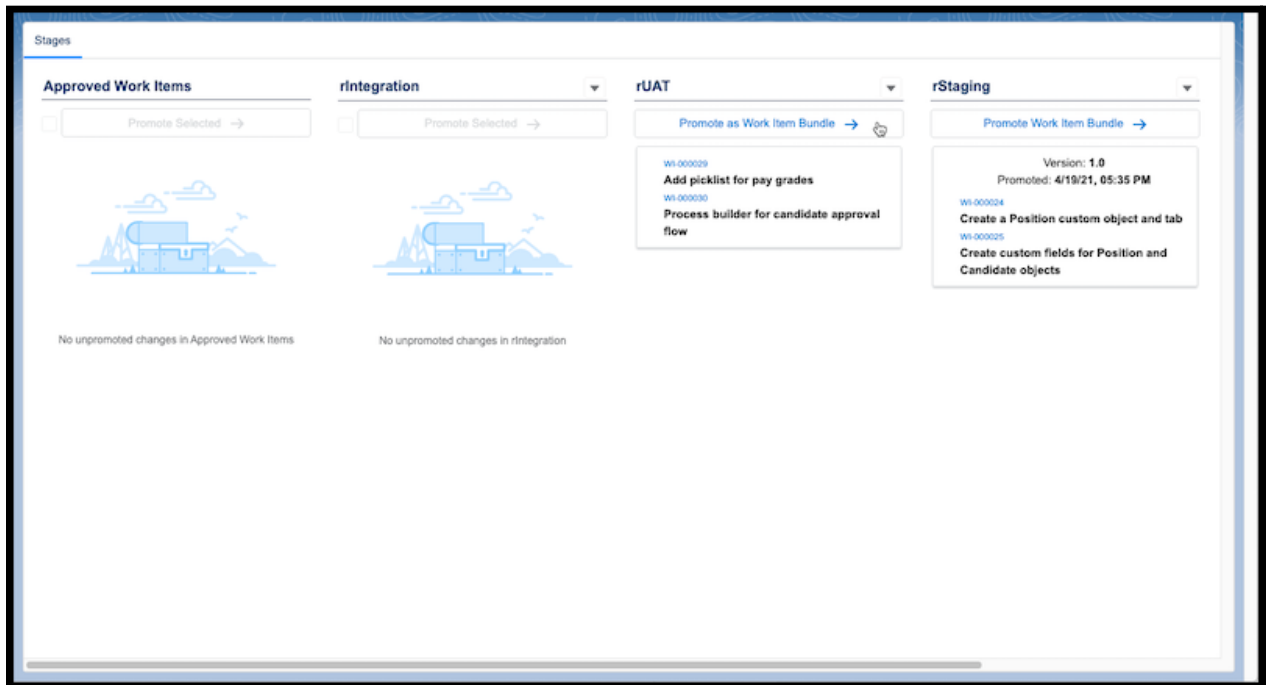


4. If it's your first time accessing this stage's environment, click **Connect** to log in to the stage's environment, then repeat steps 2–3.
5. In the Promotion Options dialog, select a [promotion option](#) and which [Apex tests to run](#).
6. Click **Promote**. The selected work items are merged and deployed to the environment associated with the next stage.

Promote as Work Item Bundle

During pipeline configuration, your project manager can define a stage as the bundling stage. Consequently, you can't select individual work items for promotion. Instead, promotions in this stage result in the creation of a work item bundle. All unpromoted work items in this stage become part of the bundle.

1. In the Pipeline view, in the bundling stage, click **Promote as Work Item Bundle**.

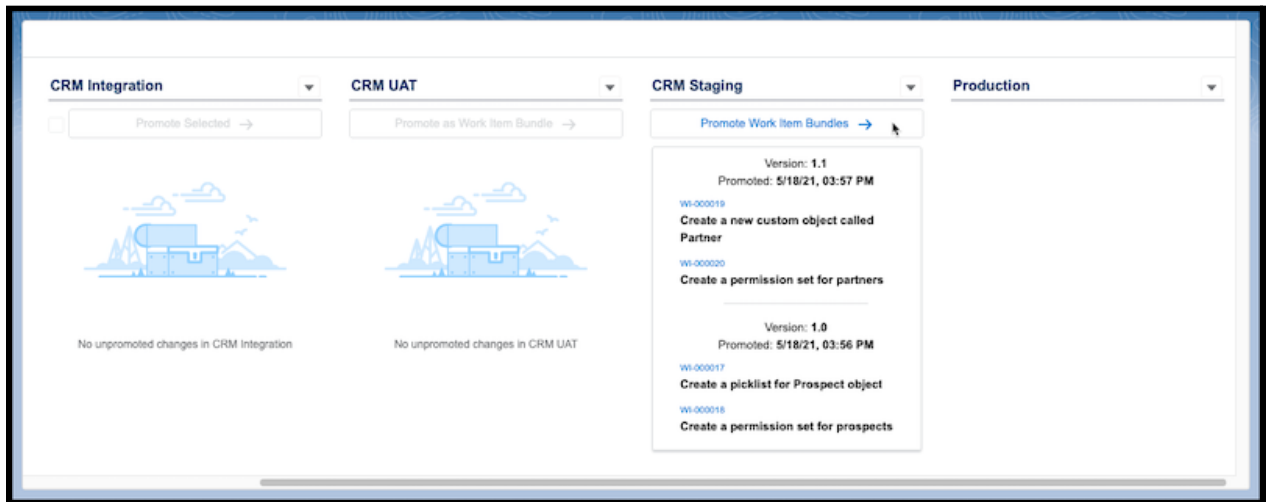


2. If it's your first time promoting to this stage, click **Connect** to log in to the stage's environment, then repeat step 1.
3. In the Promotion Options dialog, enter a unique version identifier. Indicate the version as an alphanumeric string of your choice.
4. Select which Apex tests to run.
5. Click **Promote**. The work item bundle is created with the version you specified, then merged and deployed to the environment associated with the next stage.

Promote Work Item Bundles

When you promote work item bundles, all unpromoted versioned bundles are promoted together to the next stage. The unpromoted work item bundles and corresponding work items are listed here.

1. In the Pipeline view, in a versioned stage, click **Promote Work Item Bundle(s)**.

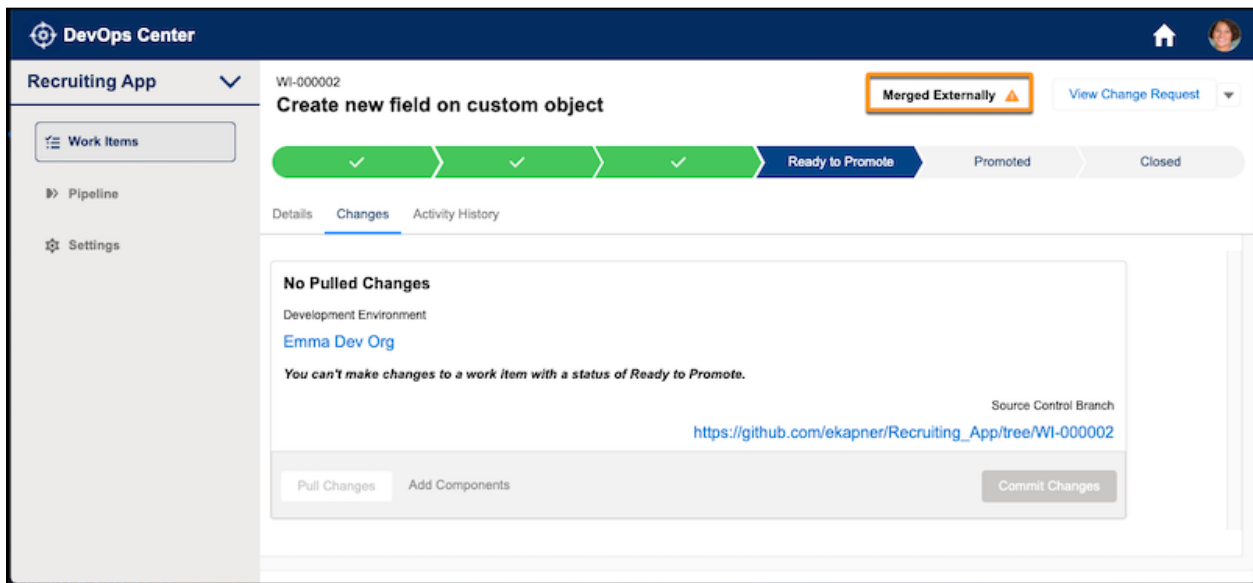


2. If it's your first time promoting to this stage, click **Connect** to log in to the stage's environment, then repeat step 1.
3. Select a [promotion option](#) and which [Apex tests to run](#).
4. Click **Promote**. Each versioned work item bundle is merged and deployed to the environment associated with the next stage. Only the latest work item bundle version is displayed, but all earlier work item bundles are also present in the stage.

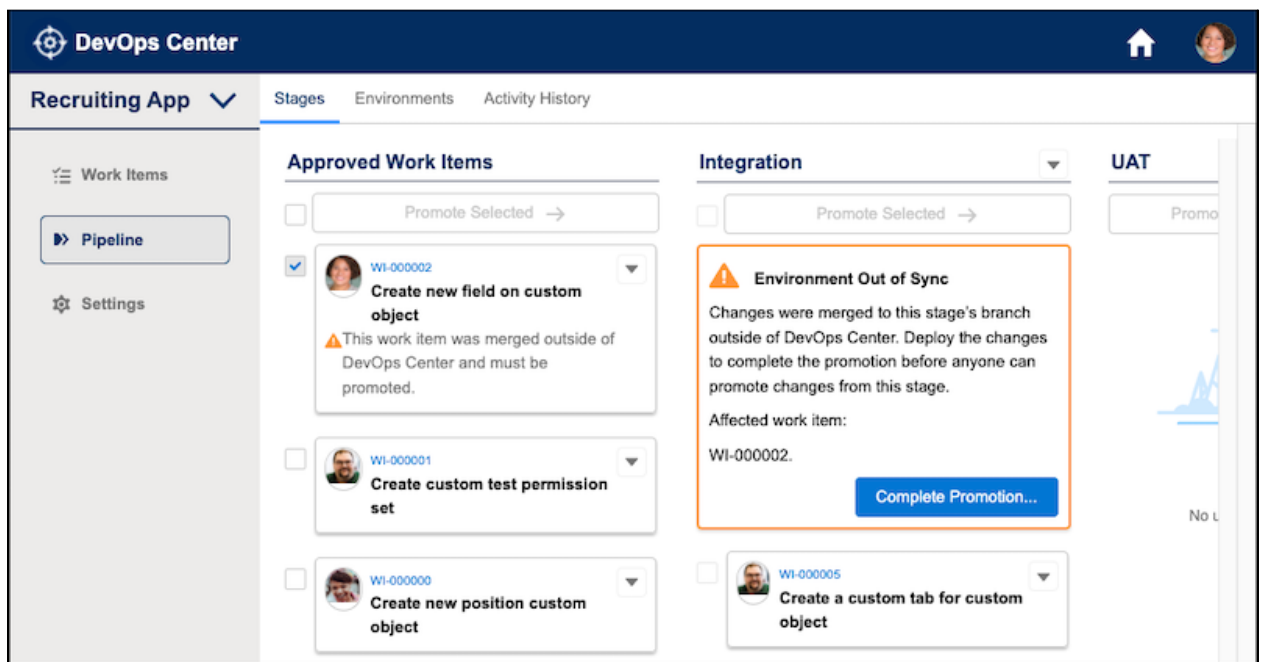
Promote Changes Merged Outside of DevOps Center

If you merge the pull request in GitHub, the changes are in a partially promoted state because they've been merged but not yet deployed. In DevOps Center, complete the promotion to deploy the changes to the environment. To keep everything in sync, whether you merged changes for a work item or changes in a downstream pipeline branch, you must finish the promotion before any more changes can be promoted to or from the stage.

In this example, you merged changes outside of DevOps Center for a work item. At this point, the work item can't be moved back to In Progress. The promotion must be completed to bring the Integration stage environment (org) in sync with the Integration stage branch in the repository.

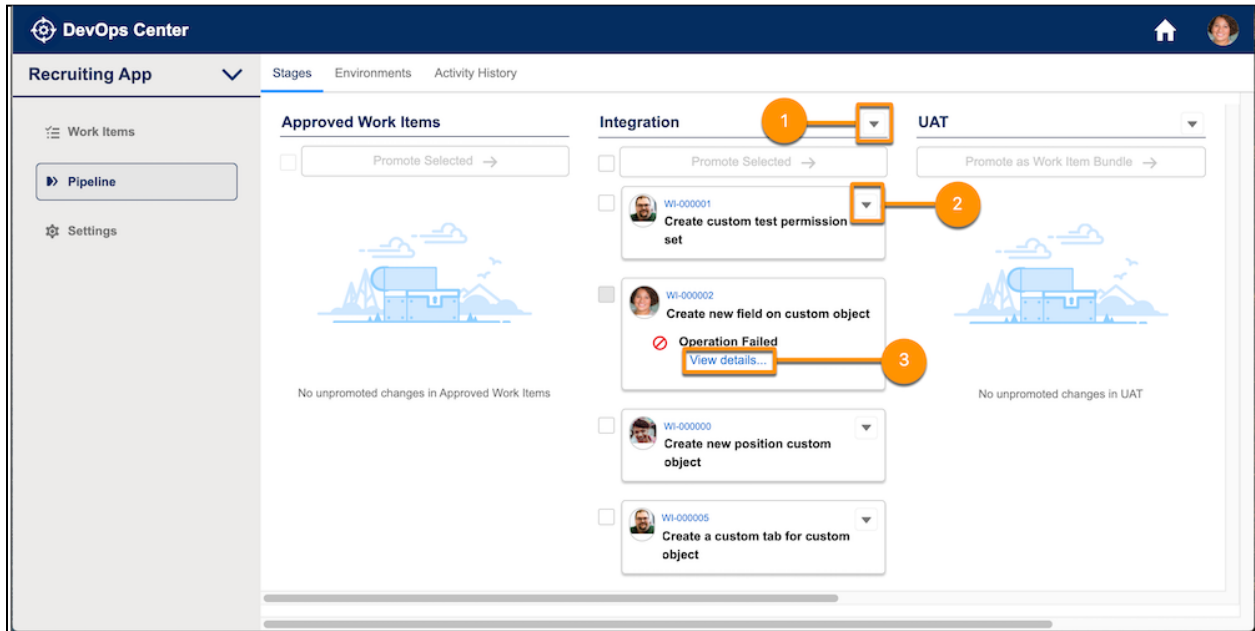


1. Click **Pipeline**.
2. Click **Complete Promotion**.



View Promotion Status

When promotion completes, you see a success banner or a failure banner where you can see details of the error. You have several options to validate the promotion or dig into errors.



From the pipeline stage dropdown menu (1):

- Open Environment – opens the environment so you can validate changes.
- View Branch in Source Control – opens the branch in source control so you can view the files in this branch.
- View Last Commit in Source Control – opens a list of commits (individual changes to a set of files) in source control. Here, you can see what's changed between two branches.

From the Work Item:

- View Change Request from the work item dropdown menu (2) – opens the pull request in source control so you can view the changes and troubleshoot any merge conflicts.
- View details (3) – opens the Activity History tab for a failed promotion, where you can view more information.

From the Activity History tab:

- View the promotion details including work items that were part of the promotion.
- View whether the promotion succeeded or failed, failure details, and the date and time of completion.

See also:

[View Pipeline Events in Activity History.](#)

View Deployment Status to Dig Deeper

If the deployment error doesn't provide sufficient information, you can see more details in the Deployment Status in the org. Open the environment and view the Deployment Status. To look at the deployment status in the org:

1. From the stage dropdown menu, select **Open Environment**.
2. From Setup, enter Deployment Status in the Quick Find box, then select **Deployment Status**.
3. From the list of Succeeded or Failed deployments, click **View Details**.
4. View and verify the actual changes in the org.

View Pipeline Events in Activity History

DevOps Center now shows a comprehensive history of all key pipeline events, including promotions, synchronizations, and errors (failures). This feature allows you to maintain and view a record of events and associated details for auditing, troubleshooting, and general visibility purposes.

The screenshot displays the 'Activity History' page for the 'Recruiting App'. The left sidebar shows navigation options: 'Work Items', 'Pipeline', and 'Settings'. The main content area is titled 'Today' and lists several pipeline events. The first event is a 'Promotion Completed' for work item WI-000000. The second event is a 'Promotion Started' for work item WI-000000. The third event is a 'Promotion Completed' for work item WI-000001, which is highlighted in red. Below this event, there is a section titled 'What Happened' and 'What can you do about it?', followed by an 'Error Details' section. The right sidebar contains filters for User, Stage, Activity Type, and Date Range.

Work Item ID	Work Item Subject	Assigned To	Approved By
WI-000000	Create a custom object called Referred by University	Emma Coder	Emma Coder
WI-000000	Create a custom object called Referred by University	Emma Coder	Emma Coder
WI-000001	Create a new field on the referred by university custom object	Emma Coder	Emma Coder

For errors, click **Error Details** to get more information to assist you in pinpointing and resolving the issue.

Conflict Detection and Resolution

Conflicts can occur when you're working with multiple development environments and moving changes between multiple pipeline stages. Conflicts can take different forms, and DevOps Center provides functionality to help you identify potential conflicts early and resolve them.

Here are some common types of conflicts.

Multiple Work Items That Modify the Same Source File

If you have multiple work items that modify the same source file (particularly in cases where multiple developers are touching the same source file in their own respective development environments), the changes can conflict or potentially overwrite each other when they're merged into the first integrated pipeline stage. DevOps Center warns you about the possibility of conflict when you promote multiple work items that contain the same source file. You can then choose to continue with the promotion or cancel to further analyze the potential conflict, or promote the work items one at a time.

Unresolvable Merge Conflict in Two Branches

When a work item is promoted from one stage to the next, the source files contained in the work item are merged from the first stage's source control branch to the next stage's source control branch. The source control system, GitHub, helps to determine if the changes are mergeable before you attempt to promote them. If GitHub detects a conflict, the promotion is blocked.

You can see more information on the merge conflict in the error dialog in the Activity History, and you can view additional details of the conflict in the pull request in the source control system. You can [resolve the conflict manually](#) directly in the source control system, and then you can attempt the promotion in DevOps Center.

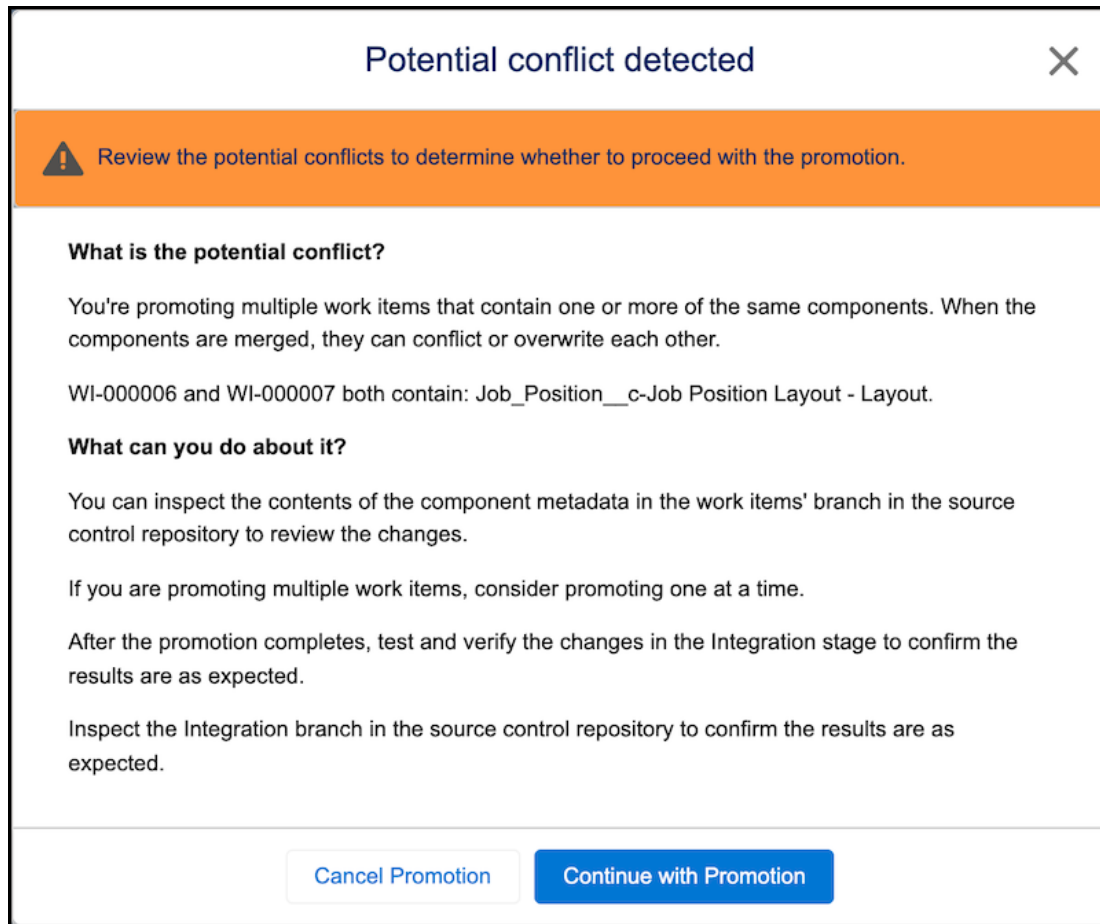
Your Component Version Is Different from Merged Component

When you have a stage that contains multiple work items that contain the same component, the stage contains a merged version of the component. If you then choose to promote only one of those work items from the stage to the next stage, it's possible that the version of the component that you promote is different from the merged component that was tested. DevOps Center warns you when this situation occurs. We recommend that you promote all work items that contain the common component together, so that the next stage contains the same merged version of the component as what was tested in the previous stage.

Resolve Merge Conflicts in GitHub

Merge conflicts can occur when multiple work items contain one or more of the same components, especially when multiple developers are making changes that impact the same components in different development environments. DevOps Center warns you about potential merge conflicts so you can investigate to ensure that you don't inadvertently overwrite desired changes.

In this example scenario, two developers are adding new fields to the same standard object, which causes potential merge conflicts in the page layouts.



Before you proceed, read the information about the potential conflict to determine what to do next.

- **Continue with Promotion**
Based on the conflict description, you feel comfortable that the changes don't overwrite each other.
- **Cancel the Promotion**
Based on the conflict description, you're uncertain that the changes don't overwrite each other, or you'd like to look at the pull request for the work item, which indicates whether GitHub has detected merge conflicts. After you complete your investigation, you either [resolve the merge conflicts in GitHub](#), sync the changes in the development environments, or continue with the promotion, which could fail if there's a merge conflict.

If the Promotion Fails

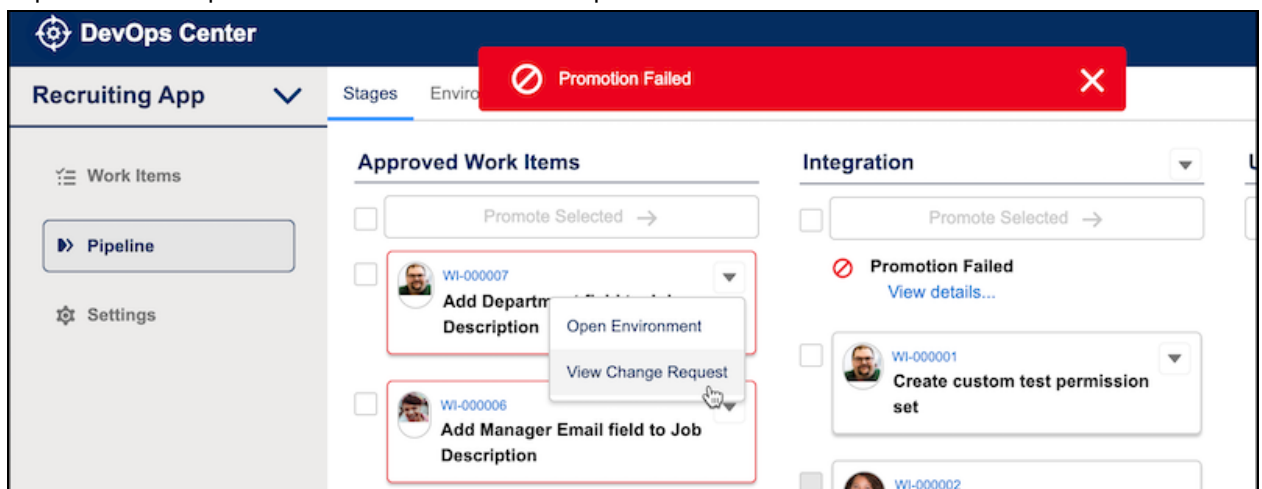
If the promotion fails, here are some options:

- Promote each work item individually.
 - If all individual work items are promoted to the next stage successfully, your work is done (for now).
 - If some work items are promoted successfully and some aren't, [review the change request in GitHub](#) to resolve the merge conflicts manually in the branch in the source control system.
- Sync the development environments, or resolve the conflicts directly in the development environments, then try the promotion process again.

Review and Address Conflicts in GitHub

After trying to evaluate and mitigate any potential merge conflicts, the promotion for a work item is still failing. You can manually edit the files to resolve the conflict directly in GitHub.

1. From the work item dropdown (on the stage or work item you're promoting from), select **View Change Request** to go into GitHub to view the pull request and see the merge conflicts. Repeat these steps for all work items where the promotion failed.



2. In GitHub, click **Resolve Conflicts**.

[DevOps Center] Merge WI-000006 to Integration #8

Open ekapner wants to merge 1 commit into [integration](#) from [WI-000006](#)

Conversation 0 Commits 1 Checks 0 Files changed 2

ekapner commented 17 hours ago

DevOps Center created this pull request. It will be merged automatically upon promotion in DevOps Center. If you merge it outside of DevOps Center, DevOps Center provides you the option to deploy the changes to complete the promotion.

Added a manager email field on the Job Position page f5cb0be

Add more commits by pushing to the [WI-000006](#) branch on [ekapner/Recruiting_App](#).

This branch has conflicts that must be resolved [Resolve conflicts](#)
Use the [web editor](#) or the [command line](#) to resolve conflicts.
Conflicting files
force-app/main/default/layouts/Job_Position__c-Job Position Layout.layout-meta.xml

Merge pull request You can also [open this in GitHub Desktop](#) or view [command line instructions](#).

In our example scenario, GitHub has identified one conflict (1) in one file, the Job Position Layout (2). If GitHub detects multiple conflicts in a file, you can use the **Prev** and **Next** links (3) to address each conflict one-by-one.

[DevOps Center] Merge WI-000006 to Integration #8

Resolving conflicts between [WI-000006](#) and [integration](#) and committing changes → [WI-000006](#)

1 conflicting file

Job_Position__c-Job Position Lay...
...c-Job Position Layout.layout-meta.xml

2

force-app/main/default/layouts/Job_Position__c-Job Position Layout.layout-meta.xml

1 conflict

Prev

Next

Mark as resolved

1

3

4

5

```
1 <?xml version="1.0" encoding="UTF-8"?>
2 <Layout xmlns="http://soap.sforce.com/2006/04/metadata">
3   <layoutSections>
4     <customLabel>false</customLabel>
5     <detailHeading>false</detailHeading>
6     <editHeading>true</editHeading>
7     <label>Information</label>
8     <layoutColumns>
9       <layoutItems>
10        <behavior>Required</behavior>
11        <fieldName>/field>
12      </layoutItems>
13      <layoutItems>
14        <behavior>Edit</behavior>
15        <field><WI-000006>
16          <field>Manager_Email__c</field>
17        </WI-000006>
18        <field><integration>
19          <field>Department__c</field>
20        </integration>
21      </layoutItems>
22    </layoutColumns>
23    <layoutColumns>
24      <behavior>Edit</behavior>
25      <field>OwnerId</field>
26    </layoutItems>
```

35

The format of the conflict as shown in GitHub is:

```
<<<<<<<<< <branch 1 name>
<code in branch 1>
=====
<code in branch 2>
>>>>>>>>> <branch 2 name>
```

In this example, the work item you attempted to promote (4) in branch WI-000006 is introducing a new field, while the CRM Integration branch doesn't contain that new field. Also, the CRM Integration branch contains a field that the work item branch doesn't contain (5), which was likely introduced by a different work item. In this case, you want to keep both these new fields, so you modify the file to eliminate the conflict.

The [Metadata API Developer Guide](#) provides the schema for each metadata type to assist you with editing the files directly. In this case, we can look at "Layout" in the guide to view the schema for <layoutItems>.



The resulting changes now look like this:

```
<layoutItems>
  <behavior>Edit</behavior>
  <field>Manager_Email__c</field>
</layoutItems>
<layoutItems>
  <behavior>Edit</behavior>
  <field>Department__c</field>
</layoutItems>
```



3. At the top of the file, click **Mark as Resolved**.
4. Continue until you resolve all conflicts in the listed files.
5. Click **Commit Merge**, which saves your changes in the work item branch, W-000006.

Notice that the conflicts are now resolved.


[DevOps Center] Merge WI-000006 to Integration #8



 Open ekapner wants to merge 2 commits into `integration` from `WI-000006` 

Conversation 0 Commits 2 Checks 0 Files changed 2


 ekapner commented 18 hours ago 


DevOps Center created this pull request. It will be merged automatically upon promotion in DevOps Center. If you merge it outside of DevOps Center, DevOps Center provides you the option to deploy the changes to complete the promotion.


 ekapner added 2 commits 18 hours ago

-  Added a manager email field on the Job Position page f5cb0be
-  Merge branch 'integration' into WI-000006 Verified c4a0e9e


Add more commits by pushing to the `WI-000006` branch on `ekapner/Recruiting_App`.

 Continuous integration has not been set up
[GitHub Actions](#) and [several other apps](#) can be used to automatically catch bugs and enforce style.

 **This branch has no conflicts with the base branch**
Merging can be performed automatically.

Merge pull request  You can also [open this in GitHub Desktop](#) or view [command line instructions](#).

- Return to DevOps Center to promote the work item, which merges the branch with the branch in the next stage, in this case, Integration.
- In DevOps Center, promote the work items again.

 Important: After you resolve conflicts, your development environments are out of sync with downstream pipeline environments. Lucky for you, DevOps Center has a feature for you.

See also:

[Synchronize Your Development Environment](#)

Manage Work Items

In DevOps Center, a team uses *work items* to track the progress of changes created to achieve a specific objective or task, such as enabling a user story or addressing a bug. Managing releases through work items makes it easier to track the progress of the overall release and identify areas of concern.

A work item encapsulates information about an objective or task while it's being worked on. After a work item is completed, the work item is an historical record if questions arise about those changes. Information in a work item includes:

- **ID:** An automatically generated unique identifier for a work item that has a **WI-** prefix (for example, WI-12345). The ID helps DevOps Center users distinguish between similar-sounding work items. It's also used in the project repository to identify the branch where changes for the work item are stored.
- **Subject:** Required. A brief description of the task to be tracked.
- **Description:** A more detailed description of the task to be tracked. If the work item is a bug, include an example of the problem to be solved. If the work item is a user story, include a hypothetical description of how the new functionality would benefit users.

It's a good idea to include acceptance criteria that identifies what the successful completion of this work would look like. Make sure the person assigned this work item understands what's required or you could lose time to iterations that could have been avoided.

- **Assigned To:** The DevOps Center user currently responsible for the completion of the task or objective tracked by this work item.

Anyone assigned the **DevOps Center** permission set can create work items and assign them to other DevOps Center users. This approach enables anyone who finds a bug to report it or suggest an enhancement to share their idea for the rest of the team's consideration.

Work Item Statuses

As work items move through the release management cycle, the work item status is updated automatically to indicate how work is progressing.



The team can review a list of project work items and judge from the statuses how much work is done and how much is left to do.

- **New:** The initial status of a work item upon creation. The New status doesn't necessarily mean that no related work has occurred. Your team could be planning, sizing, scheduling, designing, and so on, in support of this work item. When it's time to start building customizations, however, select a development environment. The work item moves to In Progress, and enables you to launch the environment directly from the work item.
- **In Progress:** Work that the listed assignee is actively pursuing in the development environment. When in this status, another team member can't assign this item to themselves or start working on it. DevOps Center creates a branch directory for it in the project repository.
- **In Review:** Work that's ready for your team members' review. When you click **Create Review**, the work item is moved to the In Review status, and a change request is created automatically. (See [Share Your Changes for Team Member Review](#) for more about pull requests.)
- **Ready to Promote:** The work item has been reviewed and approved, and is ready to be promoted. GitHub determined that the work item doesn't have any merge conflicts and that all mergeability rules have been met.
- **Promoted:** This work item has been promoted to a pipeline stage. When it's promoted to the last pipeline stage, it's Closed.
- **Closed:** Work that's complete, reviewed, fully tested and verified, and promoted through the entire pipeline.

Create Work Items

Work items are organized by project and belong to the project where you created them.

1. From the Projects page, click the name of the project for which you're creating work items to open that project.
2. Click **New Work Item**.
3. Enter a descriptive subject.
4. Use the Description field to provide information about the scope of work, acceptance criteria, and so on.
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 to track and assign project work. All DevOps Center users can create additional work items as the project progresses.

Edit Work Item Details

Work items must be kept up to date to provide full value to the team. For example, if new acceptance criteria is identified, edit the work item to ensure the assignee knows of the change. Or if you're ready to start on an unassigned work item, assign the work item to yourself, which lets other team members know you've picked it up.

1. From the Projects page, click the name of the work item's project.

When you open a project, the Work Items tab is shown by default and lists all the current work items.

2. Locate the work item you want to edit, then click its ID to open it. Work items open in the Changes tab.
3. Edit the fields as needed.
 - In the Changes tab, choose a different development environment only if no changes have been committed for the work item from the current development environment.
 - Switch to the Details tab to edit the Subject, Description, or Assigned To fields.

Troubleshooting

You can encounter these errors while using DevOps Center. If needed, get help from your project manager or Salesforce admin to resolve these issues.

Error During Pull: “There was an error. Please reload the page or contact your system administrator if the problem persists.”

Cause: A likely cause is that you're using a sandbox without source tracking enabled. For details, see the *DevOps Center Setup Guide*.

Action: If your sandbox doesn't have source tracking enabled, follow the steps in [Track Changes Between Your Project and Org](#) in the *Salesforce DX Developer Guide*. Then refresh the sandbox.

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* in the *DevOps Center Setup Guide* for instructions on working around this issue.

Error: “The request was invalid. URL does not reference a valid SFDX project. *projectUrl*”

Cause: A project repository must contain a Salesforce DX project to work with DevOps Center. The easiest way to create a repository is to use a template we provide in GitHub to create your project repository.

Action: Log in to GitHub and use the provided repository template at github.com/forcedotcom/dx-empty to create your project repository. For details, see the *DevOps Center Setup Guide*.

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](#).