Quick Start Guide

DevOps Center

Pilot, Summer ’21

@salesforcedocs

Last updated: June 4, 2021

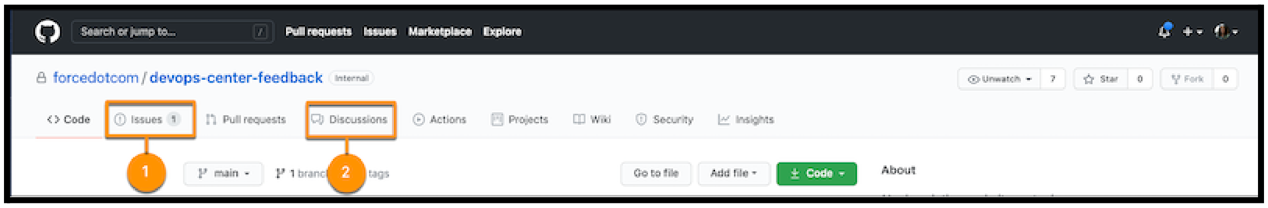
CONTENTS

Welcome to the DevOps Center Pilot

We thank you for taking the time to use DevOps Center and provide feedback. Your input will be 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.

This private shared repository is the central location for sharing information and collecting feedback on the pilot. Use the Issues tab (1) at the top of the repo to capture bugs, feature requests, and general feedback. Use the Discussions tab (2) for general discussion or questions. Access to this repository is limited to participants of the pilot and members of the Salesforce DevOps Center product team.



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 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

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 for release work include:

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

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

The team also needs a Salesforce admin to create and manage DevOps Center users, including assigning the permission sets. This user could also be the team manager user. 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*.

DevOps Center Pilot Scope

Right now, the DevOps Center experience assumes that actions related to source control and deployments are being initiated from within the DevOps Center UI. Specifically, commits, pull requests, and merges that occur from outside of DevOps Center are not reflected in DevOps Center UI, and will likely cause unexpected behavior. Similarly, deployments that are performed outside of DevOps Center are not reflected in the DevOps Center UI. In a subsequent release, we plan to support this type of "hybrid" development model where actions can happen from inside or outside of DevOps Center and will be properly reflected throughout the experience.

Get Set Up with GitHub and the Repository

You *must* have a cloud-based, GitHub-hosted, github.com account to use the DevOps Center in this pilot release. 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

Reach out to your team manager if you haven't received instructions regarding the tasks to perform to complete setup. We assume that you already completed these tasks before you start working with DevOps Center.

Open DevOps Center

Before you start: Locate the *Welcome to Salesforce* email generated when your team manager added you as a user to the DevOps Center org.

1. Use the link in the *Welcome to Salesforce* email you received to log in to the DevOps Center org.
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

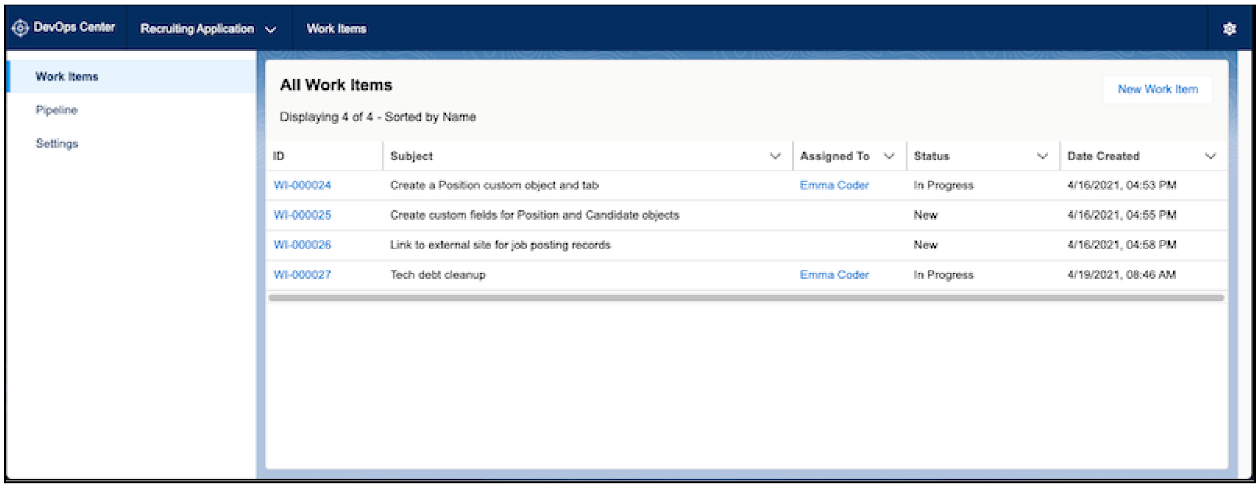
A team uses *work items* to track the progress of changes created to achieve a specific objective or task, such as enabling a user story, creating a new feature, or addressing a bug. Managing releases through work items makes it easier to track the progress of the overall release and identify areas of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

When you open a project for the first time, you might 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'll see them listed here. Both you and other team members can create additional work items as the project progresses.

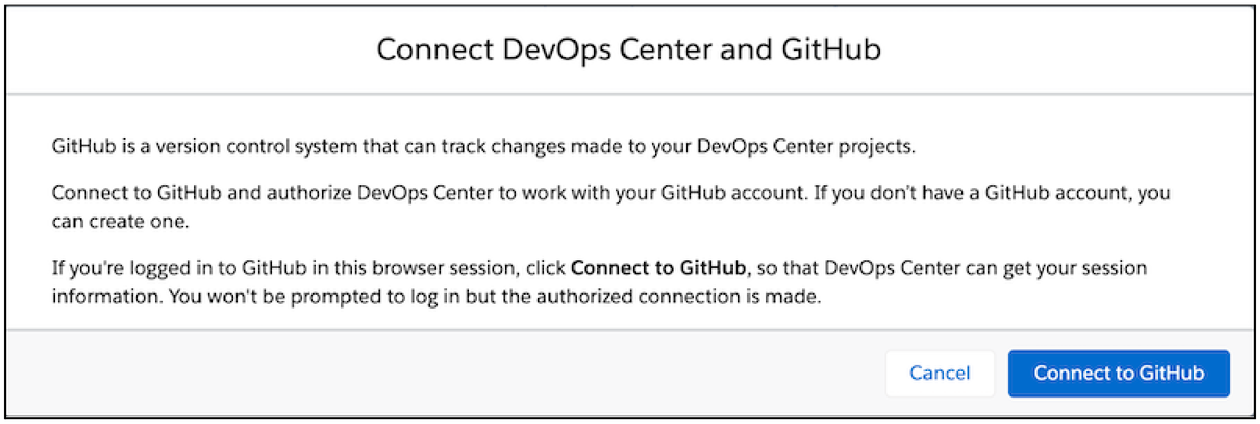


All Work Items

Displaying 4 of 4 - Sorted by Name

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.

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

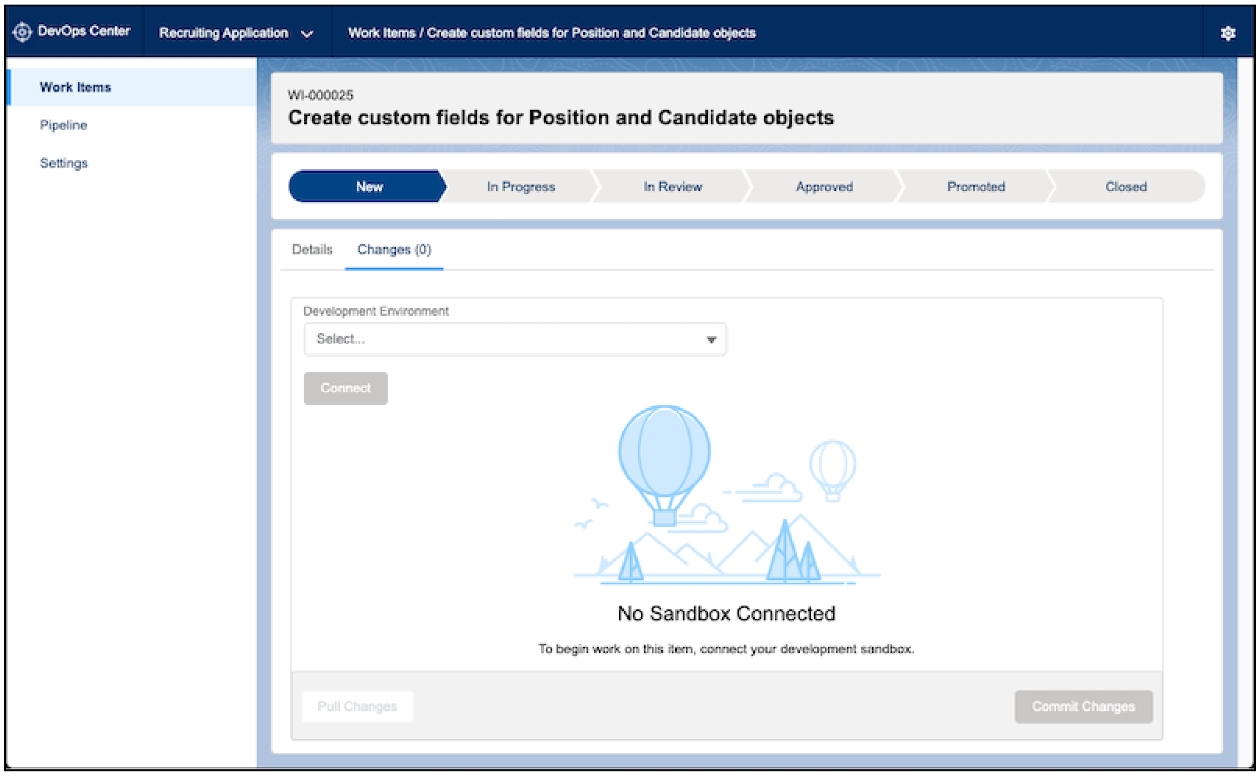
When you're ready to begin work, connect to the environment you want to use for this work item. For background information on work items and work item stages, see Manage Work Items.

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.

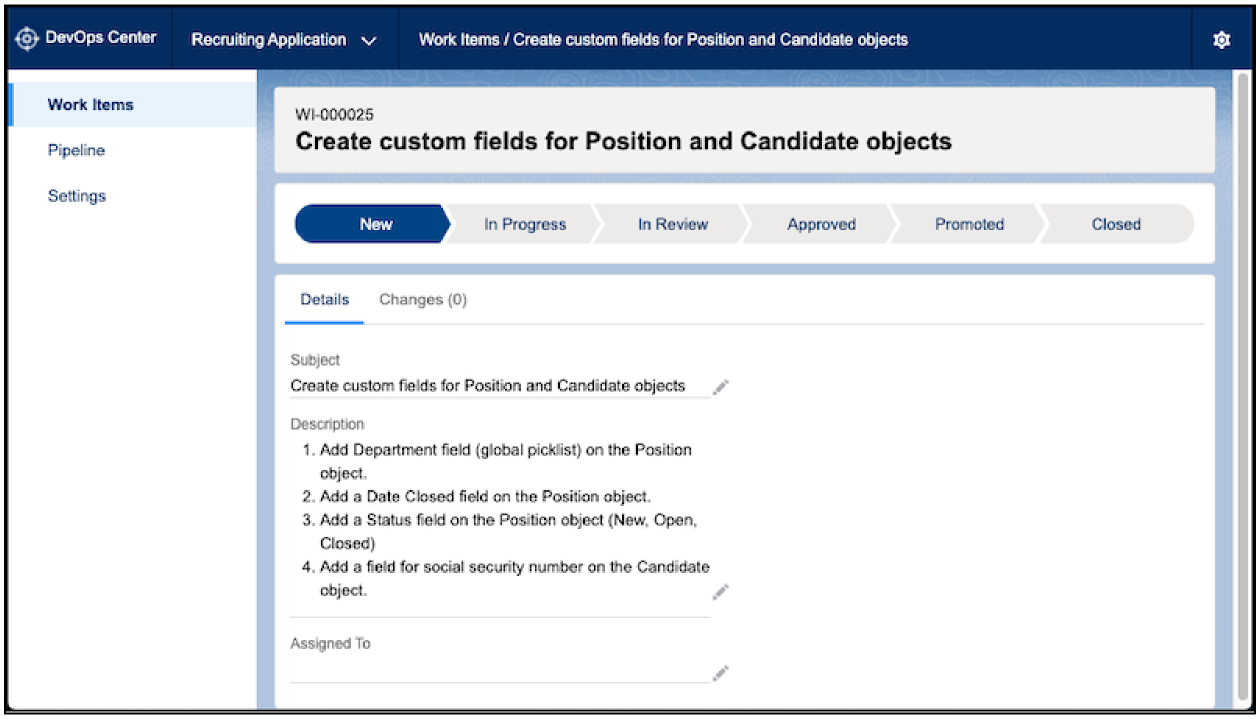
Note: Go to the Settings page in DevOps Center to see the nickname and URL for each environment to confirm you're connecting to the correct environment.

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.



WI-000025

Create custom fields for Position and Candidate objects

Subject

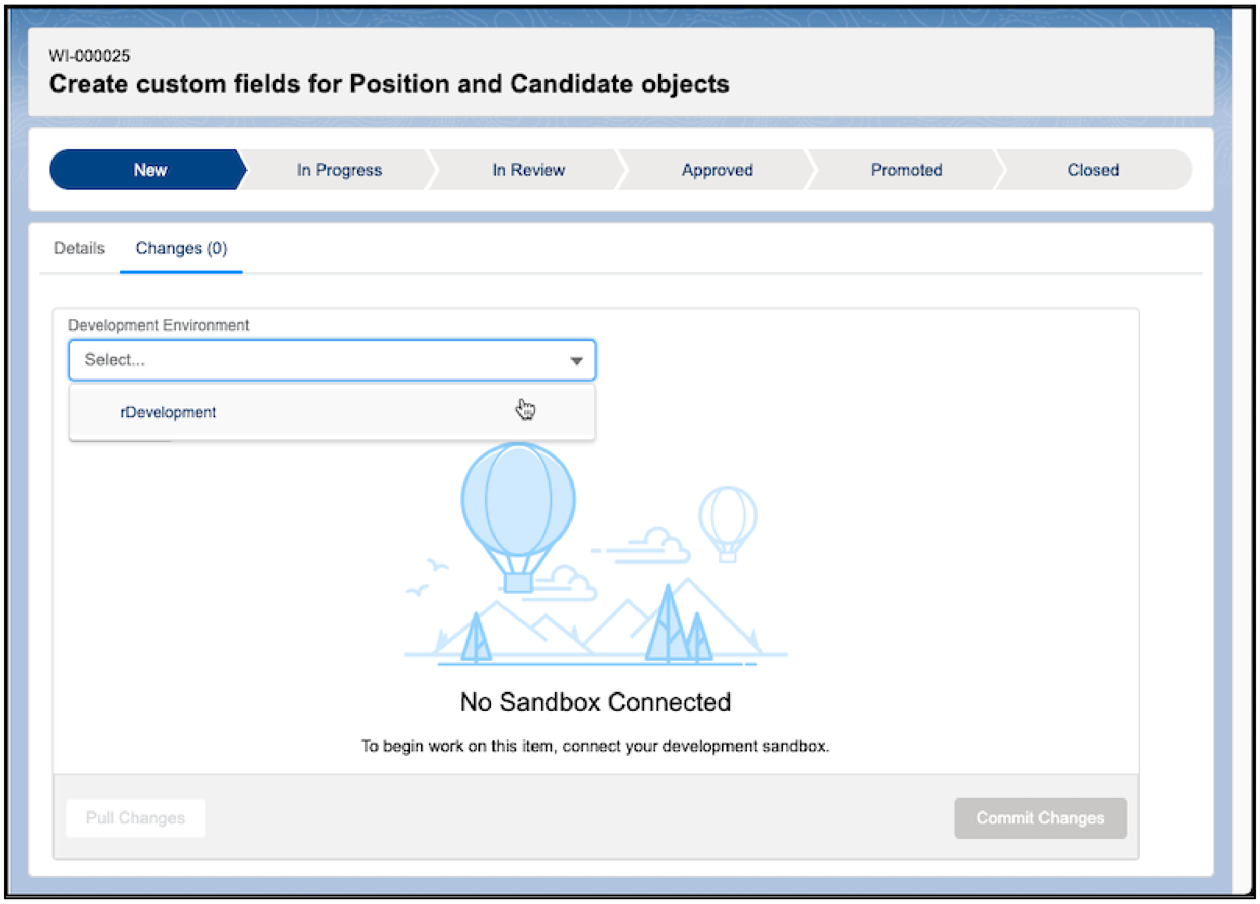
Create custom fields for Position and Candidate objects

Description

1. Add Department field (global picklist) on the Position object.
2. Add a Date Closed field on the Position object.
3. Add a Status field on the Position object (New, Open, Closed)
4. Add a field for social security number on the Candidate object.

Assigned To

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


WI-000025

Create custom fields for Position and Candidate objects

Development Environment

Select...

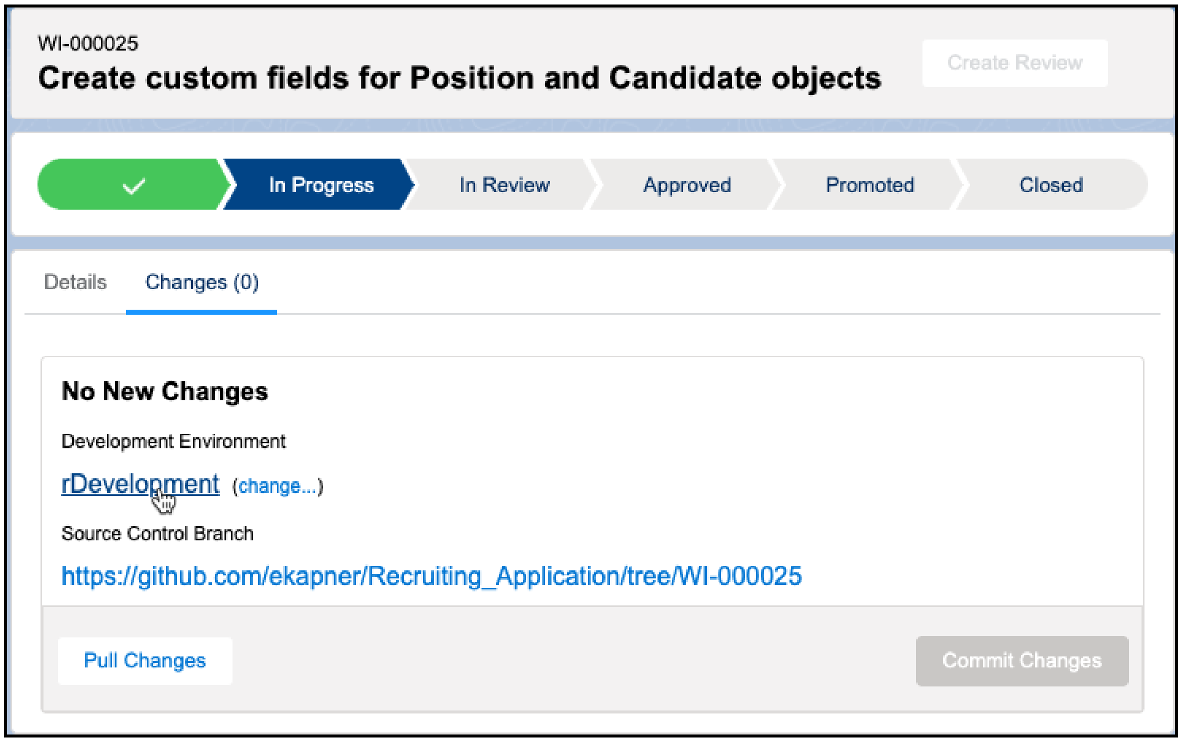
rDevelopment

No Sandbox Connected

To begin work on this item, connect your development sandbox.

4. Click **Connect** to log in. After you connect to a development environment, the work item moves to the In Progress stage. 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.


WI-000025

Create custom fields for Position and Candidate objects

No New Changes

Development Environment

rDevelopment (change...)

Source Control Branch

https://github.com/ekapner/Recruiting_Application/tree/WI-000025

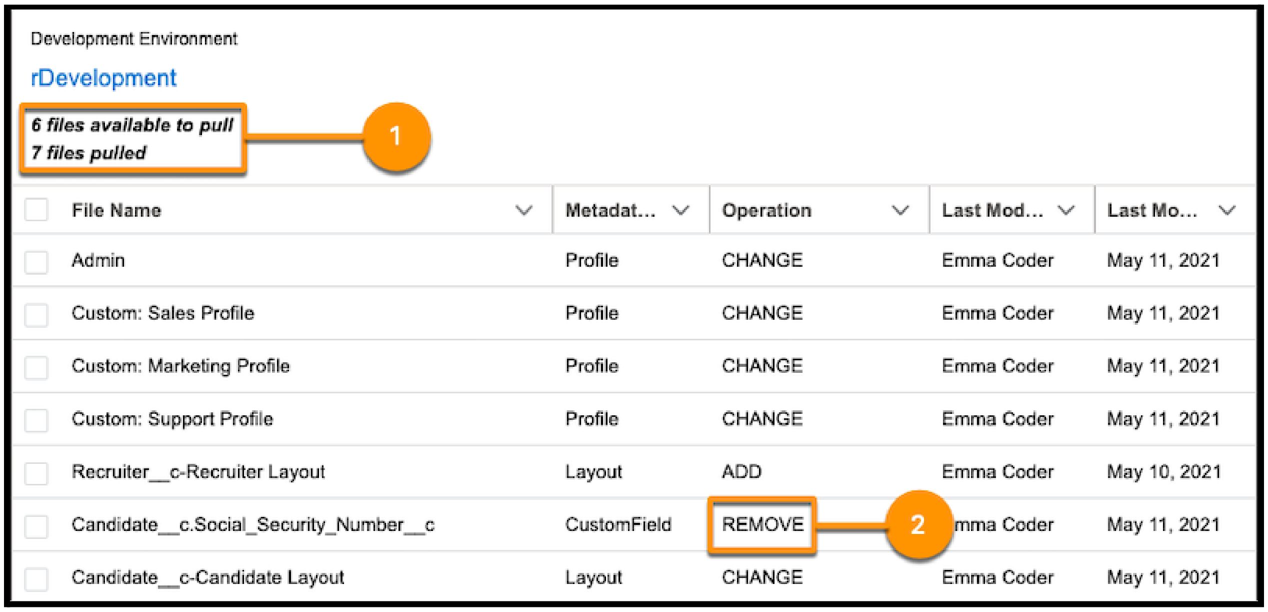
View and Pull Changes from the Development Environment

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

	File Name	Metadat...	Operation	Last Mod...	Last Mo...
☐	Admin	Profile	CHANGE	Emma Coder	May 11, 2021
☐	Custom: Sales Profile	Profile	CHANGE	Emma Coder	May 11, 2021
☐	Custom: Marketing Profile	Profile	CHANGE	Emma Coder	May 11, 2021
☐	Custom: Support Profile	Profile	CHANGE	Emma Coder	May 11, 2021
☐	Recruiter__c-Recruiter Layout	Layout	ADD	Emma Coder	May 10, 2021
☐	Candidate__c.Social_Security_Number__c	CustomField	REMOVE	mma Coder	May 11, 2021
☐	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 is not 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, if you found a Profile component file in the list that wasn't part of your intended changes for the work item, don't select it to leave it out of your commit.

   Note: When you don't include files, they are not 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 Changes to the Project Repository

After you are 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 might be similar to:

`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.

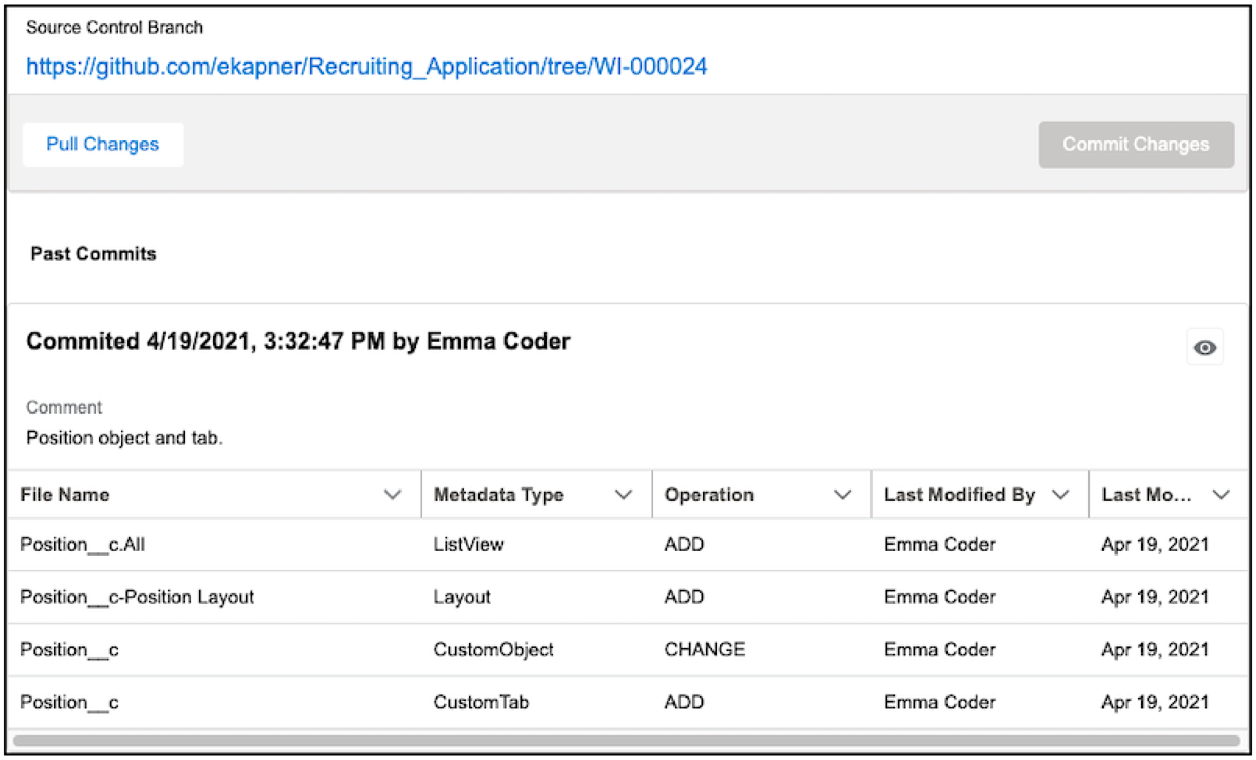
   Include what's distinctive about the changes, so you can identify an individual commit by more than just its timestamp. A single work item can have multiple commits. You can continue to make changes in your development environment and commit a new group of changes for the same work item.

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 in your development environment is reset. We reset the source tracking, so changes that are already committed don't keep showing up when you pull changes from the development environment.

   After the page refreshes, the work item's committed changes are listed at the bottom of the work item's Changes tab, most recent to oldest if there's more than one.


Source Control Branch
https://github.com/ekapner/Recruiting_Application/tree/WI-000024
Pull Changes
Commit Changes
Past Commits

Commited 4/19/2021, 3:32:47 PM by Emma Coder

Comment
Position object and tab.

File Name	Metadata Type	Operation	Last Modified By	Last Mo...
Position__c.All	ListView	ADD	Emma Coder	Apr 19, 2021
Position__c-Position Layout	Layout	ADD	Emma Coder	Apr 19, 2021
Position__c	CustomObject	CHANGE	Emma Coder	Apr 19, 2021
Position__c	CustomTab	ADD	Emma Coder	Apr 19, 2021

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 just committed in the source control repository.

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 actually been retrieved out of the development environment yet. The file isn't actually retrieved from the development environment until it's committed.

Once 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 may have ramifications on the other. Examples are listed below:

- 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 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 may 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 might 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 have been committed and are 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, the team member needs to review 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.

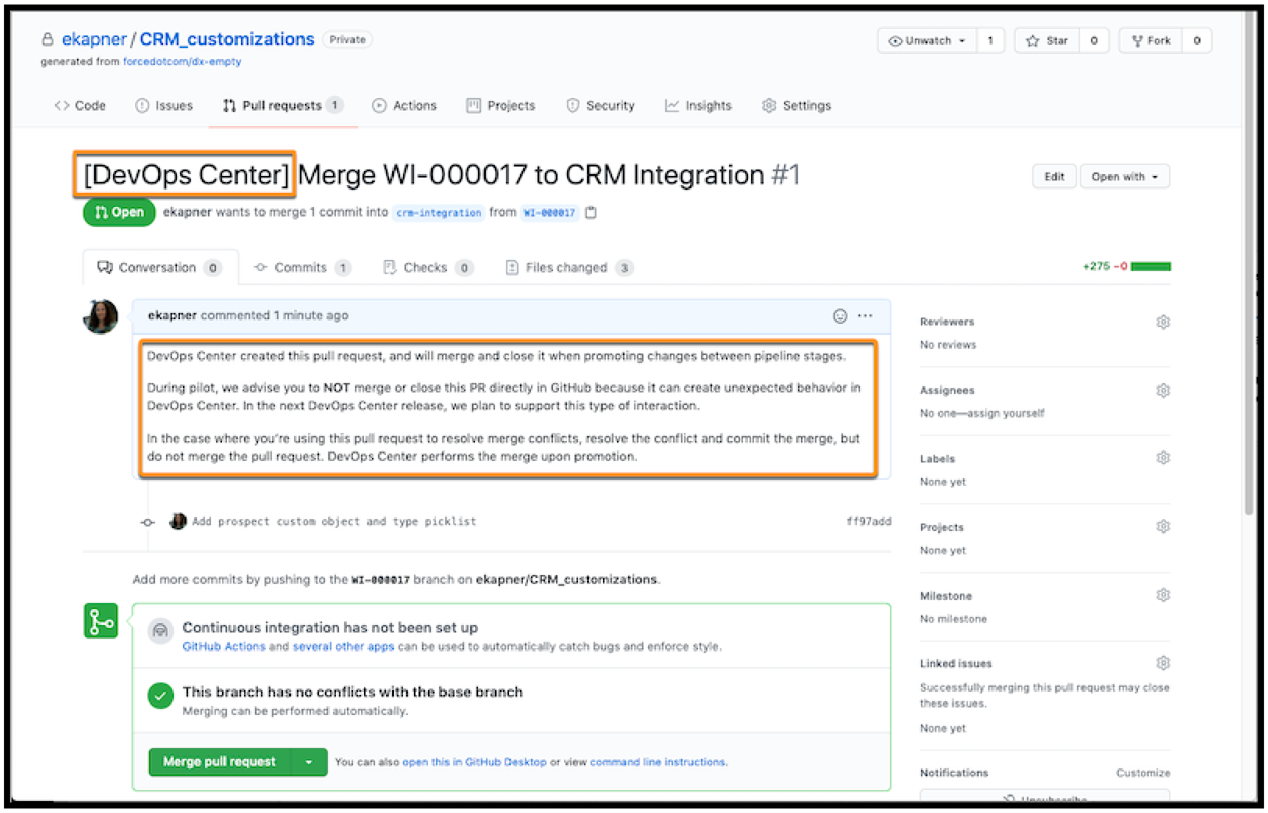
1. From the work item, click **Create Review** to move the work item to In Review.

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

2. 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.

Warning: In general, we don't recommend making changes directly in GitHub that affect what DevOps Center displays. Changes you make in GitHub, such as approving, merging, or closing a pull request aren't reflected in the work item in DevOps Center. These actions will likely cause unexpected behaviors in DevOps Center. DevOps Center merges work items (changes) automatically as part of the promotion process. We plan to support working outside of DevOps Center and directly in the source control system in a future release.

Exception: You'll want to review and comment on changes directly in GitHub, as well as review and resolve merge conflicts.



[DevOps Center] Merge WI-000017 to CRM Integration #1

Open ekapner wants to merge 1 commit into `crm-integration` from `WI-000017`

ekapner commented 1 minute ago

DevOps Center created this pull request, and will merge and close it when promoting changes between pipeline stages.

During pilot, we advise you to **NOT** merge or close this PR directly in GitHub because it can create unexpected behavior in DevOps Center. In the next DevOps Center release, we plan to support this type of interaction.

In the case where you're using this pull request to resolve merge conflicts, resolve the conflict and commit the merge, but do not merge the pull request. DevOps Center performs the merge upon promotion.

`Add prospect custom object and type picklist` `ff97add`

Add more commits by pushing to the **WI-000017** branch on **ekapner/CRM_customizations**.

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.

You can also open this in GitHub Desktop or view command line instructions.

3. Team members can add comments and have discussion in the source control pull request about the changes.

4. Go back to DevOps Center to approve the work item.

Approve Work Items

After the work item is reviewed, a team member approves the work item. Work items are not available for promotion through the pipeline until they are marked as Approved.

Work items are approved individually by clicking the **Approve?** toggle. Click the toggle again to withdraw the approval. You can revoke the approval until the work item is promoted.

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 set up 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, although 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:

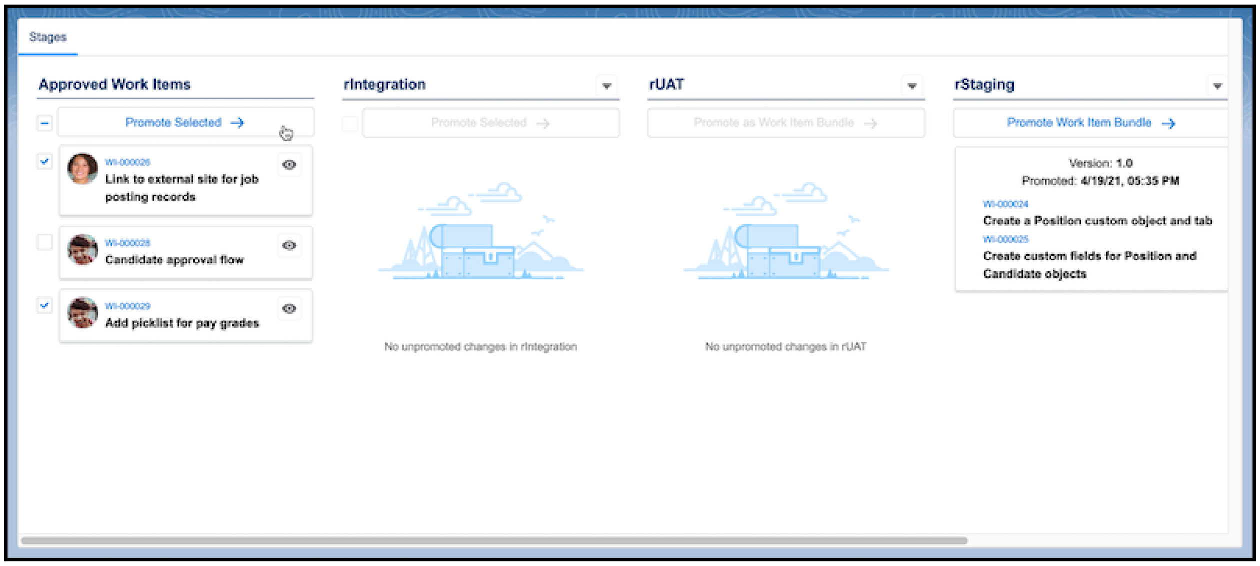
- Allow you to move work items individually through the entire pipeline.
- Allow you to move work items individually in early stages of the pipeline, and as a work item bundle in later stages. The point in the pipeline where you transition from the more flexible/individually-selectable work item promotion 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 have not 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(s) 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

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**.



WI-000026
Link to external site for job posting records

WI-000028
Candidate approval flow

WI-000029
Add picklist for pay grades

No unpromoted changes in rIntegration

No unpromoted changes in rUAT

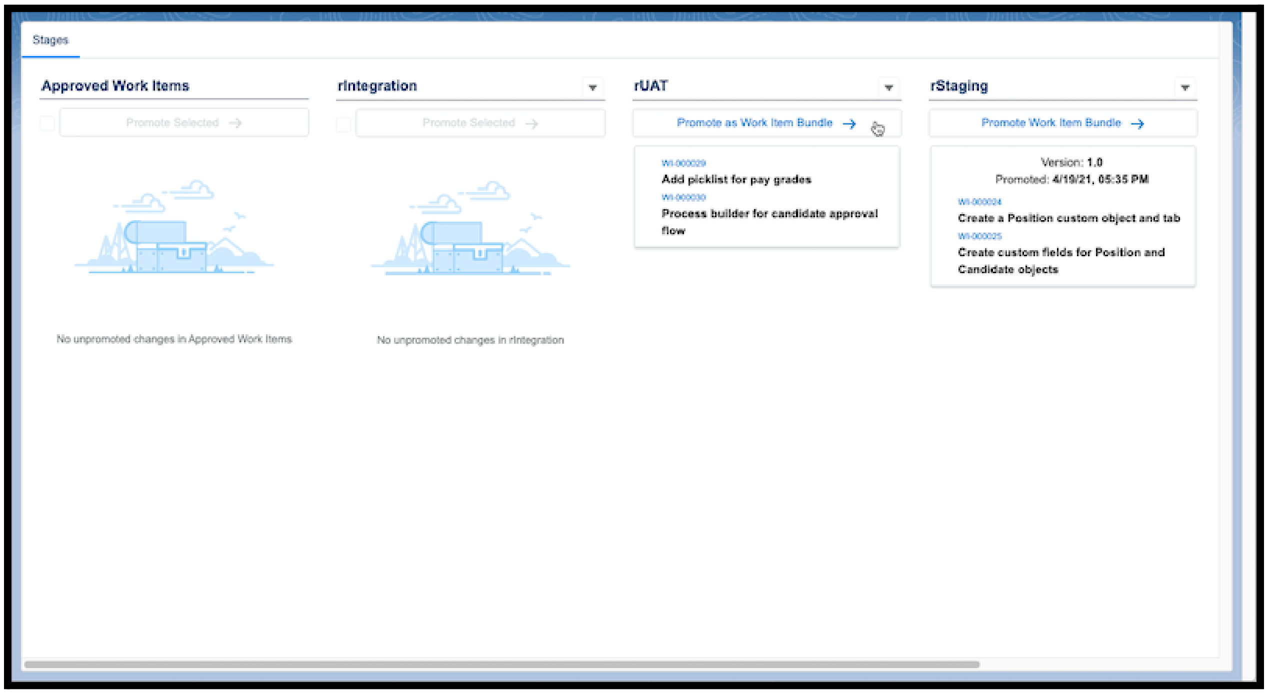
Version: 1.0
Promoted: 4/19/21, 05:35 PM
WI-000024
Create a Position custom object and tab
WI-000025
Create custom fields for Position and Candidate objects

4. If this is your first time promoting to this stage, click **Connect** to log in to the stage's environment, then repeat steps 2-3.
5. In the Promotion Options dialog, select whether to run all Apex tests.
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**.



No unpromoted changes in Approved Work Items

No unpromoted changes in rIntegration

WI-000029
Add picklist for pay grades
WI-000030
Process builder for candidate approval flow

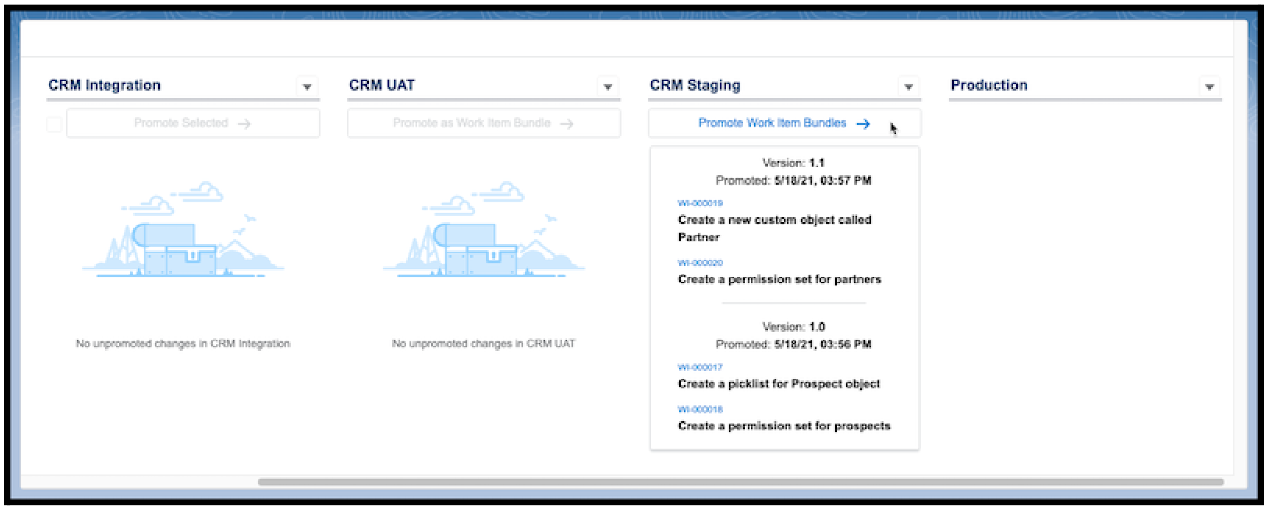
Version: 1.0
Promoted: 4/19/21, 05:35 PM
WI-000024
Create a Position custom object and tab
WI-000025
Create custom fields for Position and Candidate objects

2. If this is 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 whether to run all Apex tests.
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)**.


CRM Integration
CRM UAT
CRM Staging
Production
Promote Selected
Promote as Work Item Bundle
Promote Work Item Bundles
No unpromoted changes in CRM Integration
No unpromoted changes in CRM UAT

Version: 1.1
Promoted: 5/18/21, 03:57 PM
WI-000019
Create a new custom object called Partner
WI-000020
Create a permission set for partners

Version: 1.0
Promoted: 5/18/21, 03:56 PM
WI-000017
Create a picklist for Prospect object
WI-000018
Create a permission set for prospects

2. If this is your first time promoting to this stage, click **Connect** to log in to the stage's environment, then repeat step 1.

3. Select whether to run all Apex tests.

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.

View Promotion Status

Pilot Participants: This section is In Progress. We'll announce once we update and finalize it.

When promotion completes you'll either see a success banner or a failure banner where you can see details of the error.

The Pipeline Stage dropdown menu provides options so you can validate the promotion or dig into errors, you :

- 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 Change Request -- opens the pull request in source control so you can view the changes and troubleshoot any merge conflicts.
- 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.

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 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.

Conflict Detection and Resolution

Conflicts can occur when you are 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 of the 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 may conflict or potentially overwrite each other when they are 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. Sometimes this merge can fail because the source control system cannot determine automatically how to resolve the conflict. You can see more information on the merge conflict in the error dialog in DevOps Center, 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 reattempt the promotion in DevOps Center.

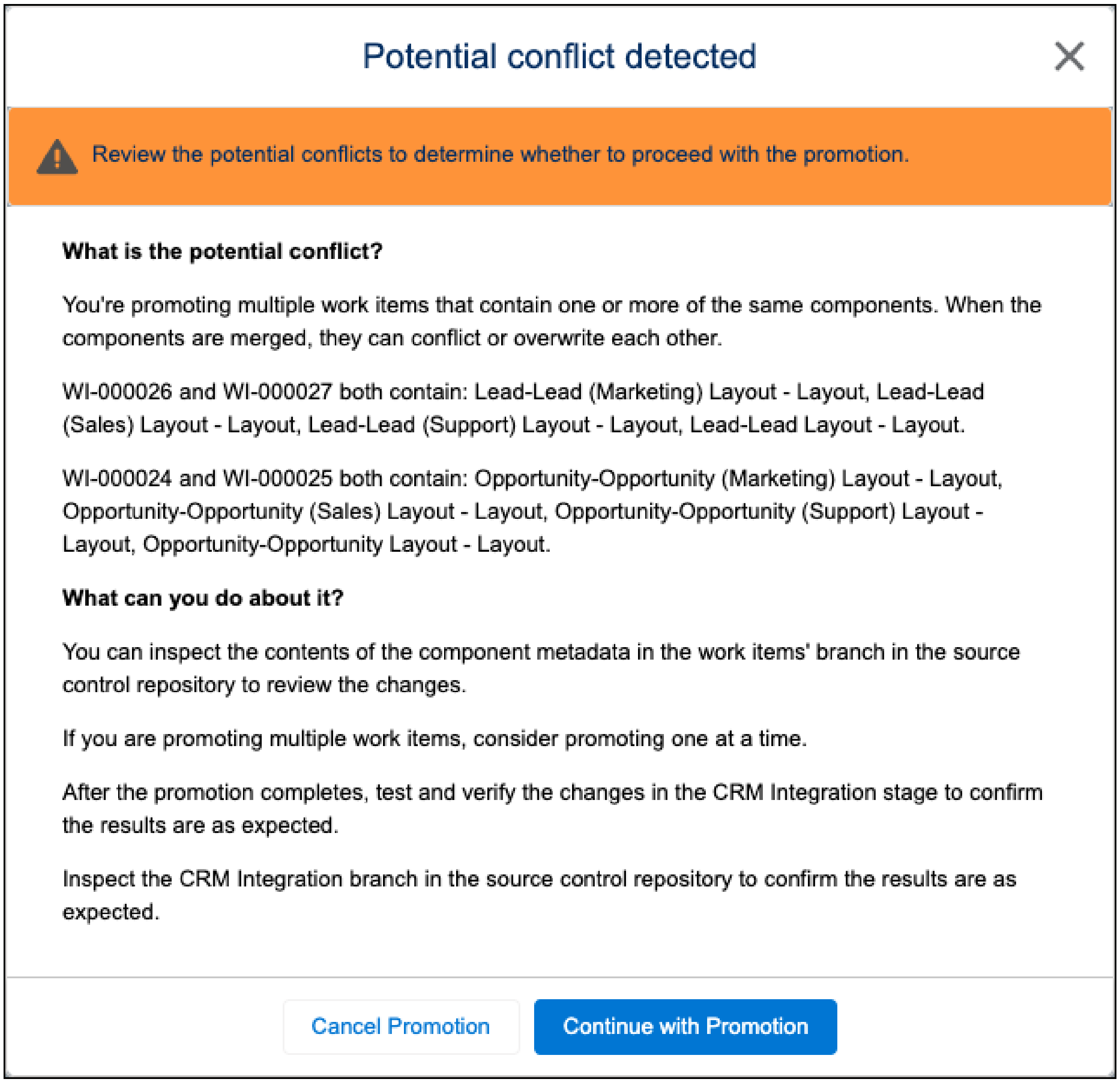
Your Component Version Is Different from Merged Component

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 objects, which causes potential merge conflicts in the page layouts.



Potential conflict detected

Review the potential conflicts to determine whether to proceed with the promotion.

What is the potential conflict?

You're promoting multiple work items that contain one or more of the same components. When the components are merged, they can conflict or overwrite each other.

WI-000026 and WI-000027 both contain: Lead-Lead (Marketing) Layout - Layout, Lead-Lead (Sales) Layout - Layout, Lead-Lead (Support) Layout - Layout, Lead-Lead Layout - Layout.

WI-000024 and WI-000025 both contain: Opportunity-Opportunity (Marketing) Layout - Layout, Opportunity-Opportunity (Sales) Layout - Layout, Opportunity-Opportunity (Support) Layout - Layout, Opportunity-Opportunity Layout - Layout.

What can you do about it?

You can inspect the contents of the component metadata in the work items' branch in the source control repository to review the changes.

If you are promoting multiple work items, consider promoting one at a time.

After the promotion completes, test and verify the changes in the CRM Integration stage to confirm the results are as expected.

Inspect the CRM Integration branch in the source control repository to confirm the results are as expected.

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 won't overwrite each other.

- **Cancel the Promotion**
  Based on the conflict description, you're unsure that the changes won't overwrite each other, or you'd like to look at the pull request for the work item, which will indicate 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 is a merge conflict.

If the Promotion Fails

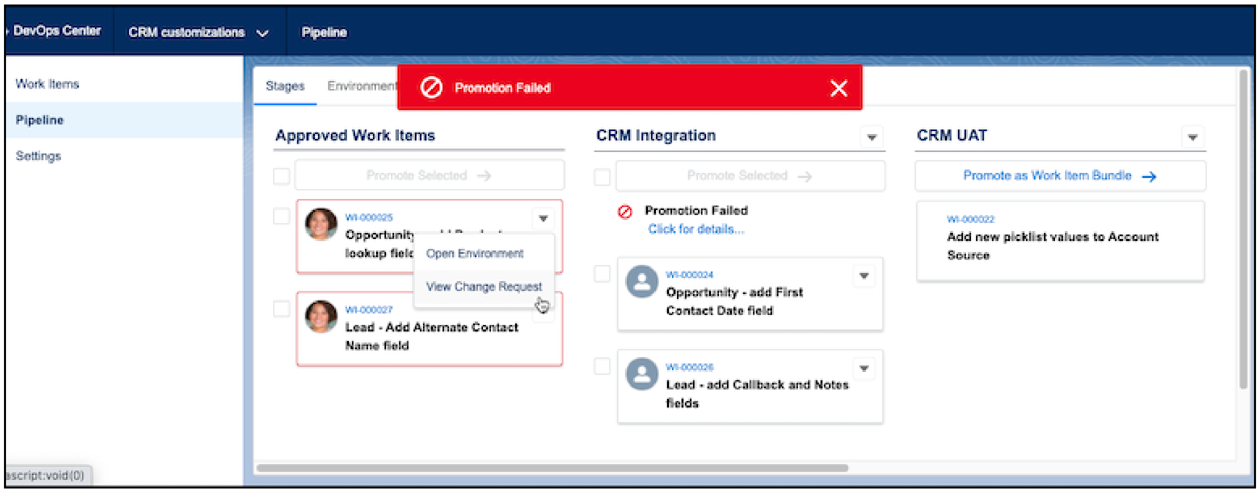
If the promotion fails, here are some options:

- 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.

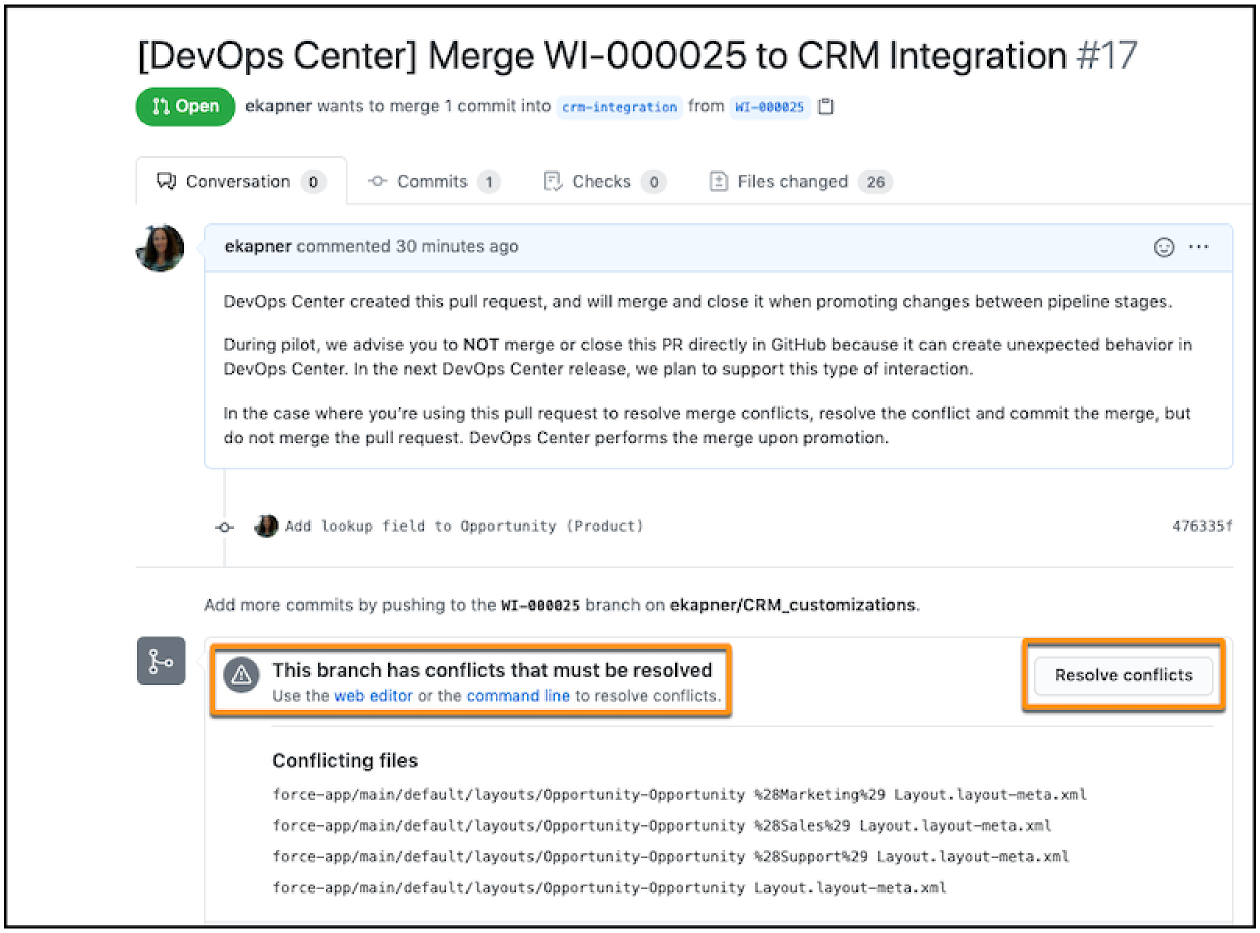
Review and Address Conflicts in GitHub

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 are promoting from), select **View Change Request** to go into GitHub to view the pull request and see the merge conflicts. You'll need to repeat these steps for all work items where the promotion failed.



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



[DevOps Center] Merge WI-000025 to CRM Integration #17

Open ekapner wants to merge 1 commit into `crm-integration` from `WI-000025`

ekapner commented 30 minutes ago

DevOps Center created this pull request, and will merge and close it when promoting changes between pipeline stages.

During pilot, we advise you to **NOT** merge or close this PR directly in GitHub because it can create unexpected behavior in DevOps Center. In the next DevOps Center release, we plan to support this type of interaction.

In the case where you're using this pull request to resolve merge conflicts, resolve the conflict and commit the merge, but do not merge the pull request. DevOps Center performs the merge upon promotion.

`Add lookup field to Opportunity (Product)` 476335f

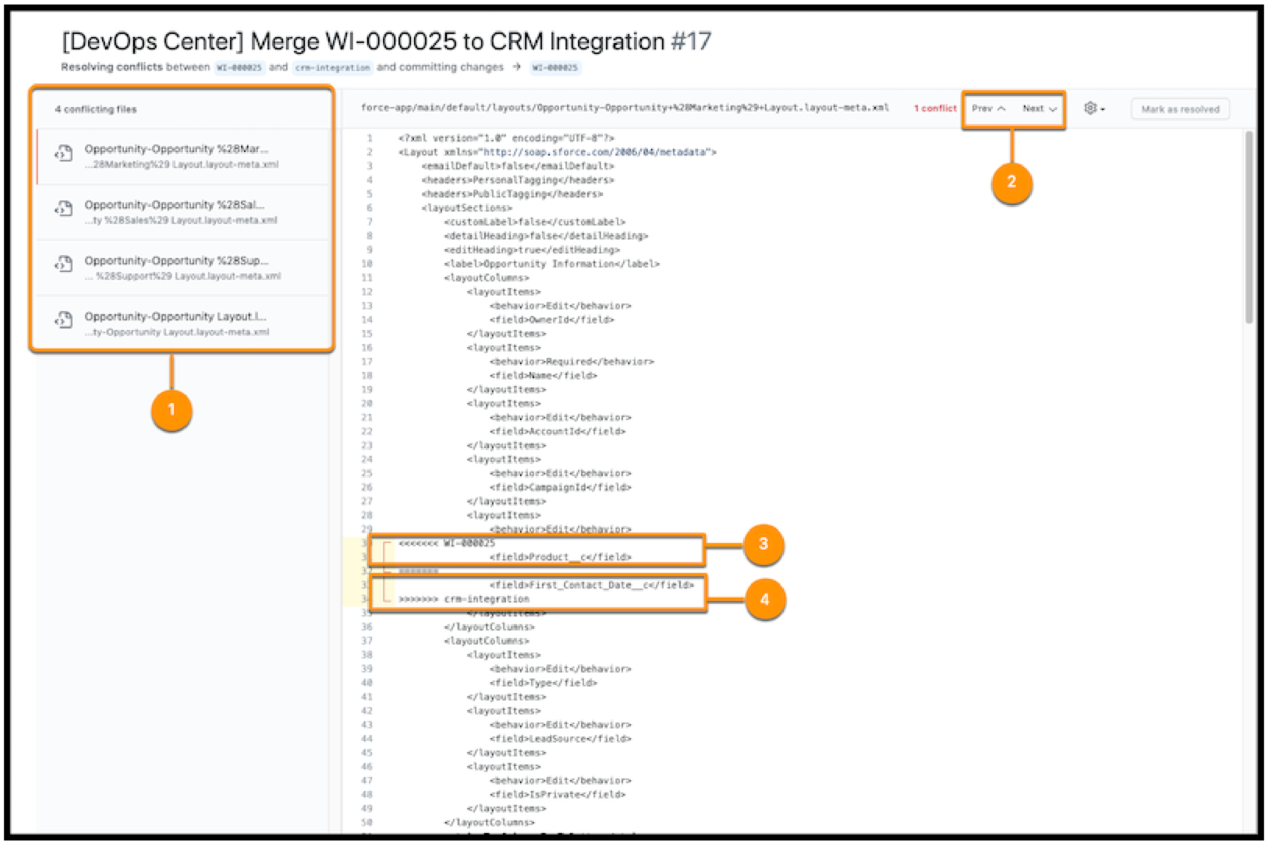
Add more commits by pushing to the **WI-000025** branch on **ekapner/CRM_customizations**.

This branch has conflicts that must be resolved
Use the web editor or the command line to resolve conflicts.

Conflicting files

```
force-app/main/default/layouts/Opportunity-Opportunity %28Marketing%29 Layout.layout-meta.xml
force-app/main/default/layouts/Opportunity-Opportunity %28Sales%29 Layout.layout-meta.xml
force-app/main/default/layouts/Opportunity-Opportunity %28Support%29 Layout.layout-meta.xml
force-app/main/default/layouts/Opportunity-Opportunity Layout.layout-meta.xml
```

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


[DevOps Center] Merge WI-000025 to CRM Integration #17
Resolving conflicts between WI-000025 and crm-integration and committing changes → WI-000025
4 conflicting files
Opportunity-Opportunity %28Mar...
...28Marketing%29 Layout.layout-meta.xml
Opportunity-Opportunity %28Sal...
...ty %28Sales%29 Layout.layout-meta.xml
Opportunity-Opportunity %28Sup...
... %28Support%29 Layout.layout-meta.xml
Opportunity-Opportunity Layout.l...
...ty-Opportunity Layout.layout-meta.xml
force-app/main/default/layouts/Opportunity-Opportunity+%28Marketing%29+Layout.layout-meta.xml
1 conflict
Prev
Next
Mark as resolved
1
2
3
4

```
<?xml version="1.0" encoding="UTF-8"?>
<Layout xmlns="http://soap.sforce.com/2006/04/metadata">
    <emailDefault>false</emailDefault>
    <headers>PersonalTagging</headers>
    <headers>PublicTagging</headers>
    <layoutSections>
        <customLabel>false</customLabel>
        <detailHeading>false</detailHeading>
        <editHeading>true</editHeading>
        <label>Opportunity Information</label>
        <layoutColumns>
            <layoutItems>
                <behavior>Edit</behavior>
                <field>OwnerId</field>
            </layoutItems>
            <layoutItems>
                <behavior>Required</behavior>
                <field>Name</field>
            </layoutItems>
            <layoutItems>
                <behavior>Edit</behavior>
                <field>AccountId</field>
            </layoutItems>
            <layoutItems>
                <behavior>Edit</behavior>
                <field>CampaignId</field>
            </layoutItems>
            <layoutItems>
                <behavior>Edit</behavior>
<<<<<<< WI-000025
                <field>Product__c</field>
=======
                <field>First_Contact_Date__c</field>
>>>>>>> crm-integration
            </layoutItems>
        </layoutColumns>
        <layoutColumns>
            <layoutItems>
                <behavior>Edit</behavior>
                <field>Type</field>
            </layoutItems>
            <layoutItems>
                <behavior>Edit</behavior>
                <field>LeadSource</field>
            </layoutItems>
            <layoutItems>
                <behavior>Edit</behavior>
                <field>IsPrivate</field>
            </layoutItems>
        </layoutColumns>
```

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 (3) in branch WI-000025 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 WI-000025 branch doesn't contain (4), 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.

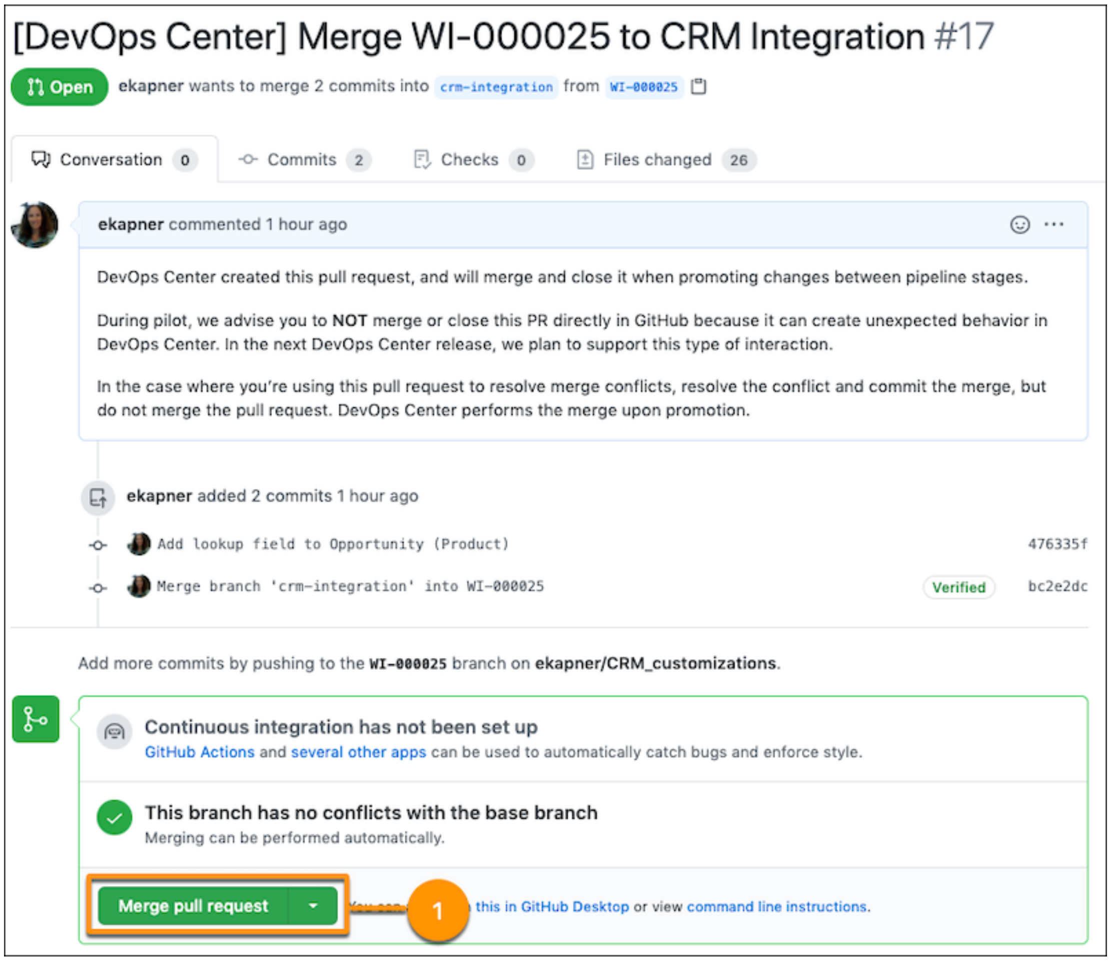
The Metadata API Developer Guide provides the schema for each metadata type to assist you with editing the files directly. In this case, we can look at "Layout" in the guide to view the schema for <`layoutItems`>.

The resulting changes now looks like this:

```
<layoutItems>
        <behavior>Edit</behavior>
         <field>Product__c</field>
</layoutItems>
<layoutItems>
        <behavior>Edit</behavior>
         <field>First_Contact_Date__c</field>
</layoutItems>
```

3. At the top of the file, click **Mark as Resolved**.
4. Click on the next file to resolve the conflict, and continue until you resolve all conflicts in the listed files.
5. Click **Commit Merge**, which saves your changes in the work item branch, W-000025.

Warning: In GitHub, don't click **Merge Pull Request** (1) to avoid unexpected behaviors in DevOps Center.



[DevOps Center] Merge WI-000025 to CRM Integration #17

Open ekapner wants to merge 2 commits into `crm-integration` from `WI-000025`

ekapner commented 1 hour ago

DevOps Center created this pull request, and will merge and close it when promoting changes between pipeline stages.

During pilot, we advise you to NOT merge or close this PR directly in GitHub because it can create unexpected behavior in DevOps Center. In the next DevOps Center release, we plan to support this type of interaction.

In the case where you're using this pull request to resolve merge conflicts, resolve the conflict and commit the merge, but do not merge the pull request. DevOps Center performs the merge upon promotion.

ekapner added 2 commits 1 hour ago

```
Add lookup field to Opportunity (Product)                476335f
Merge branch 'crm-integration' into WI-000025   Verified  bc2e2dc
```

Add more commits by pushing to the **WI-000025** branch on **ekapner/CRM_customizations**.

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.

this in GitHub Desktop or view command line instructions.

You can now close the GitHub tab and return to DevOps Center to merge the work item branch with the branch in the next stage, in this case, CRM Integration.

6. In DevOps Center, promote the work items again.

Important: Once you resolve conflicts, your development environments are out of sync with downstream pipeline environments. We recommend that you refresh your development sandboxes to get them in sync.

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 the changes a work item tracks are completed, tested, and deployed, the work item is a record the team can come back to if a question arises about those changes. Information in a work item includes:

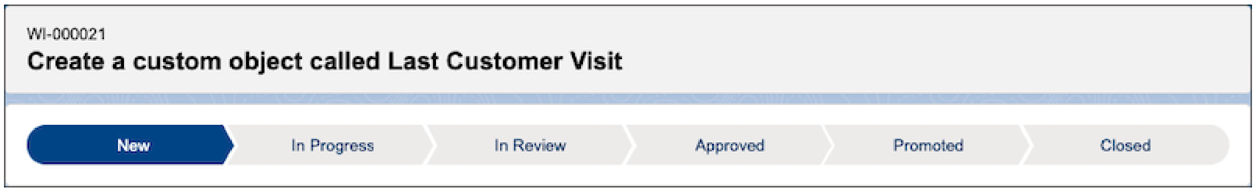
- **ID:** An automatically generated unique identifier for a work item that has a **WI-** prefix (for example, WI-12345). The ID helps DevOps Center users distinguish between similar-sounding work items. It's also used in the project repository to identify the branch where changes for the work item are 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'll 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 and anyone who wants to suggest an enhancement to share their idea for the rest of the team's consideration.

Work Item Stages

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



The team can review a list of project work items and judge from the stages 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 might 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, which moves the work item 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. Once in this stage, 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` stage, and a change request is created automatically. (See Share Your Changes for Team Member Review for more about pull requests.)
- **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 as needed 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, edit it to 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. It's likely you'll need help from your project manager or Salesforce admin to resolve these issues.

Promotion Failed with Ambiguous Metadata Deployment Error

Cause: If you selected Run Apex Tests when promoting changes, likely causes are inadequate test coverage and/or test failures. However, deployments fail for many reasons. Looking at the Deployment Status in the org can help identify the exact cause.

Action: Log in to the org. From Setup, enter `Deployment Status` in the Quick Find box, then select **Deployment Status**. Under Failed, select **View Details** next to the deployment failure.

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*.

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.