Business Central

Contribution Guide

November, 2024.

Stefan Šošić

Contents

How do contributions work?

Whether you are a developer, consultant, or user, you have the opportunity to contribute to Business Central and enhance it for everyone.

The Concept

The idea is simple: instead of everyone working in isolation on the same issues, bugs, and features, we share our implementations. This collaborative approach ensures that contributions made by one person can benefit the entire community. For instance, you might develop a solution today, and in the future, another community member might create something that you find incredibly useful.

A Brief History of Open-Source Contributions

- Early 2018: The first open-source extensions from Microsoft were released.
- May 2019: The System App was made open source.
- September 2022: The Base App followed suit.

As of now, almost all components, except for localizations and platforms, have been open-sourced, fostering a rich environment for community-driven development.

By contributing to Business Central, you become part of a dynamic ecosystem where shared knowledge and resources lead to continuous improvement and innovation.

Open source repos

System Application & Business Foundation -> https://github.com/microsoft/BCApps

Base Application -> https://github.com/microsoft/BusinessCentralApps

AL-Go -> https://github.com/microsoft/AL-Go

BcContainerHelper -> https://github.com/microsoft/navcontainerhelper

Add-On Apps (extensions) -> https://github.com/microsoft/ALAppExtensions

AL Compiler -> https://github.com/microsoft/AL

BingMaps App -> https://github.com/microsoft/bcsamples-bingmaps.appsource

Bank Import Formats -> https://github.com/microsoft/BusinessCentralBankImportFormat

Pitch new BC Idea

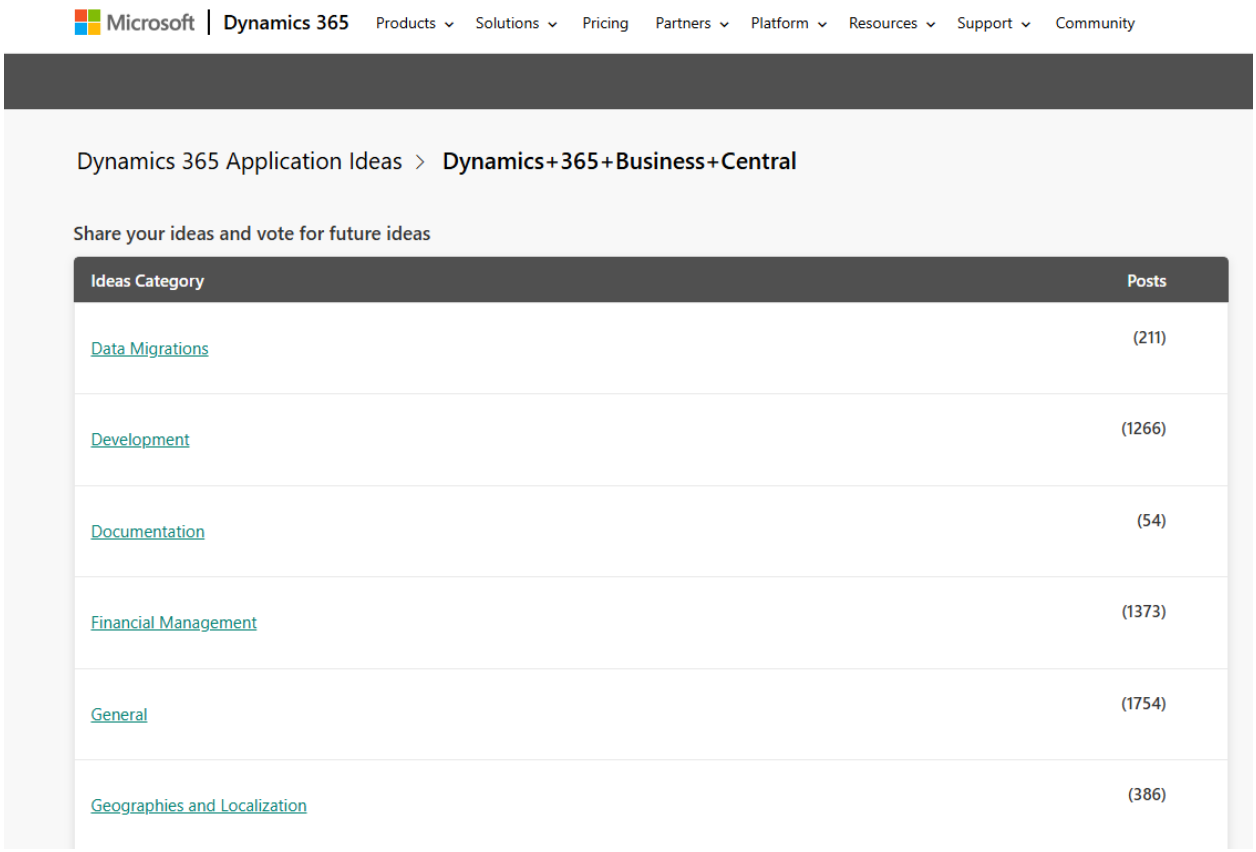
There are two ways to pitch a new BC Idea, through the Dynamics 365 Business Central forum or the GitHub repository.

Dynamics 365 Business Central Forum

The primary platform for pitching new ideas or voting on existing ones for Business Central is here. This platform is designed to gather community input and prioritize features based on user feedback.

https://aka.ms/bcideas

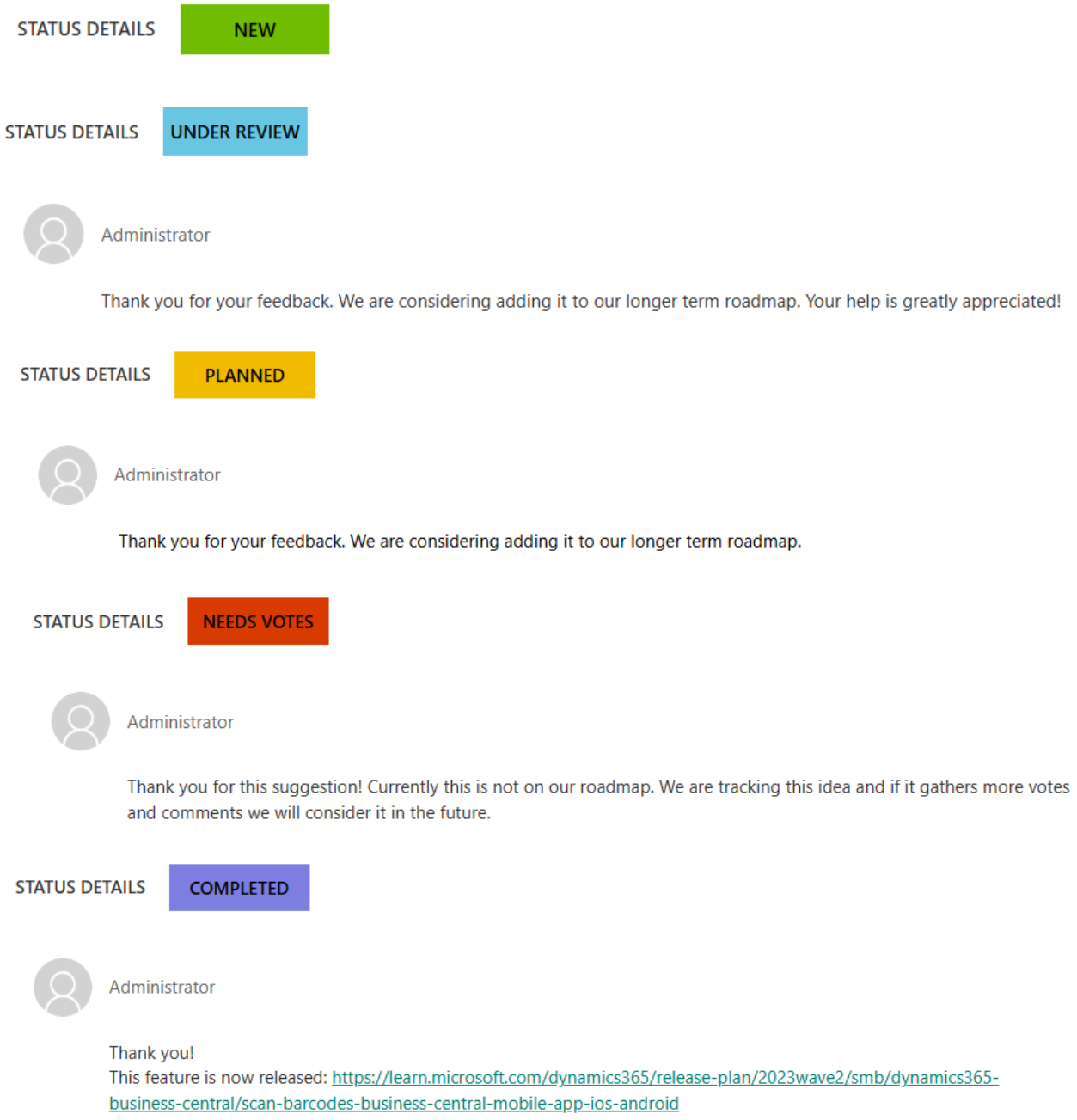
The Ideas section is organized into multiple categories to streamline the process.



By categorizing ideas, it becomes easier for users to find, pitch, and vote on ideas relevant to their needs and interests. This structured approach helps ensure that the most valuable and impactful ideas are brought to the forefront and considered for future updates.

To support an existing idea or submit a new one on the Business Central Ideas platform, you need to register. Ideas should aim to benefit the entire community, rather than being limited to individual customizations.

Idea statuses:

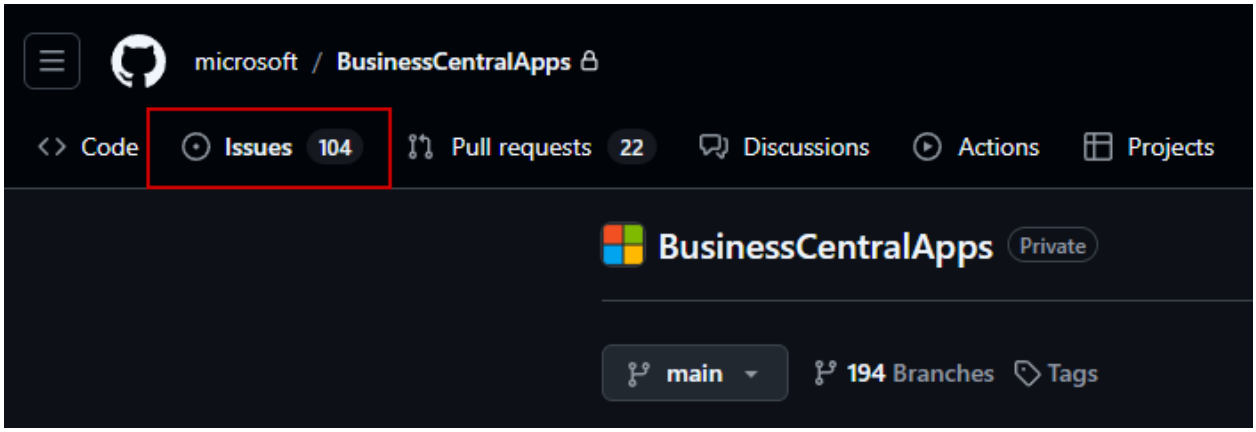


GitHub Repository

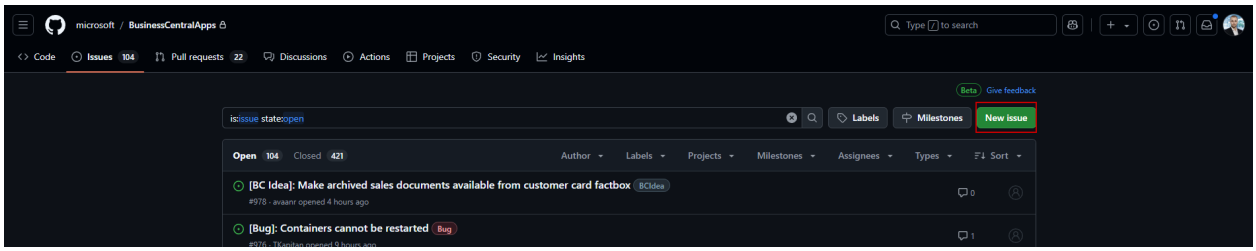
When your idea gains enough support from the community on the forum, you can then create a BC Idea directly in the repository related to that idea.

In order to do that, follow next steps:

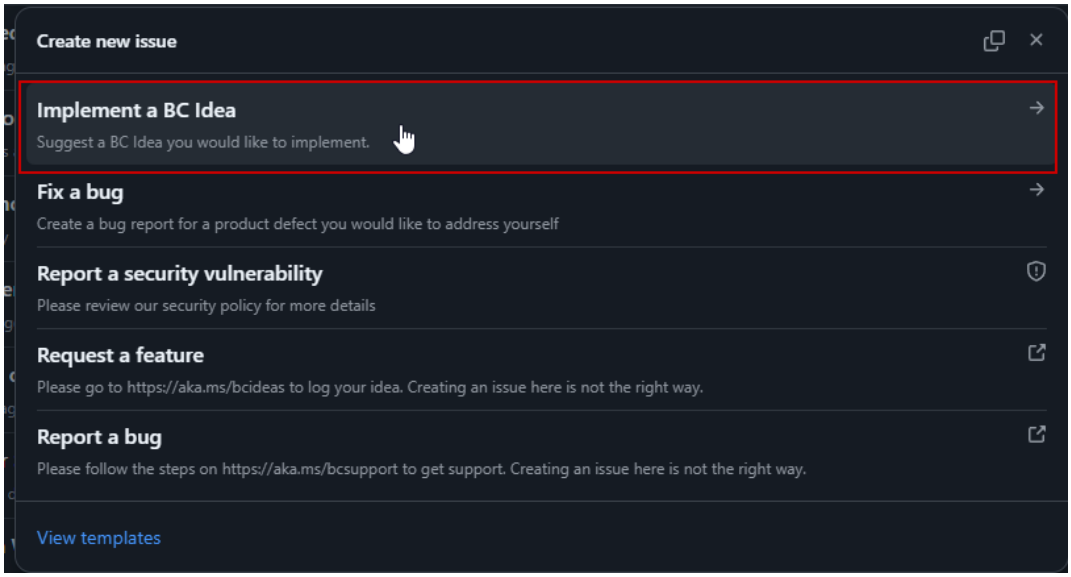
1. Go to GitHub repository



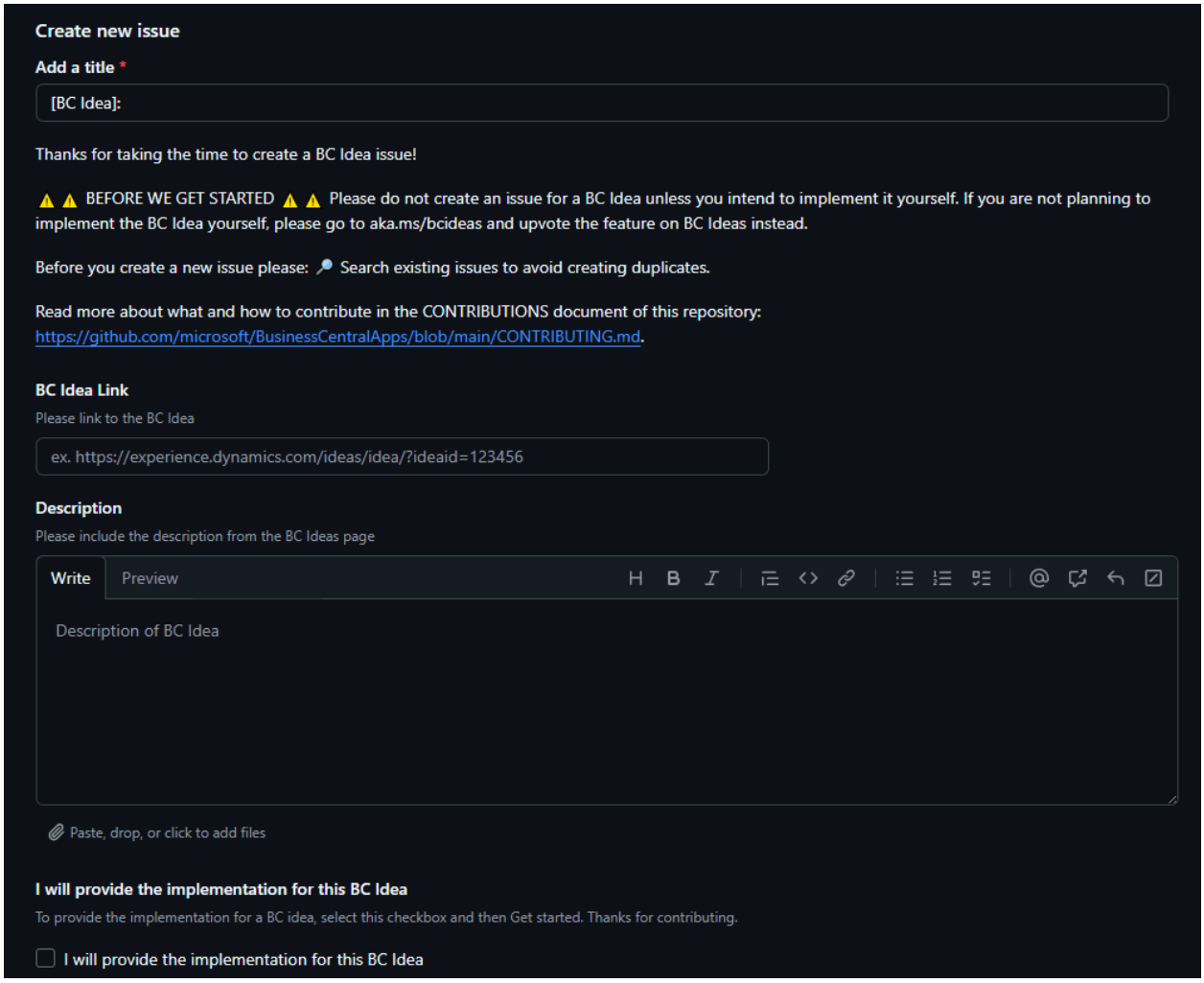
2. Navigate to the “Issues” tab.
3. Click on the “New Issue” button



4. Select “Implement a BC Idea

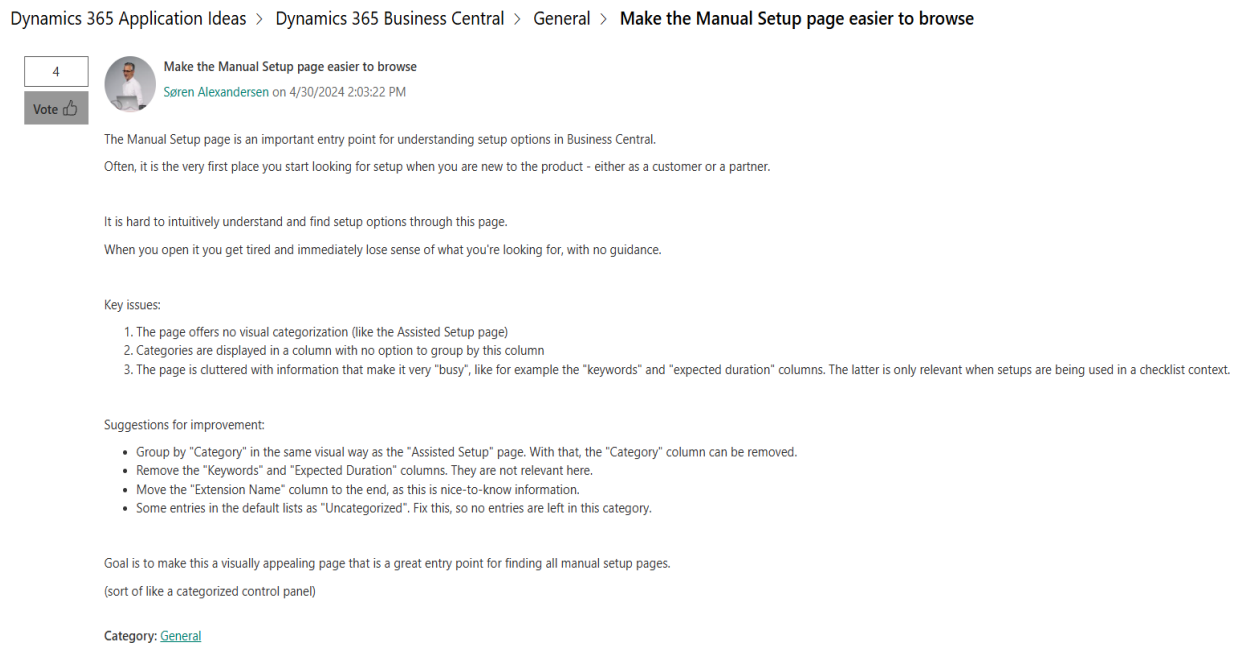


Now it opens a page where you would need to fill out additional information about BC Idea.

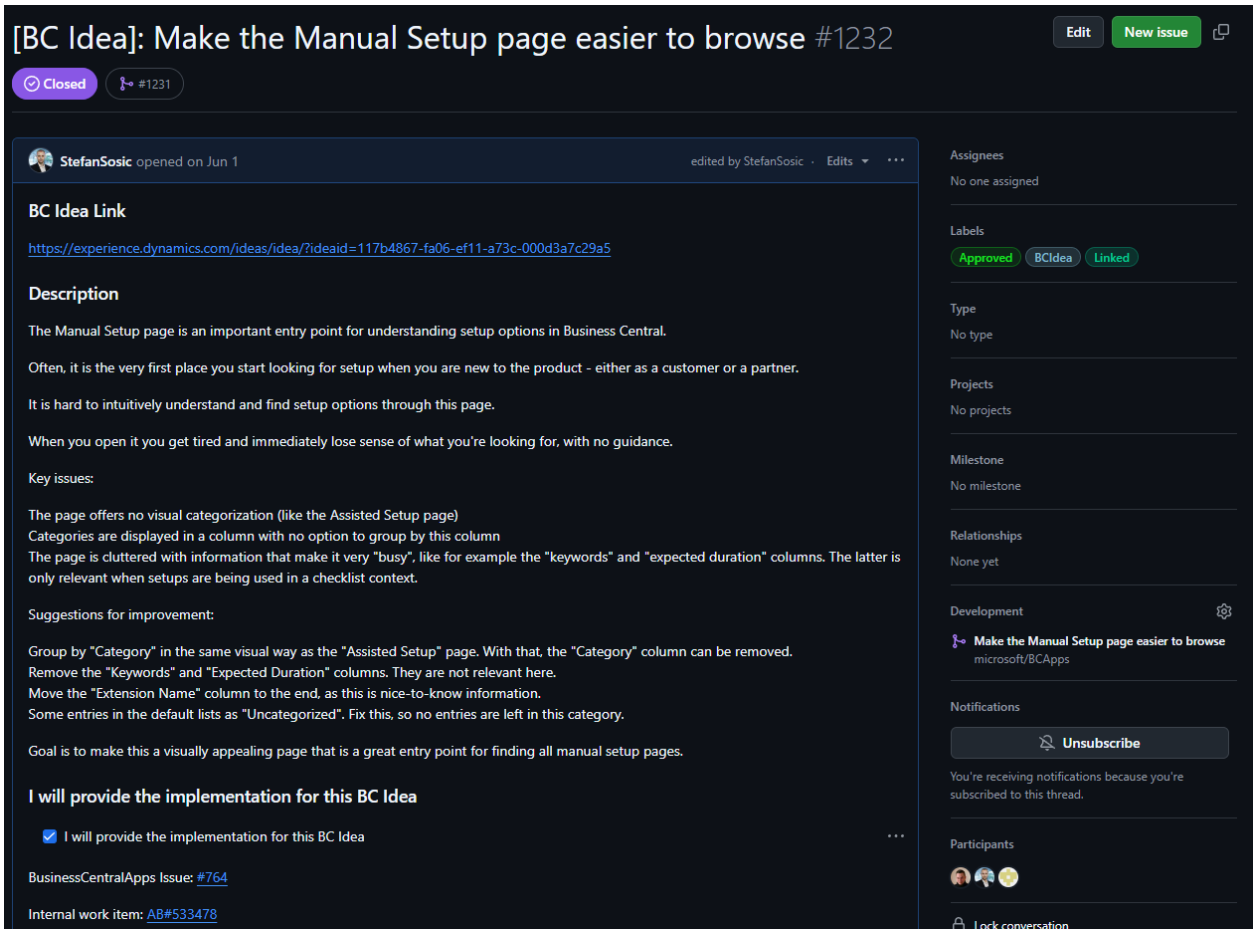

Create new issue
Add a title *
[BC Idea]:
Thanks for taking the time to create a BC Idea issue!
⚠ ⚠ BEFORE WE GET STARTED ⚠ ⚠ Please do not create an issue for a BC Idea unless you intend to implement it yourself. If you are not planning to implement the BC Idea yourself, please go to aka.ms/bcideas and upvote the feature on BC Ideas instead.
Before you create a new issue please: 🔎 Search existing issues to avoid creating duplicates.
Read more about what and how to contribute in the CONTRIBUTIONS document of this repository:
https://github.com/microsoft/BusinessCentralApps/blob/main/CONTRIBUTING.md.
BC Idea Link
Please link to the BC Idea
ex. https://experience.dynamics.com/ideas/idea/?ideaid=123456
Description
Please include the description from the BC Ideas page
Write
Preview
Description of BC Idea
Paste, drop, or click to add files
I will provide the implementation for this BC Idea
To provide the implementation for a BC idea, select this checkbox and then Get started. Thanks for contributing.
I will provide the implementation for this BC Idea

For example:

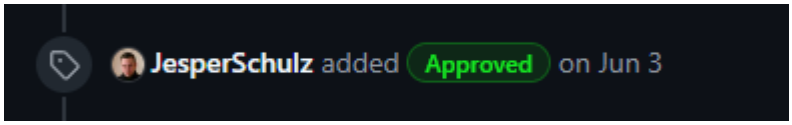
On Forum BC Idea is created:



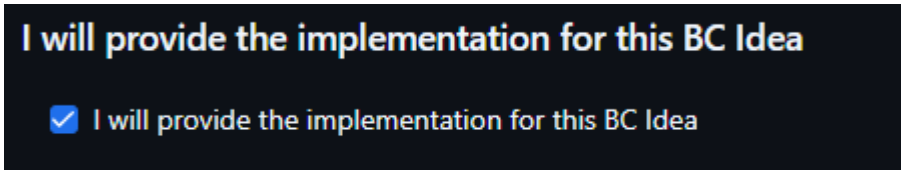
The same Idea gets created also on the responsible repository on GitHub:



You wait for if Microsoft has any questions regarding the idea, if not you get the “Approved” tag:

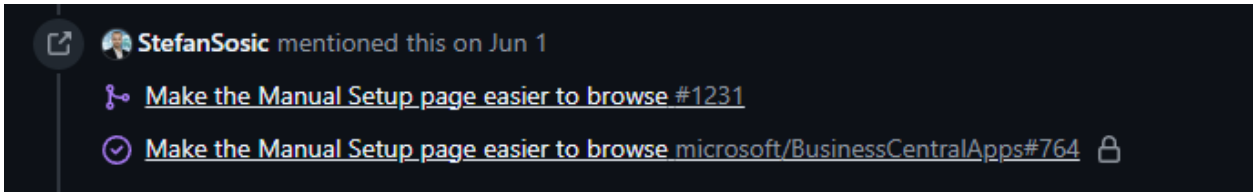


Afterward, if you ticked the checkbox indicating that you will provide the implementation of the BC idea:

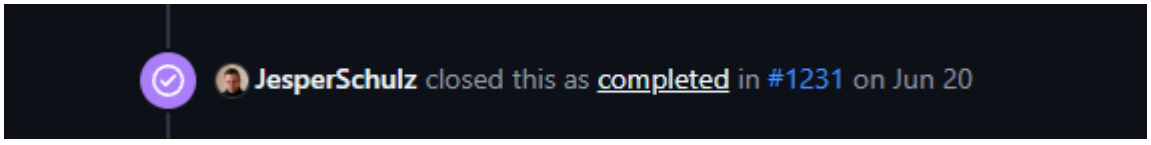


You could start to work on it, if not then some of the other community contributors will take it and provide an implementation for it.

Eventually, you will create PR which will get merged to master (Pull Request creation process among development will be covered later on in this guide):



And the issue be completed:

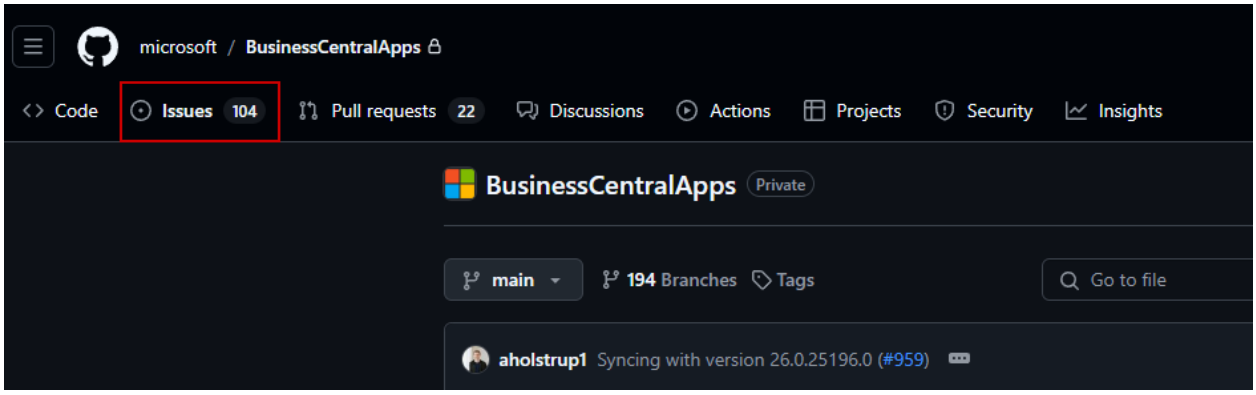


Creating a Bug

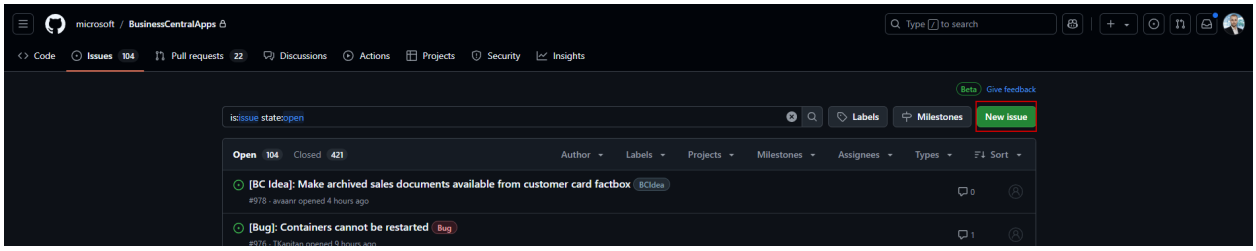
If you find the bug in Business Central lying around for some time, you could report it, get approval, and submit your fix for it.

The process of reporting a bug is quite straightforward.

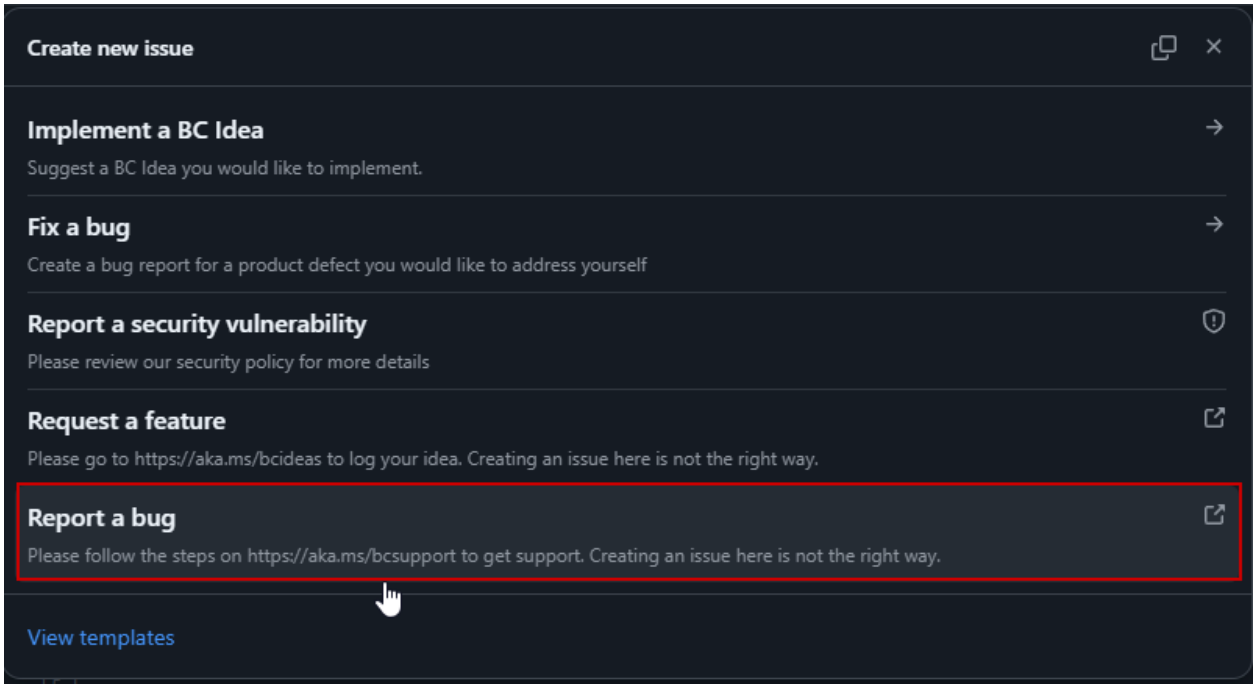
Go to the responsible repository where the bug exists and go to the “Issues” tab:



Then click on the “New Issue” button

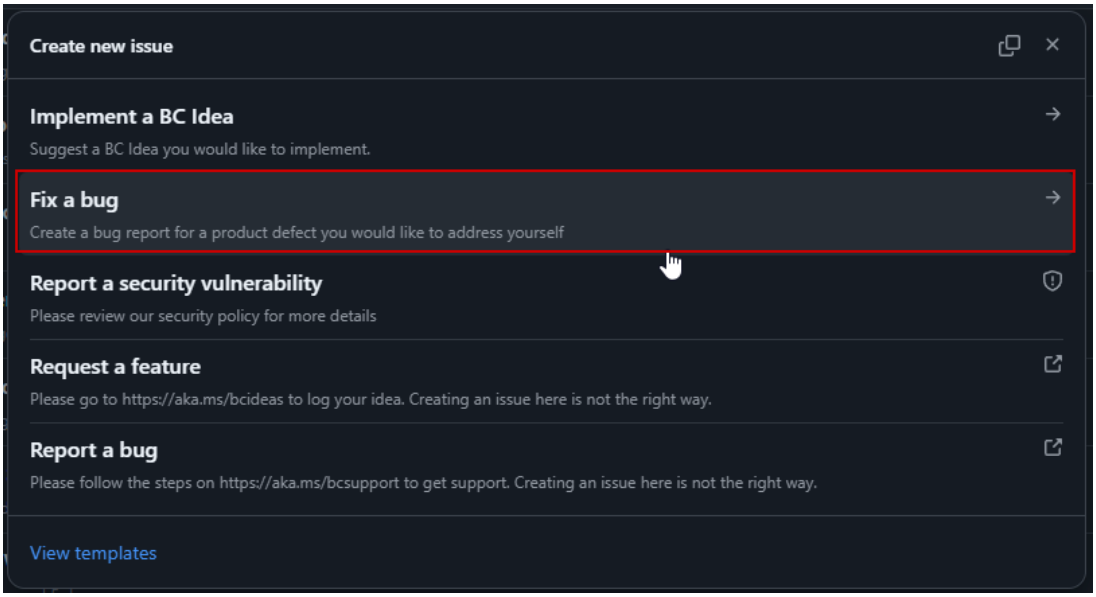


Keep in mind that it’s not the right way for all issues to create an issue on the git hub repository:

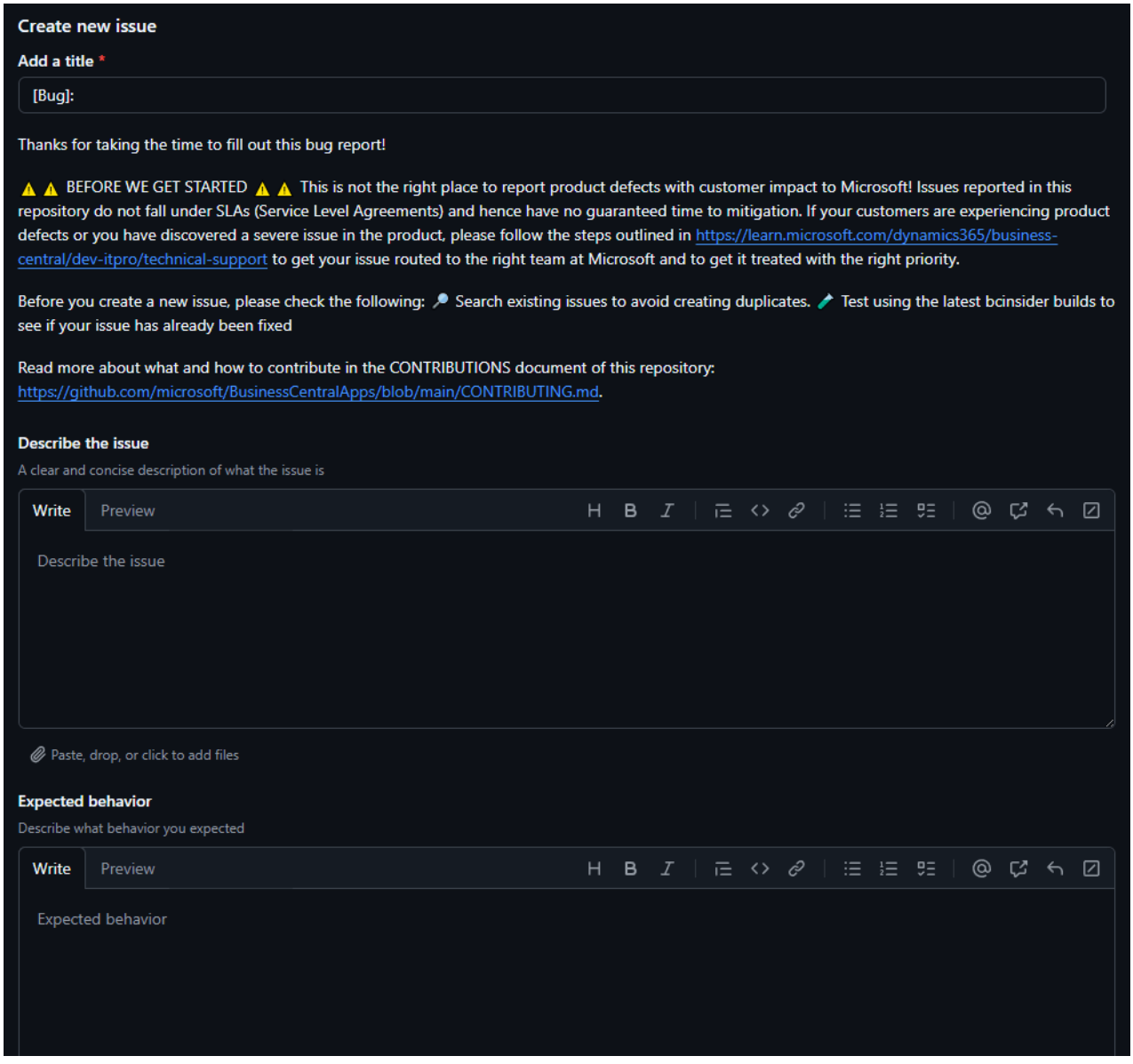


You can navigate to “Report a bug” and read more on the docs page.

We will be focusing on the bugs that are valid and we want to provide a fix for them or get community members to fix them:

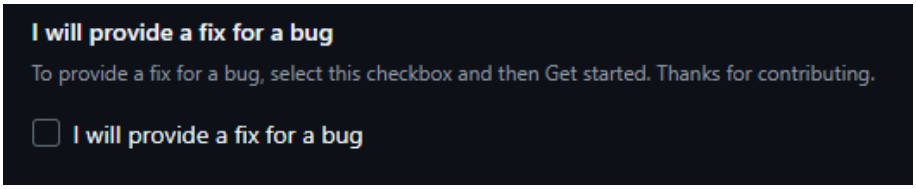


New page will open where you would need to put details for a bug which you are reporting:



It's crucial to provide as much information as possible to get approval quickly. Here's how to do it:

1. Issue Description: Clearly describe the problem and add some screenshots.
2. Expected Behavior: Explain how the feature should work.
3. Reproduction Steps: Provide exact steps to reproduce the issue. Make sure to use a clean environment without any customizations to ensure no third-party extensions are causing the problem. Screenshots or even GIFs can be helpful here.
4. Planning to Fix: If you plan to resolve the bug yourself, tick the checkbox.

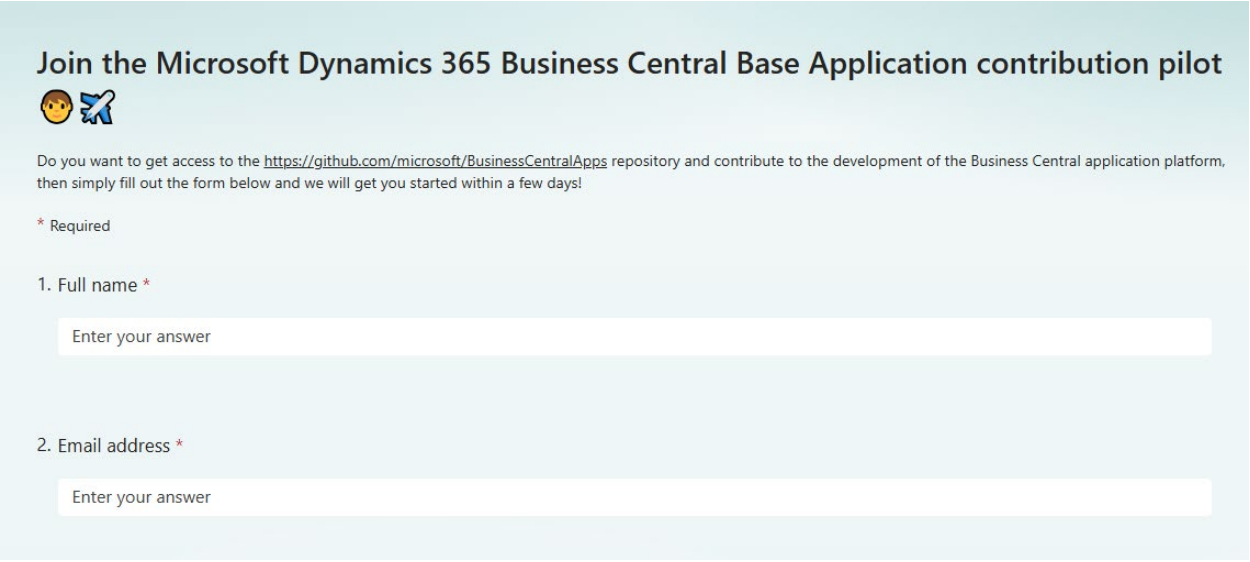


This thorough approach ensures that your bug report is clear and can be promptly addressed.

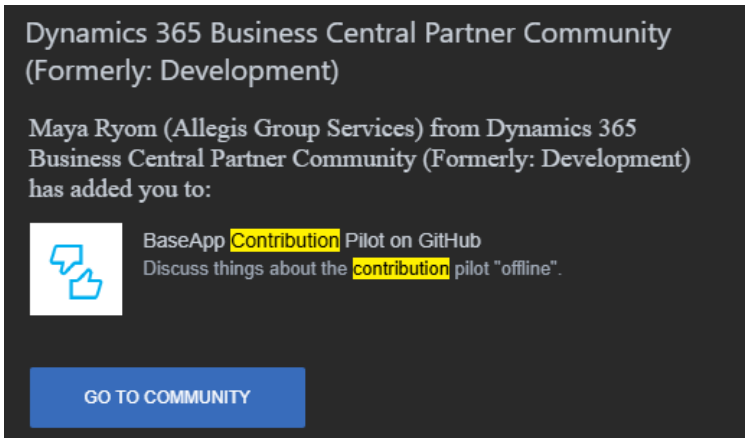
Prerequisite to contributing to Base Application

As this is a private repository, you won't see the repository without being granted access by Microsoft.

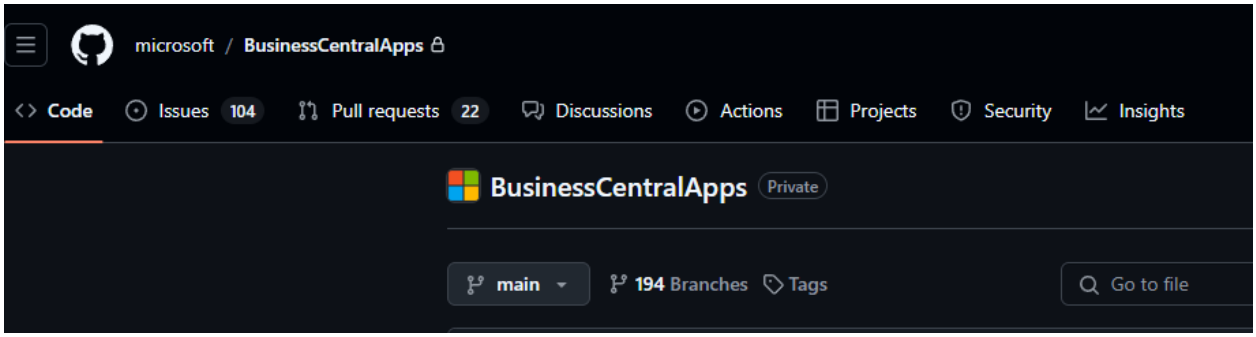
You can request access to it by filling out the following form:
https://aka.ms/JoinBCCodevelopmentPilot



Once you request is processed, you will get access also to Viva engage group:



And also the most important, GitHub repository:



Contributing as developer

If you are planning to contribute as a developer, you would need the following knowledge:

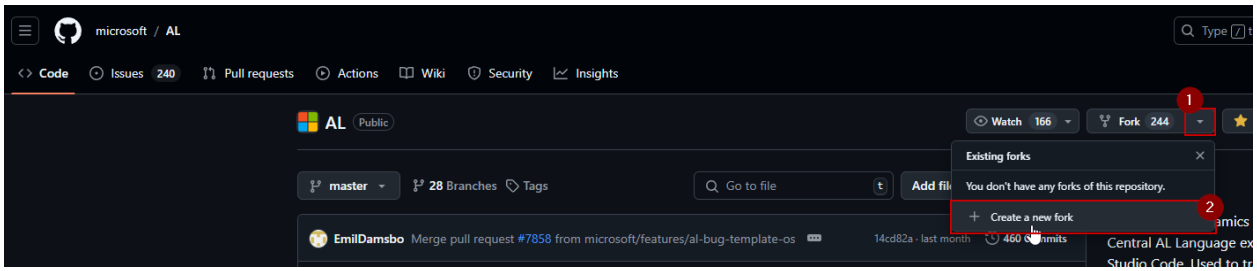
First steps of preparation for starting developer contribution

To set up your development environment effortlessly, follow these steps in the exact order. These steps should guide you through setting up your environment smoothly, ready for development.

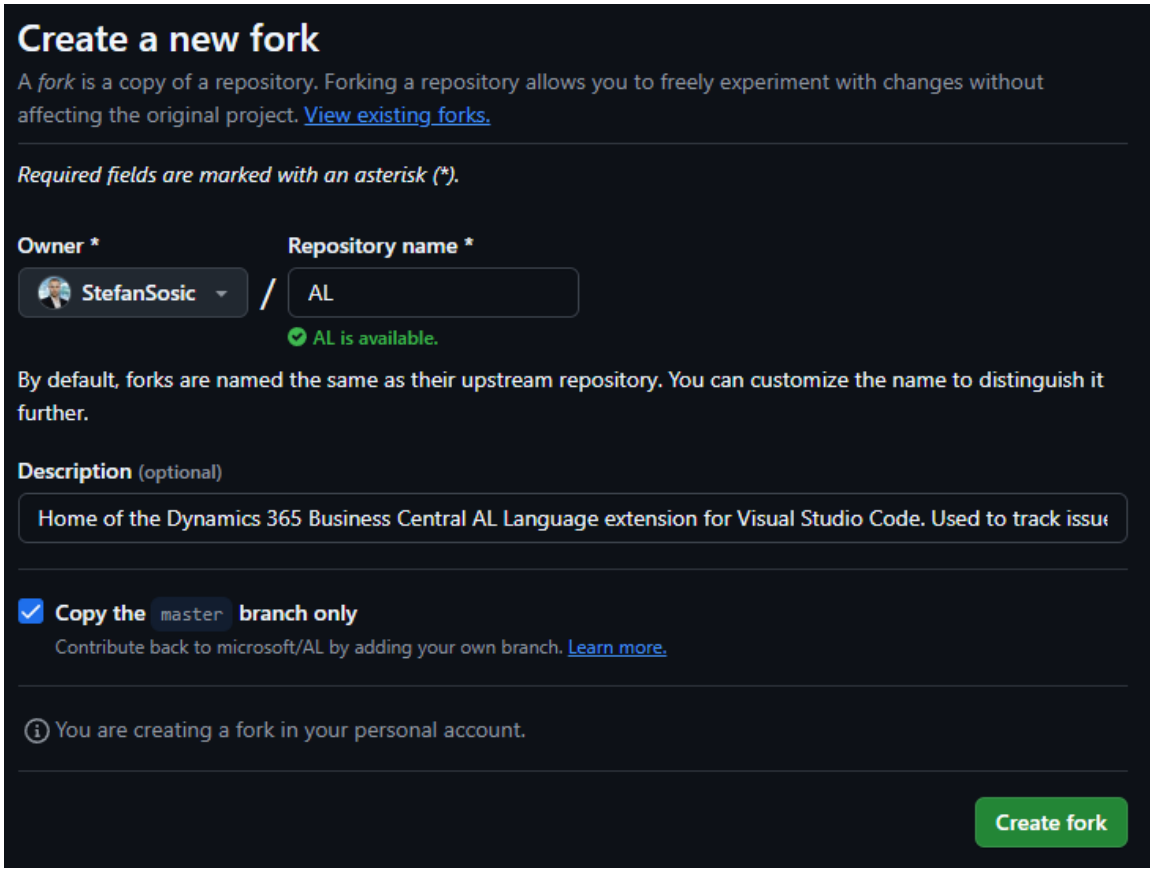
1. Creating a fork on GitHub

Community contributors cannot create new branches directly on the main repository. Each contributor should have their own fork based on the main Microsoft repository.

To create a fork, navigate to the main repository, click on the dropdown for forks, and select "Create a new fork."



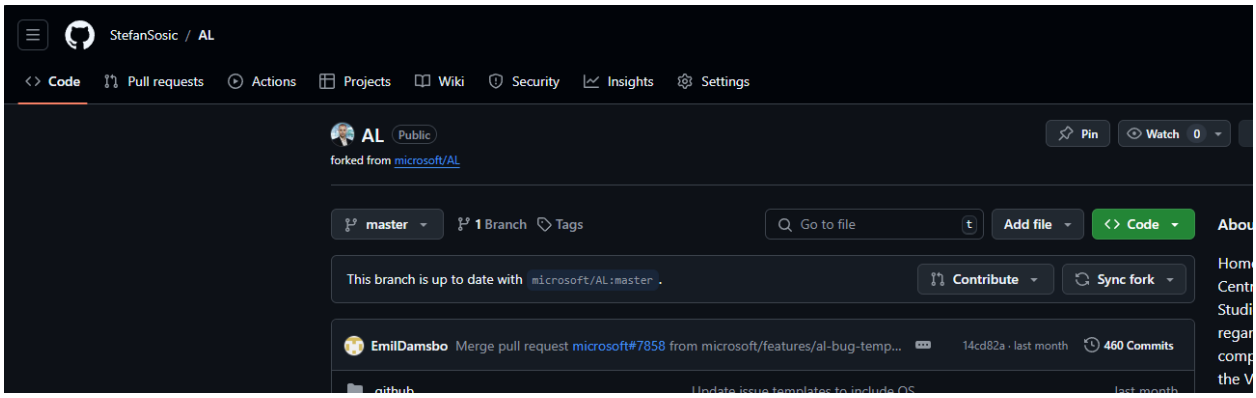
A new page will appear:



I wouldn't change anything and keep everything as prefilled.

Note that you are able to create only **one** fork per the original main repository.

Once you click on the “Create fork” button, you will be redirected to your fork:

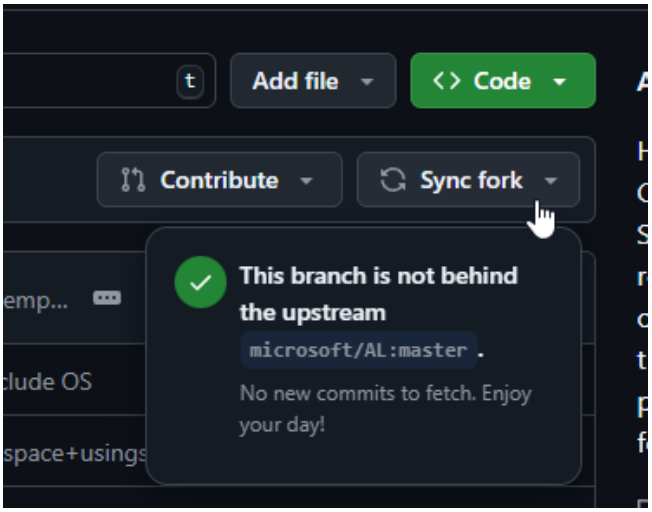


It is basically a copy of the original repository.

1.1 Make sure your fork is up to date

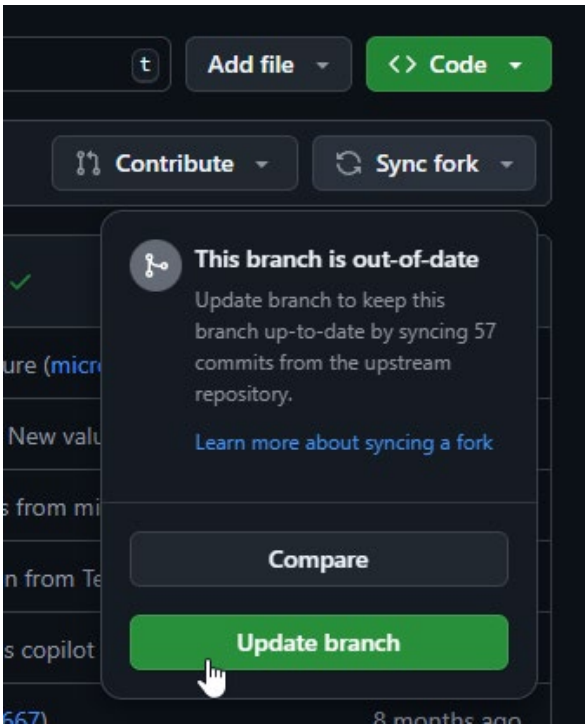
In order to avoid any merge conflict and to work on the most recent code, you will need to keep your fork up to date.

In order to do that, click on the “Sync fork” button:



If it shows like this, that branch is not behind the upstream, you are good to go.

But if not you will see this view:

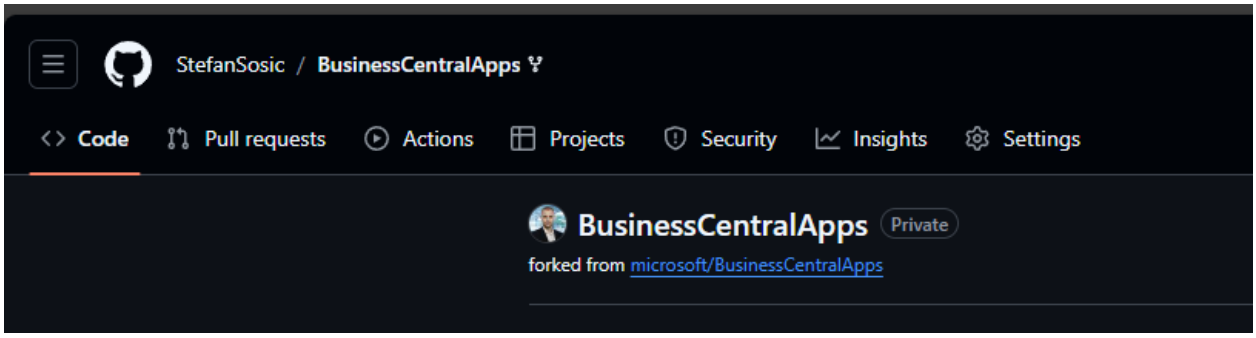


Click on “Update branch” before proceeding any work.

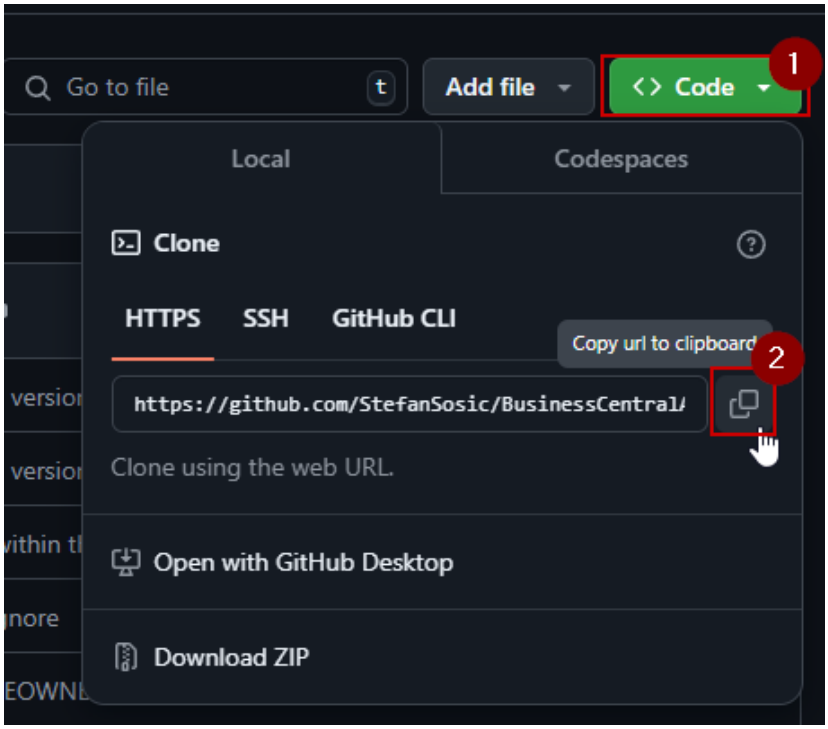
Note: If you already have repository cloned, don’t forget to sync it!

2. Cloning the GitHub repository

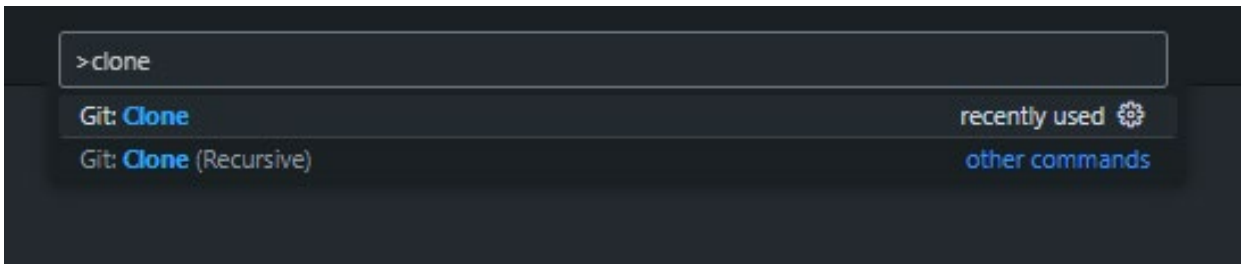
Make sure you are on your own fork:

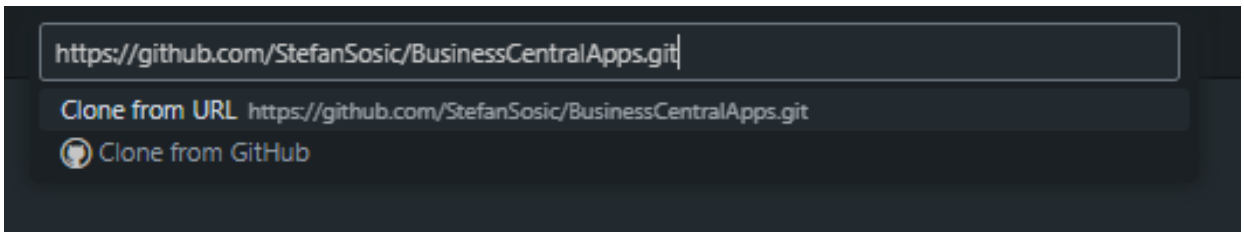


Click on the “Code” button and copy the URL:



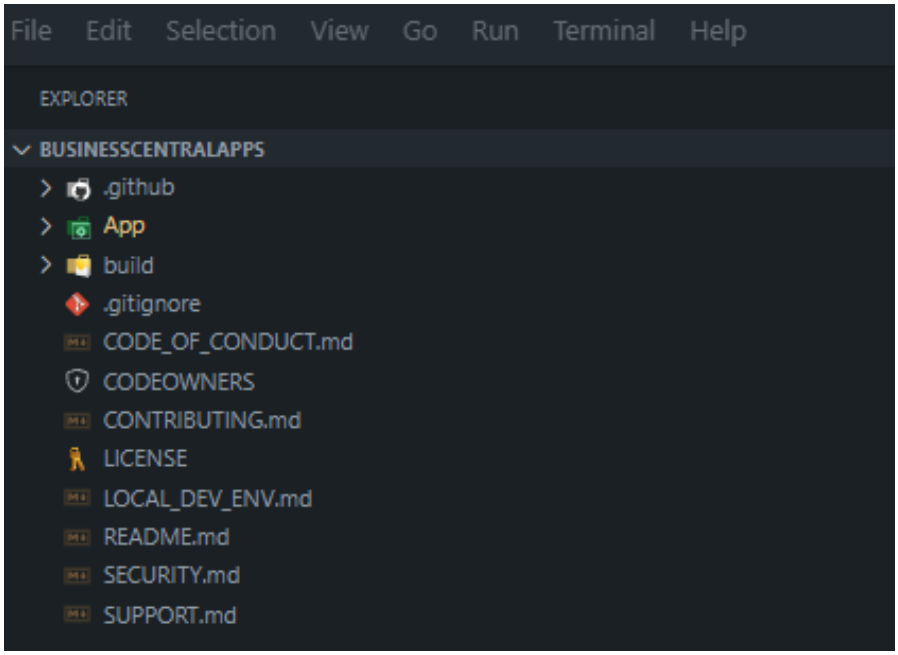
Open Visual Studio Code and open the command prompt. There you type “clone” and press “Enter”:



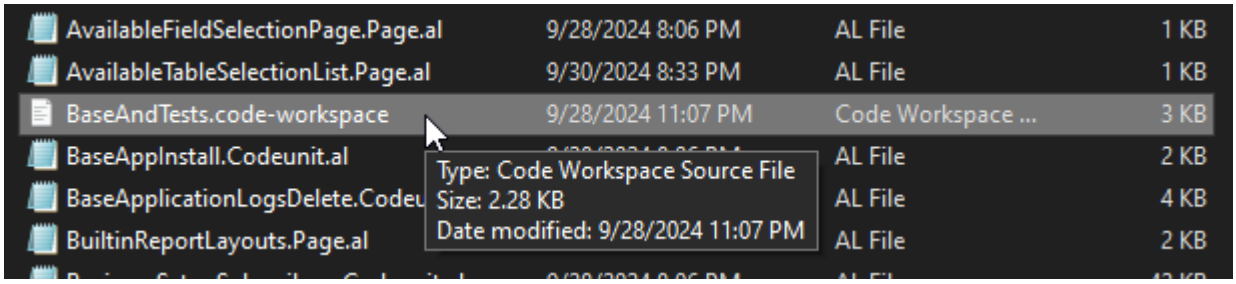


Paste your fork URL copied before and press “Enter”.

If you click open repository after cloning, you will get a view like this:



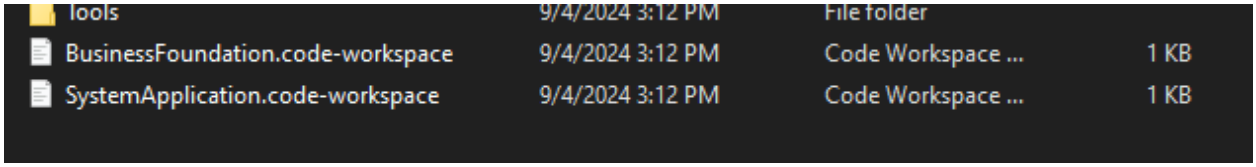
This is not the right way to open your development environment. Each repository has its own Visual Studio workspace file.



For Base Application it's under path:

BusinessCentralApps\App\Layers\W1\BaseApp

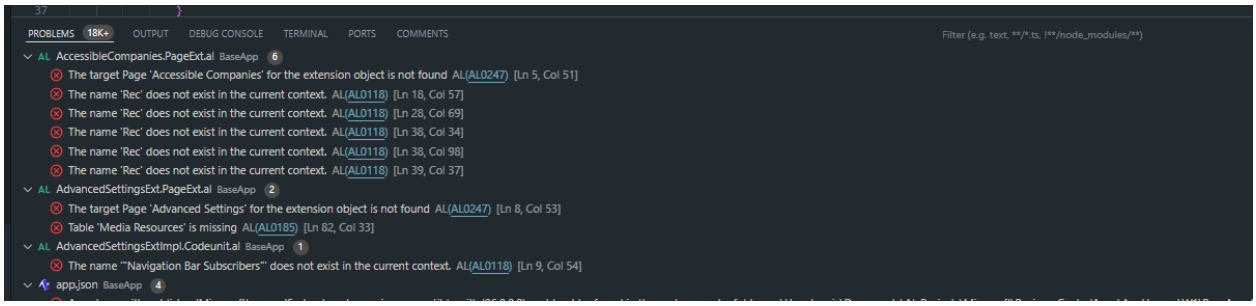
For System Application, BCApps\src:

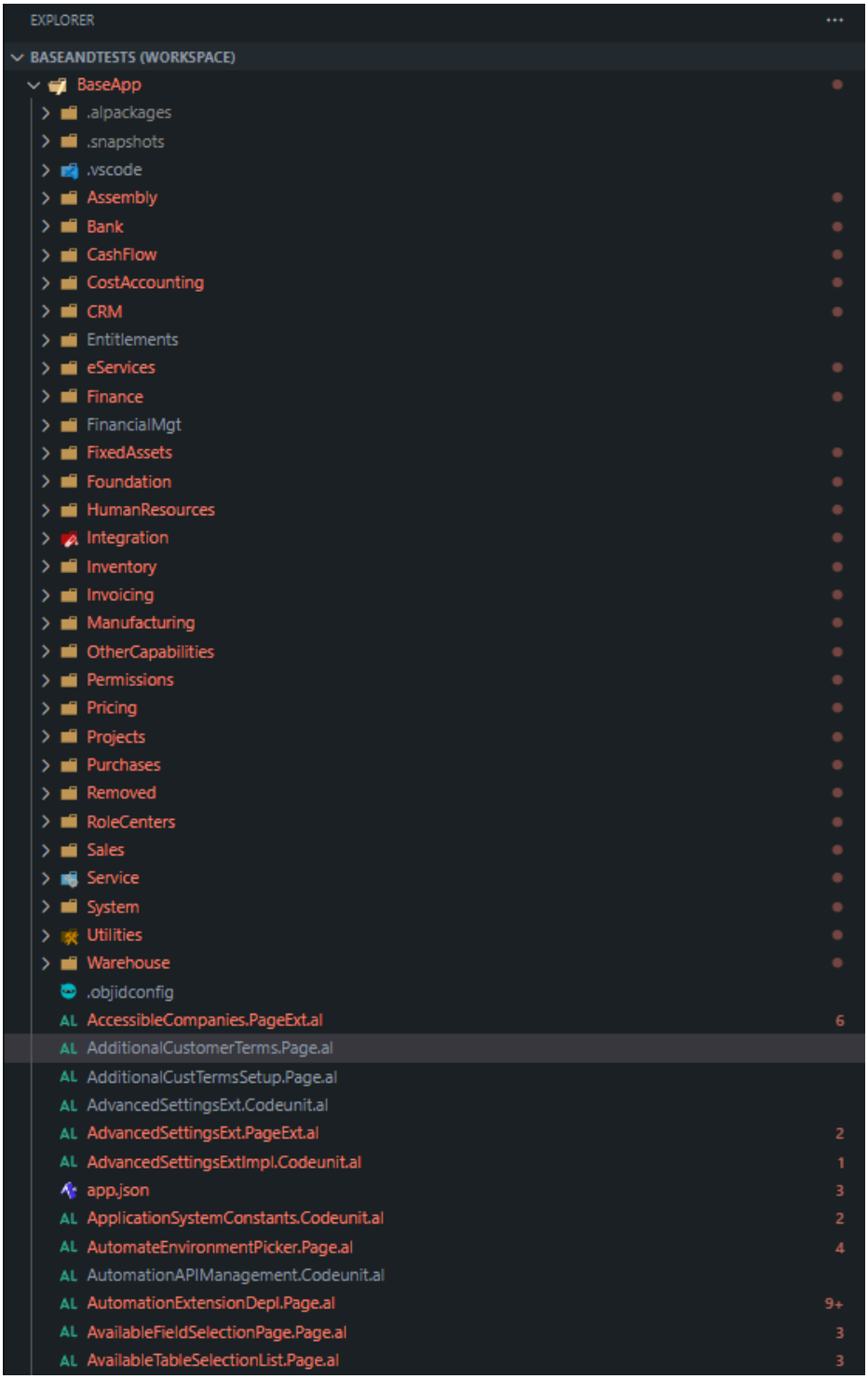


Let's focus on the Base Application.

Open workspace file.

Upon loading the workspace, you will see a large number of errors:





Nothing to worry about for now.

You are missing two crucial things:

1. AL Packages
2. Dot Net files

We will cover that in the next parts of the guide since in order to get those, you will need your local docker environment.

3. Creating a local docker environment

Of course, in order to do this part, you would need Docker installed on your local machine.

Docker Download Link: https://www.docker.com/products/docker-desktop/

It is also possible to use Docker Server without a GUI environment.

3.1. Running script for local docker environment

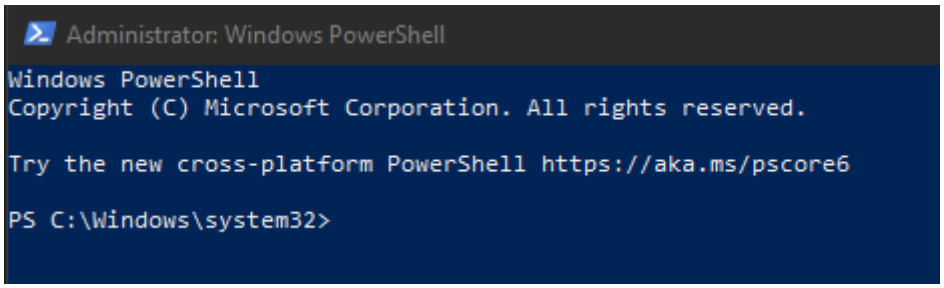
I will focus as before on Base Application. Basically, the process is the same on all repositories. Since our focus is on.AL-Go folder which is basically created out of a template and then adjusted for each repository.

The script is named: localDevEnv.ps1

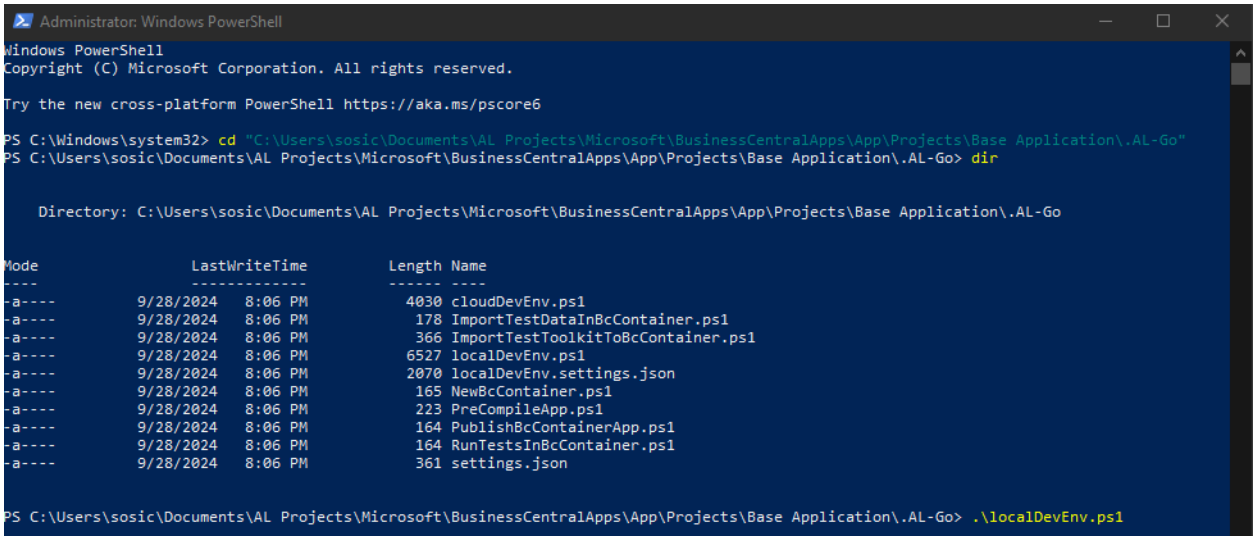
The path for the Business Central repository is:

BusinessCentralApps\App\Projects\Base Application\.AL-Go

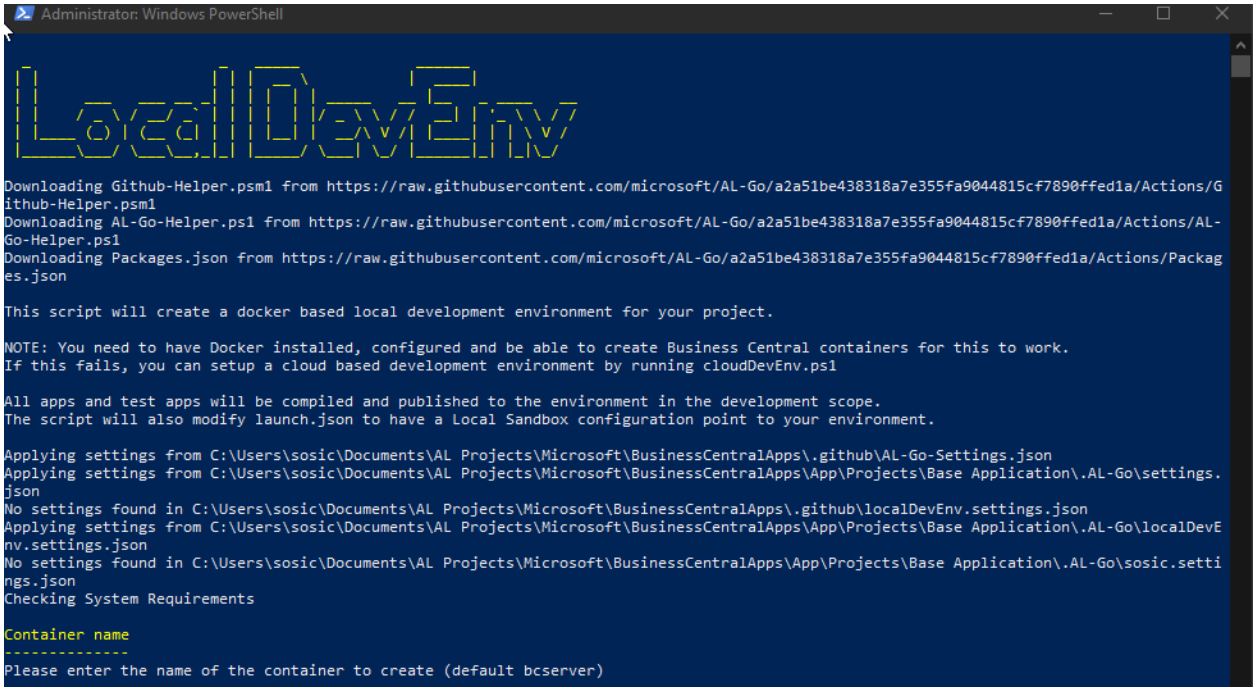
Open PowerShell as Administrator:



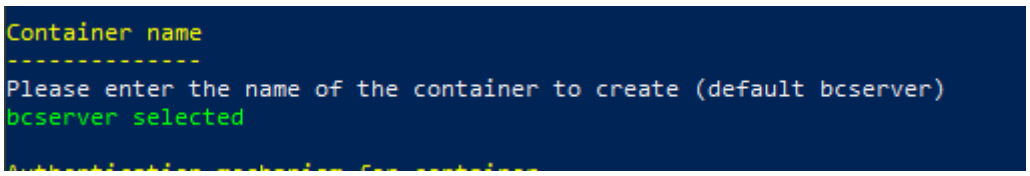
Go to the path and execute the script:



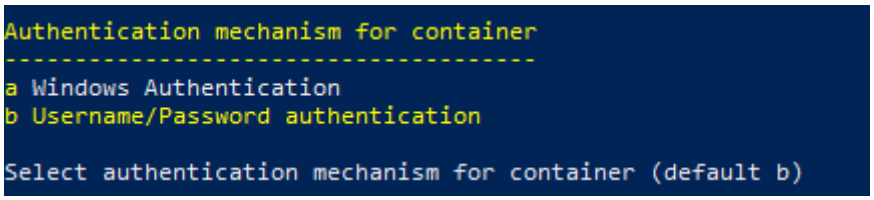
When executed first you will be asked for the Docker container name:



I keep it personally by default as “bcserver”

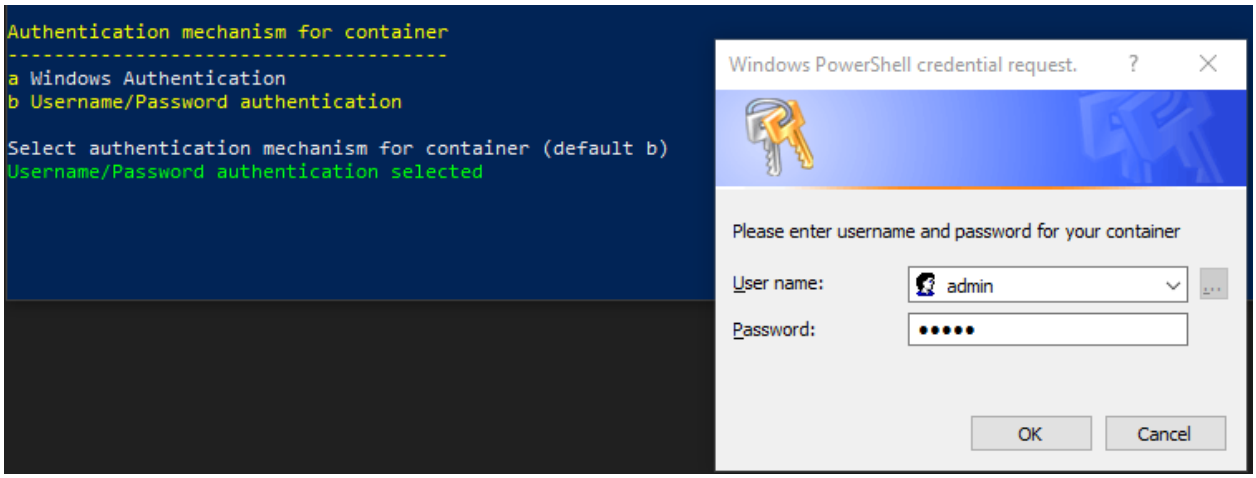


Next Authentication:

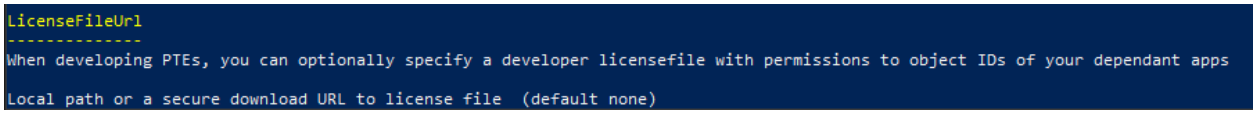


Choose the one that you got used to, we will stick to the default (User/Pass).

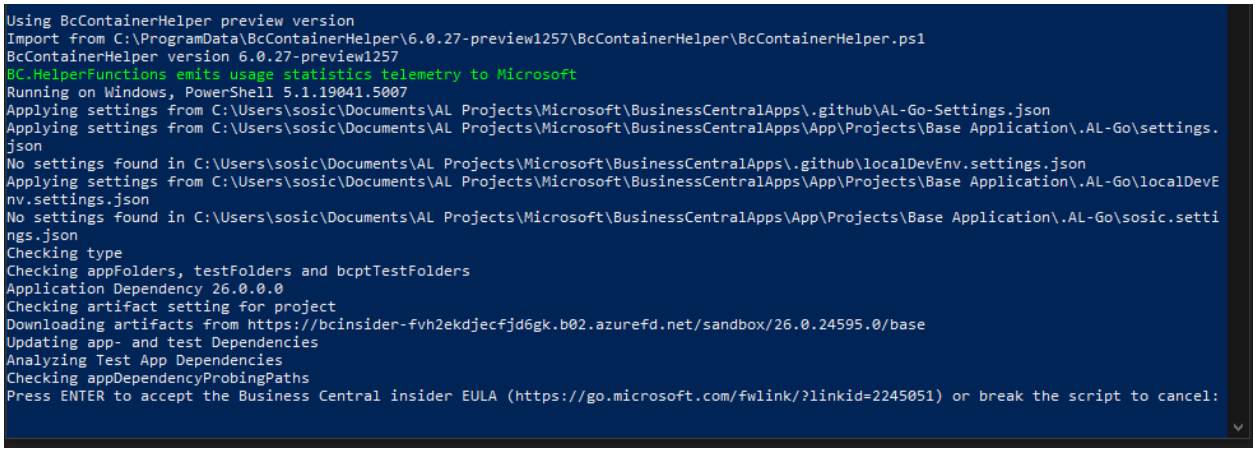
When selected, a credential prompt will show up where you can input the Password of your choice:



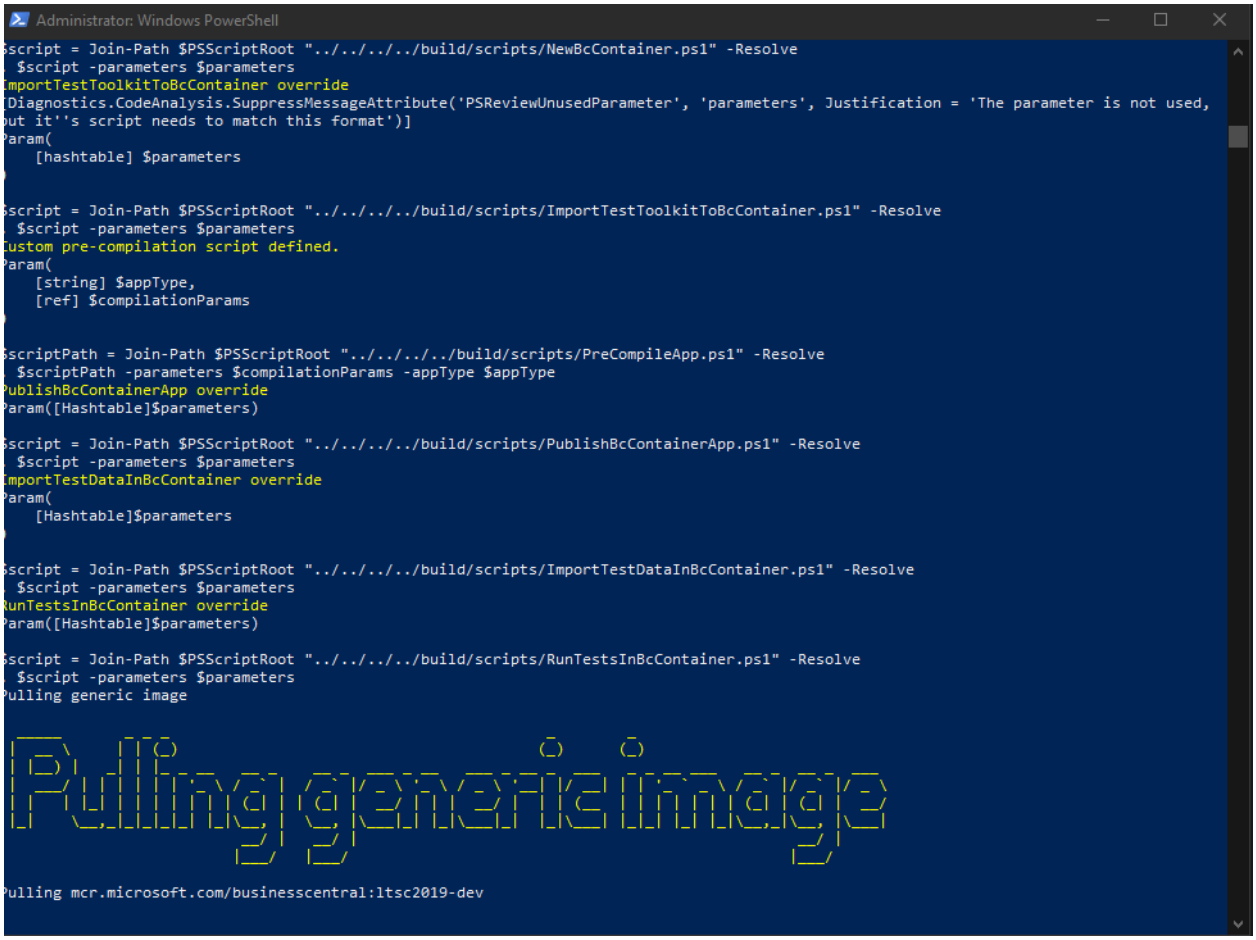
License file if and when developing PTEs, we don't need it for Base Application contributions.



Since we run the file without accept the EULA parameter, now we will get the question to accept EULA, just press enter:



Now, leave as is:

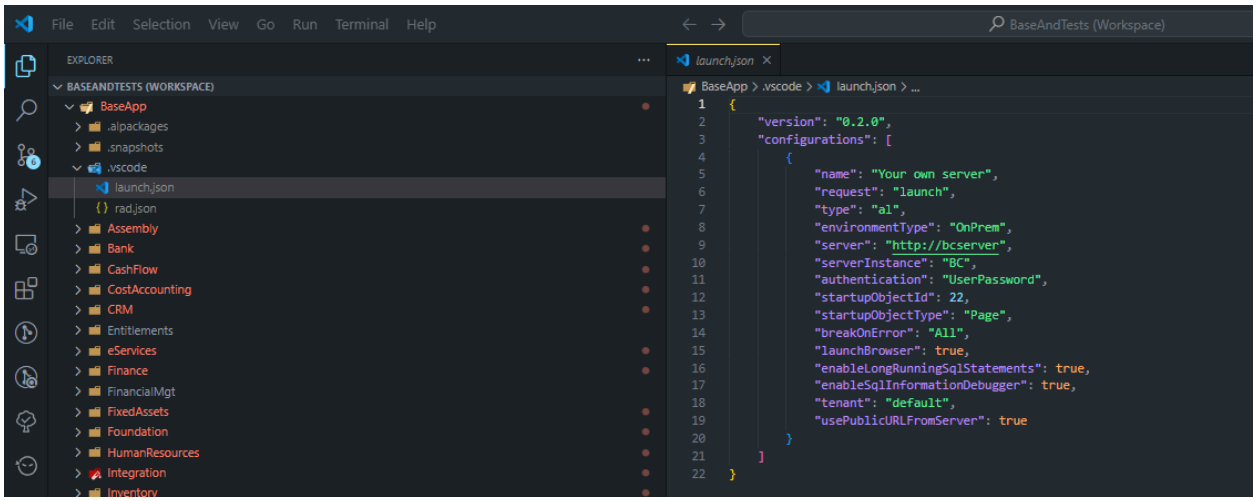


Keep the script running, it needs a lot of time. On my machine, which has quite some performance, it needs approximately one and a half hours.

3.2. Downloading AL Packages

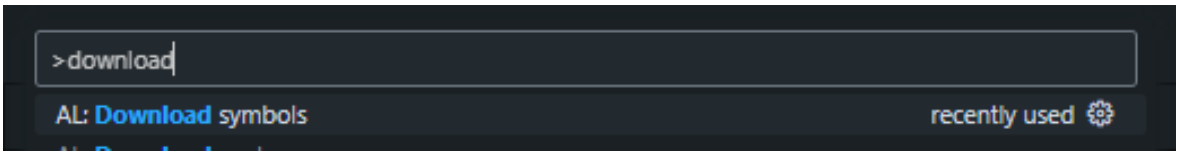
Now, once we have a local docker environment we are ready to download the symbols needed for Base Application.

Open Visual Studio Code workspace:



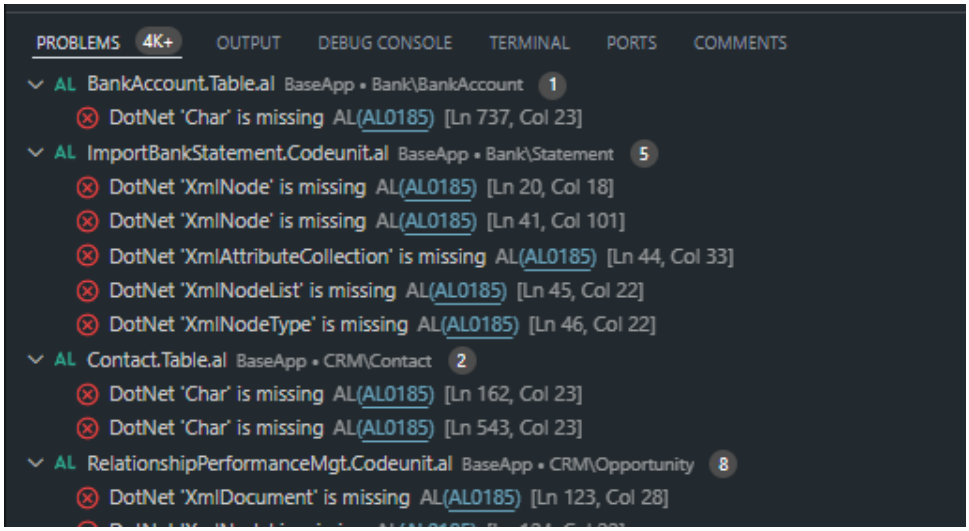
The script should have already prepared launch.json for your local environment. If not, as you can see, it's a simple one.

Execute the "Download symbols" command:



Then input your username and password.

You should get the message "Symbols have been downloaded."



You have a lot of problems resolved, but what about missing DotNet's. Follow the next step to resolve those.

3.3. Defining assembly probing path for DotNet's

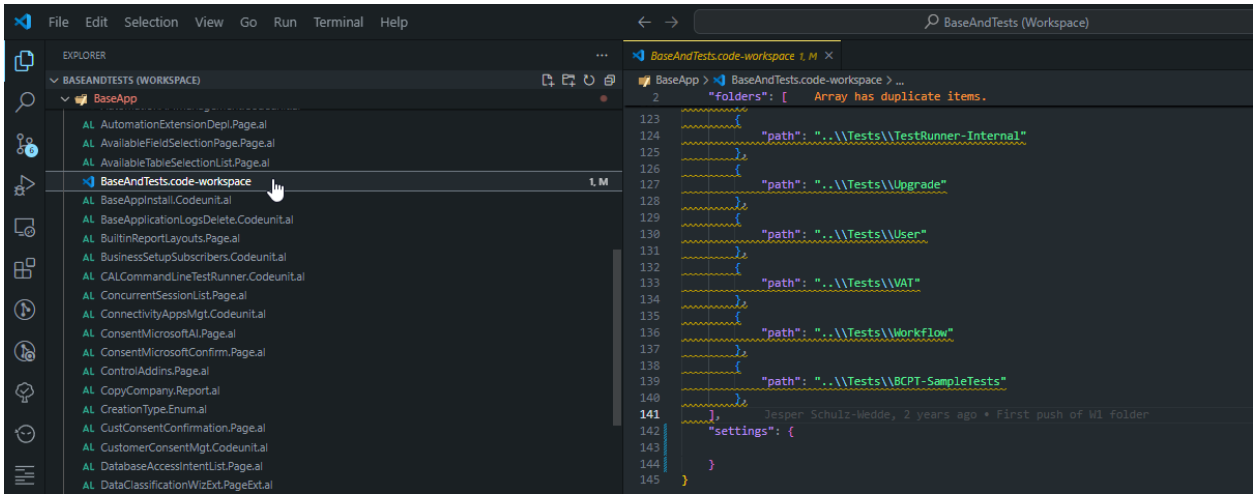
In order to resolve issues related to missing DotNet files you will need to define the assembly probing path.

Where are the DotNet's?

C:\ProgramData\BcContainerHelper\Extensions\bcserver\.netPackages

Where do I put that path?

Open your workspace file in the VS Code editor:

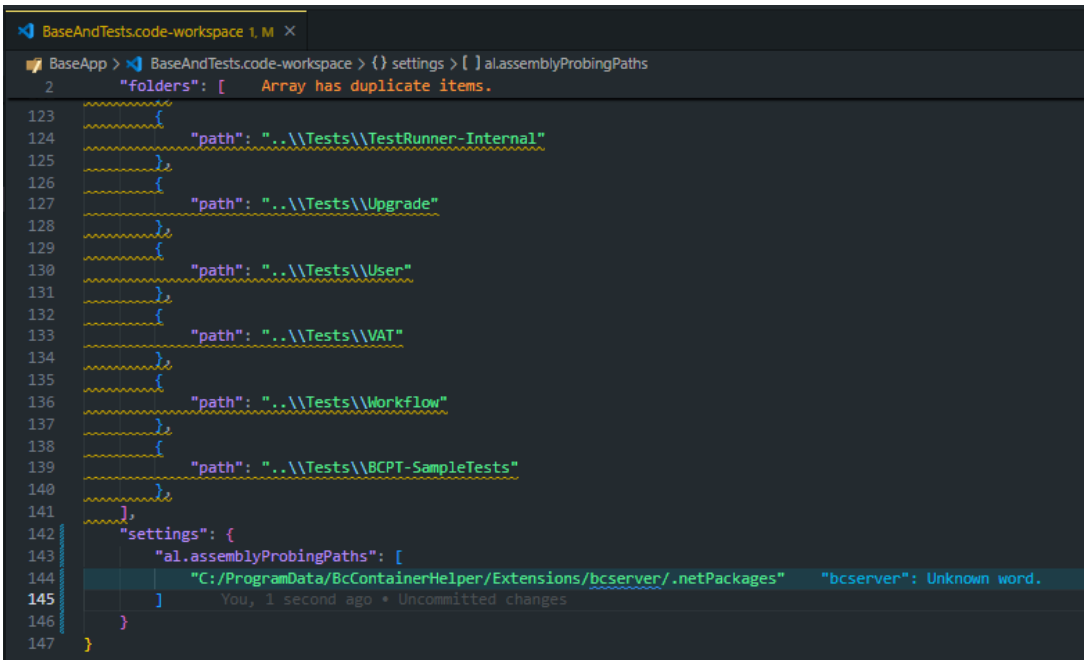


Just click on it and it will open up.

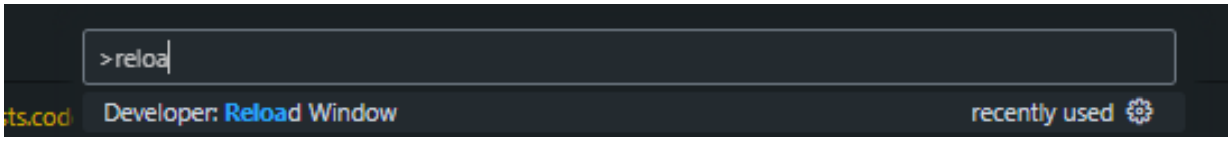
Inside “settings” part, you will need to define the following:

```
"al.assemblyProbingPaths": [
        "C:/ProgramData/BcContainerHelper/Extensions/bcserver/.netPackages"
]
```

It will look like:

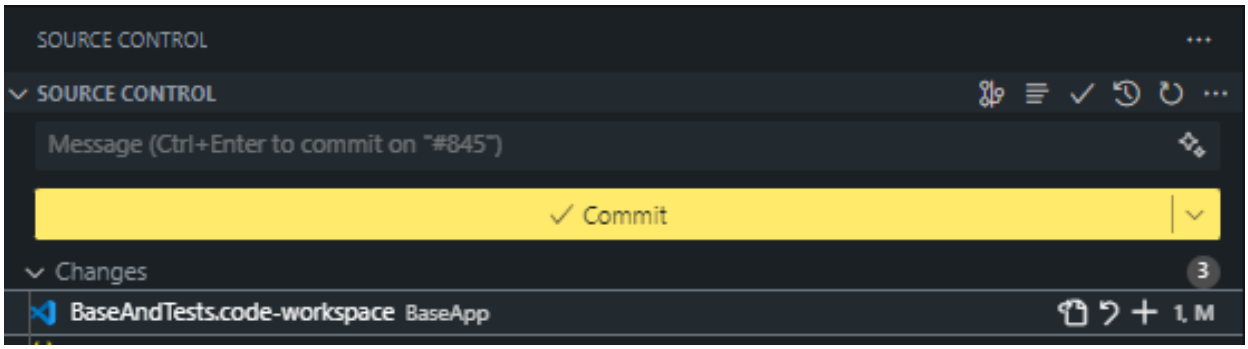


Wait a bit and errors will be gone. If not, restart your VS Code:



You could do that with the “Reload Window” command.

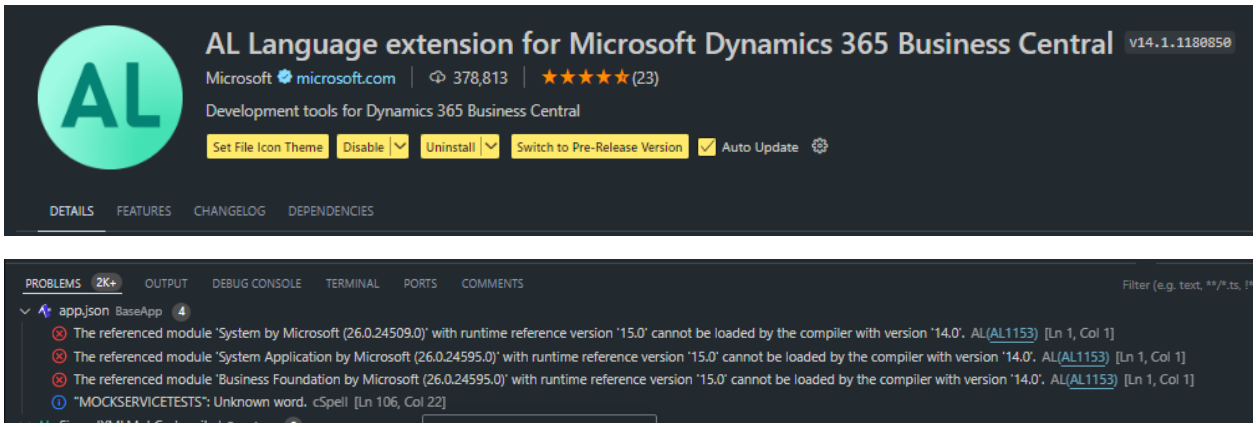
When you did this, this file will show up in Source Control:



Make sure you don't commit this file!

3.4. Use the correct AL Language extension for VS Code

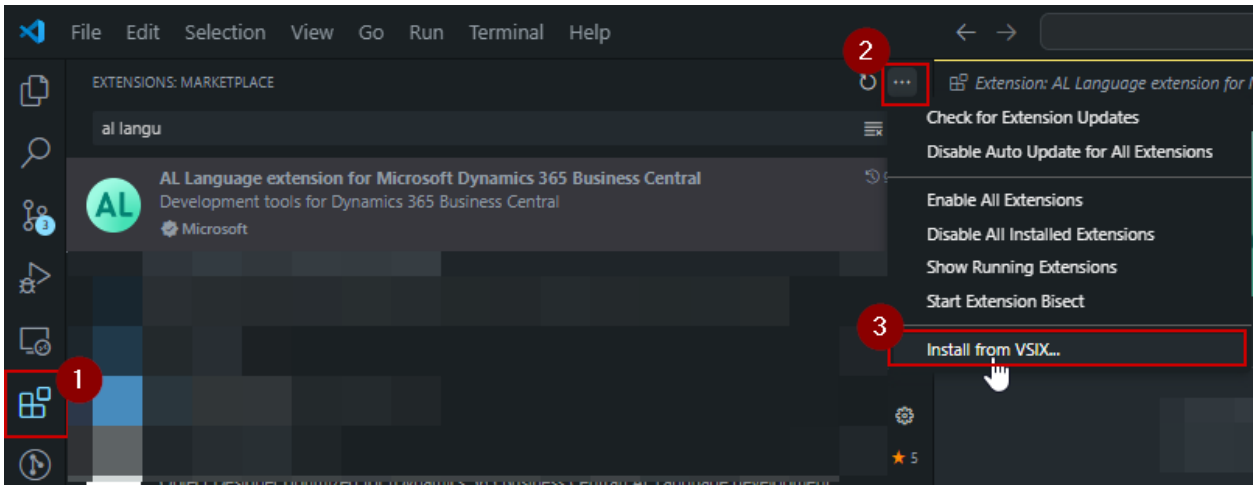
When working with the insider version you can't use the Release version of AL Language:

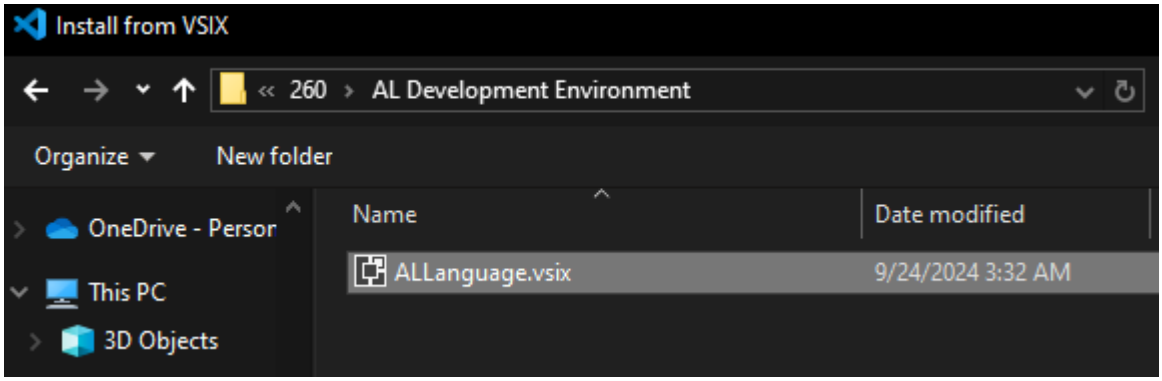


And if not just runtime version error, you will get more errors.

Even using a pre-release version can cause errors. The safest method is to use the VSIX file which comes with your local environment.

Go to the extensions tab in your Visual Studio Code:





Where is my VSIX file?

Example path is:

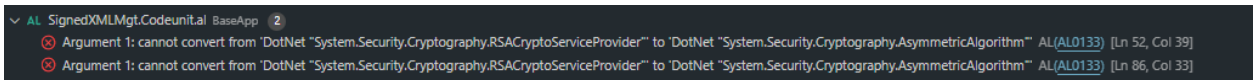
C:\bcartifacts.cache\sandbox\26.0.24595.0\platform\ModernDev\program files\Microsoft Dynamics NAV\260\AL Development Environment

Why it's said example path?

Because the Business Central version is increasing it will be your sandbox version, in this case 26.0.24595.0 as long as the "260" version with the new major release of Business Central.

3.5. Other possible errors

At the moment of creating this guide there is one error that is currently present on the repository in Base Application:



It's about conversion in "SignedXMLMgt" codeunit file.

Feel free to comment it out. But keep in mind when pushing changes to your branch to exclude that file.

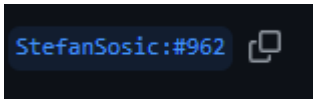
Branch policy on your fork

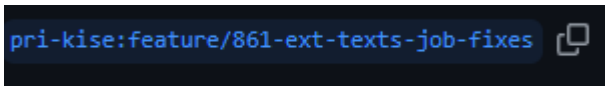
Keep in mind that you will be missing a branch policy where your "main" branch is protected and requires a Pull Request to add a new code to it.

Once you would like to start working on some issue, sync your "main" branch and than create a new one, so your "main" one gets always clean.

Branch naming

There isn't a strict rule about naming your branches, since they are also on your forked repository, but however, it will be much easier for you and for everyone if the branch name is the ID of the issue, for example:





Testing your code

Tests are basically always needed!

If you're fixing a bug, you need to check why it wasn't already covered by tests and whether it can be covered now. This way, if someone tries to break it in the future, tests will detect it.

When developing a new feature, tests are essential. The question isn't whether you need tests, but if you've written enough to cover the entire feature.

Minor improvements in code that already has tests might not need additional tests if they are of minor importance.

When making tests make sure you follow guidelines.

```
1082 +     [Test]
1083 +     [HandlerFunctions('MessageHandlerCombineShipment')]
1084 +     [Scope('OnPrem')]
1085 +     procedure CombineSingleServiceShipmentWithCustomerCombineShipment()
1086 +     var
1087 +         Customer: Record Customer;
1088 +         ServiceHeader: Record "Service Header";
1089 +         Quantities: List of [Decimal];
1090 +         ItemNo: Code[20];
1091 +     begin
1092 +         // [FEATURE] [Combine Shipments] [Description Service Line]
1093 +         // [SCENARIO 363166] Creating single Service Shipments with Customer which has set
      Combine Shipments to "true"
1094 +         Initialize();
1095 +
1096 +         // [GIVEN] Create new Customer with Combine Shipments = true
1097 +         CreateCustomerWithCombineShipments(Customer);
1098 +
1099 +         // [GIVEN] Create new Item
1100 +         ItemNo := LibraryInventory.CreateItemNo();
1101 +
1102 +         // [GIVEN] Create single Service Order
1103 +         CreateServiceOrderForCustomerAndDoShipment(ServiceHeader, Customer, Quantities,
      ItemNo);
1104 +         LibraryService.PostServiceOrder(ServiceHeader, true, false, false);
1105 +
1106 +         // [WHEN] Combine Shipment with "Post Invoices" and "Copy Text Lines" = true
1107 +         LibraryVariableStorage.Enqueue(CombineShipmentMsg);  // Enqueue for MessageHandler.
1108 +         RunCombineShipments(Customer."No.", false, true, false, true);
1109 +
1110 +         // [THEN] Verify Posted Service Invoice contains single lines with correct data
1111 +         VerifyPostedServiceInvoice(Quantities, ItemNo);
1112 +     end;
```

Your tests should contain:

- Feature name of tests
- Scenario description which you are testing
- Test structure GIVEN -> WHEN -> THEN as well as description among those tags

New AL Files

When you are creating fully new files you need to pay attention to a few things.

1. Object range

Let's start with the object range. Object ranges are defined per module, you can try to find the next ID but always double-check with Microsoft.

2. Object header with copyright information

Next, when you are creating a new object in any repository, you would need to put at the beginning of the object copyright information:

```
1   // ------------------------------------------------------------------------------------------------
2   // Copyright (c) Microsoft Corporation. All rights reserved.
3   // Licensed under the MIT License. See License.txt in the project root for license information.
4   // ------------------------------------------------------------------------------------------------
5
6   namespace System.Integration;
7
```

// --

// Copyright (c) Microsoft Corporation. All rights reserved.

// Licensed under the MIT License. See License.txt in the project root for license information.

// --

3. Creating a new module

Ask for the object range which you should use in your new module.

In app.json, internals should only be visible to test libraries if at all.

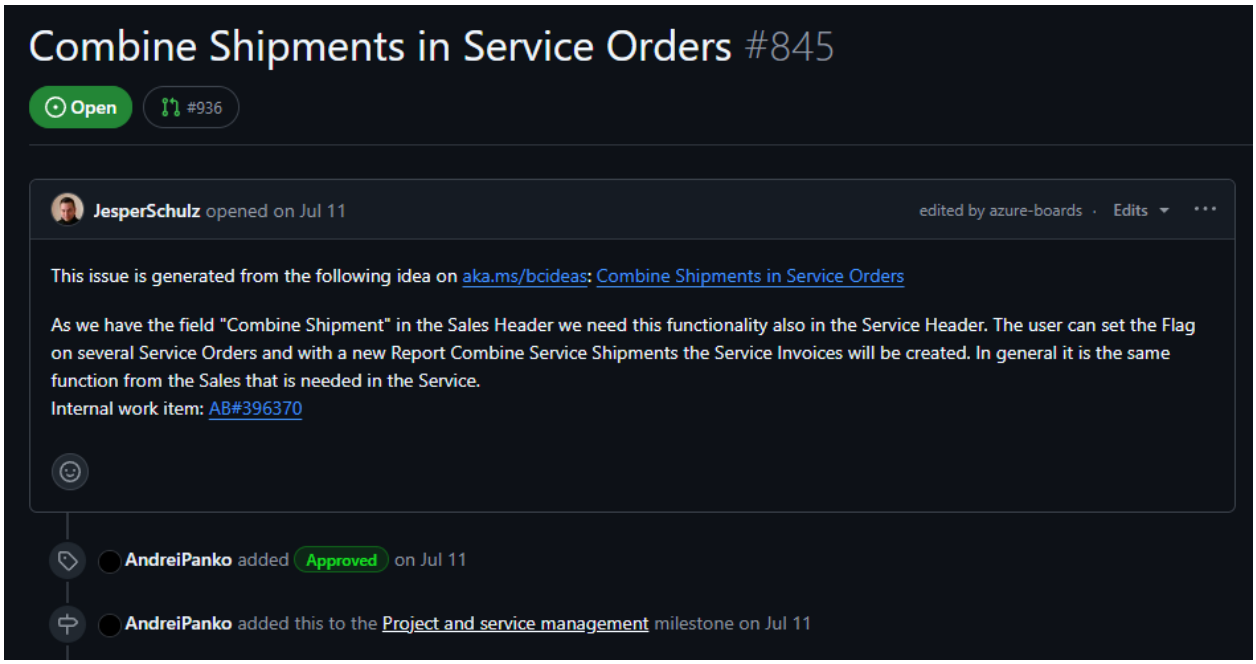
Describe a new module inside the README.md file.

And of course, create tests inside your new module.

Creating your first Pull Request

Take the issue on GitHub

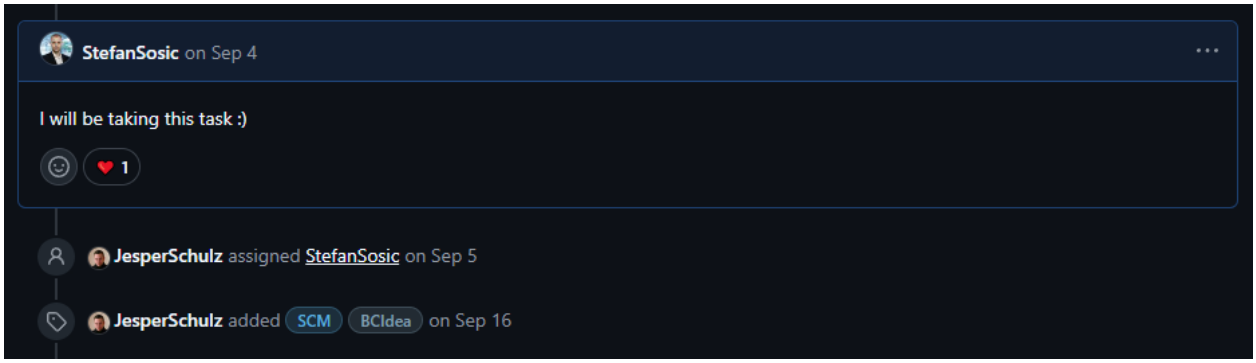
Find the issue which you find interesting and would like to provide a solution for it.



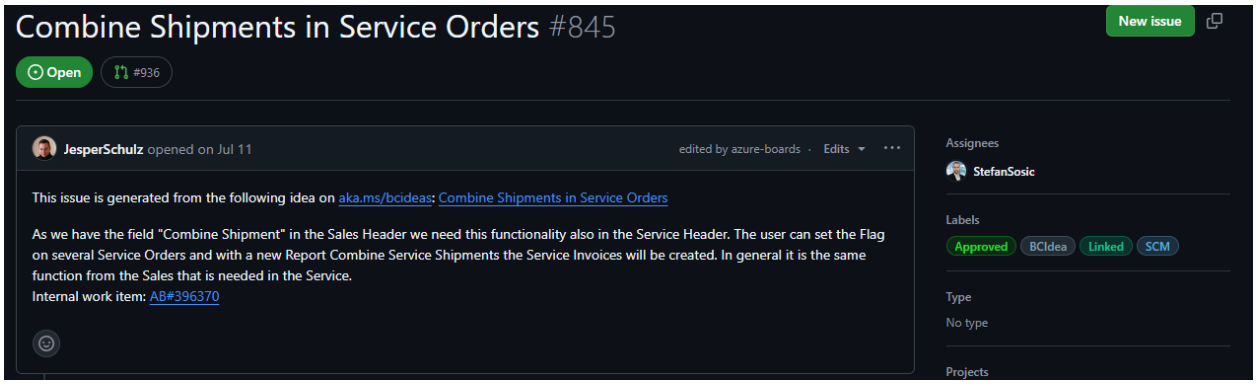
Make sure it has an “Approved” tag on it.

What is the benefit?

You will be handling this issue and other community members will know that somebody has already taken that issue into development and will be providing a solution soon.



So now our issue looks like this:



It has an “Approved” tag, and assignee to it. You can move forward now into the development phase.

Create a branch

Go to your cloned fork and create a branch based on the “main” branch. Make sure you have synced your fork and main branch.

Pre-Pull Request

You finished your development. Now what?

Make sure you do the following:

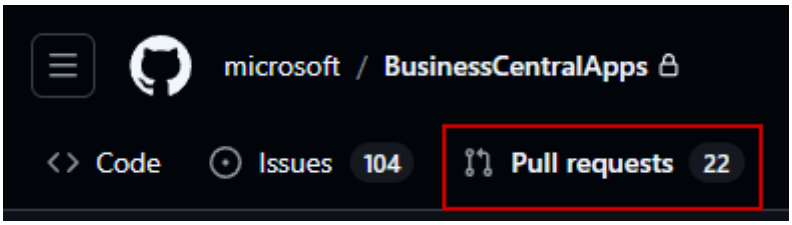
- Variables are sorted
- Usings are sorted
- You don’t leave unused variables
- Check if you are making breaking changes (if you are doing breaking changes, make sure you put obsolete property and CLEAN tags)
- Check spelling in your Labels/Texts/Procedures/Descriptions
- Casing of methods, variables, procedures
- Missing parenthesis
- Empty lines if mistakenly added
- Correct spacing/formatting

These are just the most commonly seen mistakes, which get most of the comments on Pull Request. But if you take 5 minutes before creating a Pull Request, or right after you create it, you will prevent a lot of unnecessary comments. Generally, pay attention to AL development guidelines.

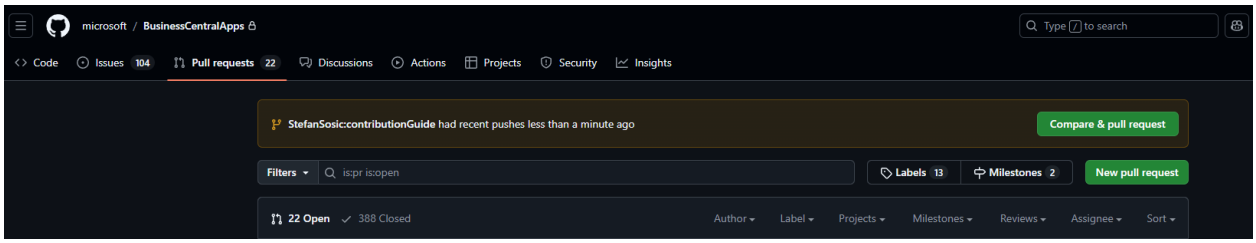
Create Pull Request

You committed all to your branch and you did Git Push or Sync.

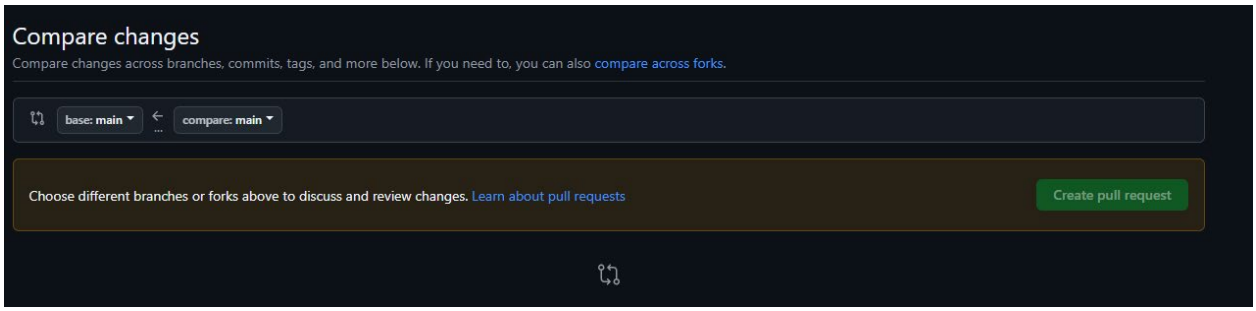
Open the GitHub Microsoft repository and go to the “Pull Requests” tab:



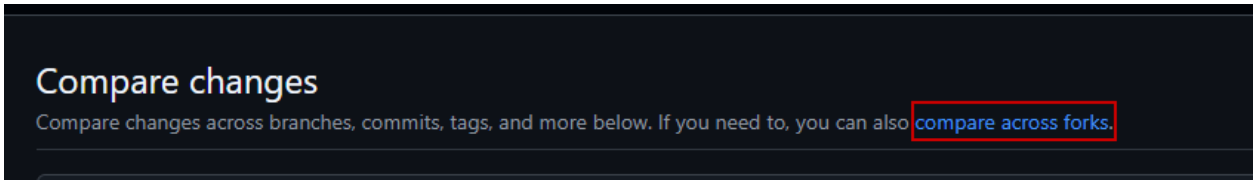
If you have done all right you will see a screen like this:



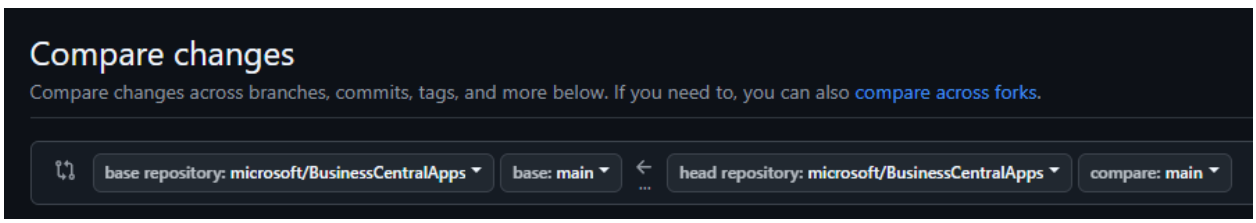
If you want to create a Pull Request for some issue that you resolved earlier but haven’t created a Pull Request yet, you can go to “New pull request”:



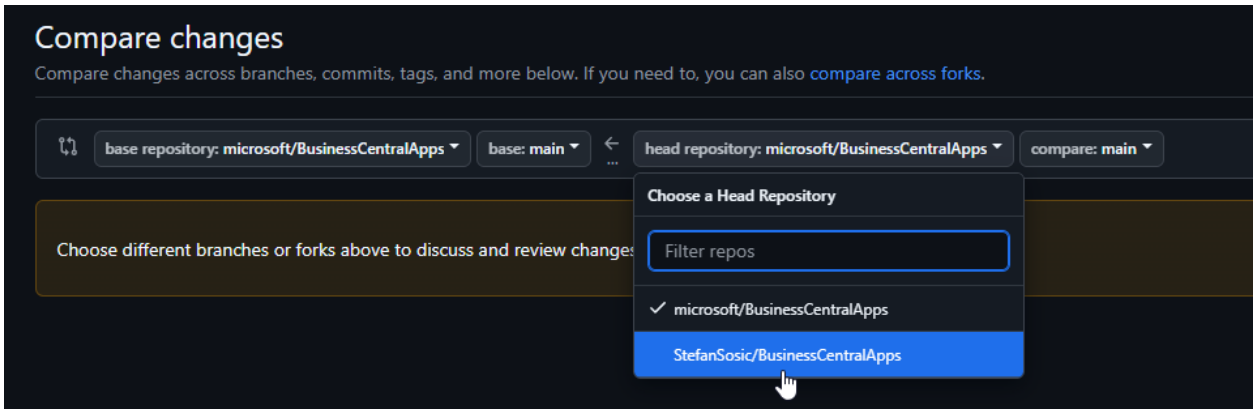
If you don’t see your branch, no worries. You will need to select “compare across forks”.



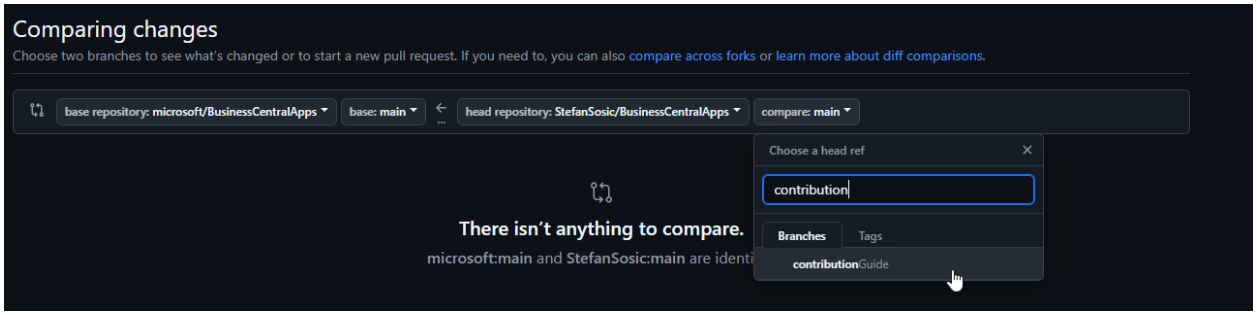
Different compare appear:



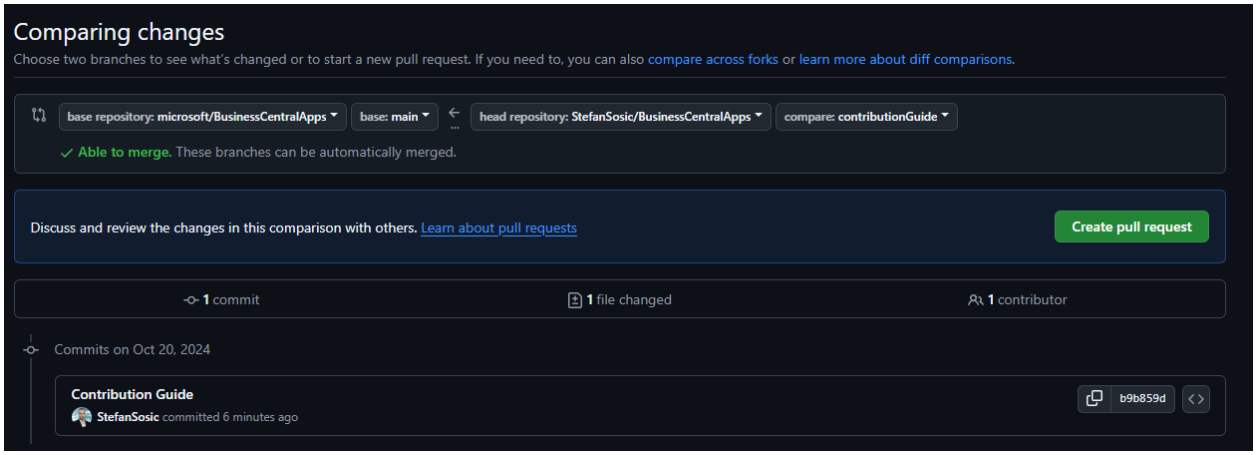
On the head repository drill down and select your fork:



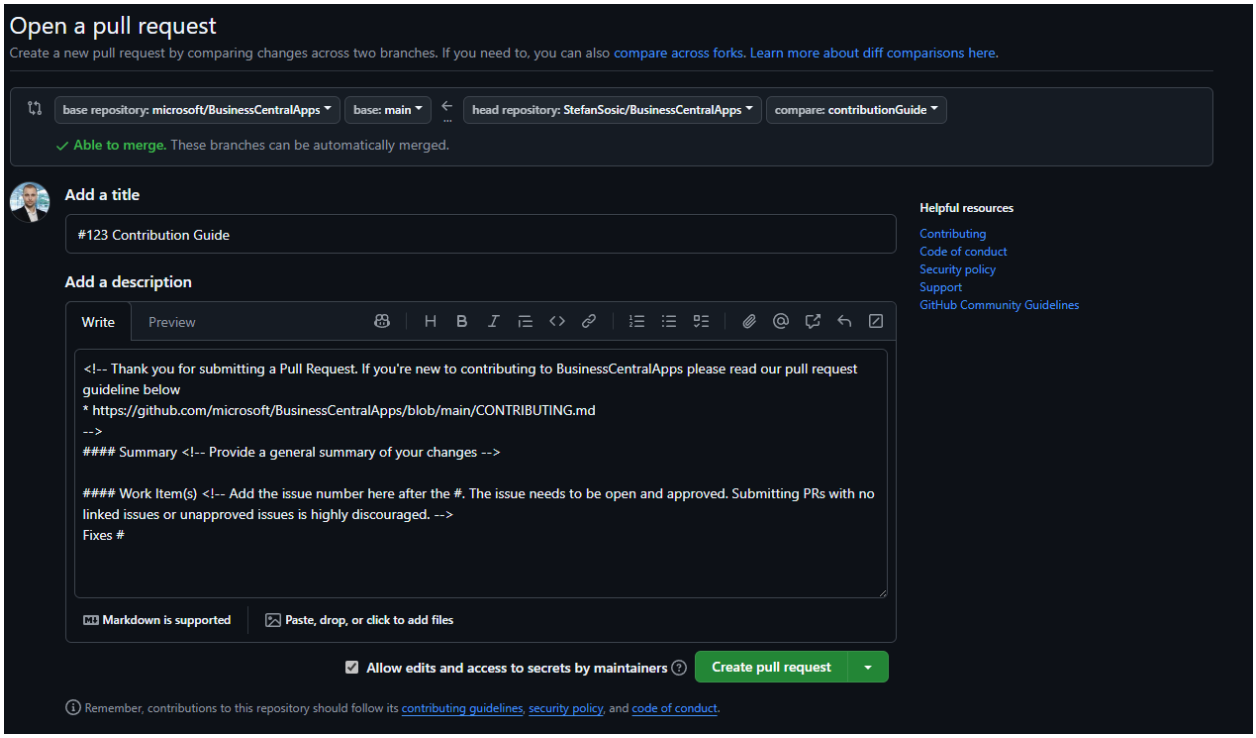
Once selected, move to drill down of compare branches:



Upon selecting, a list of your commits will appear, along with a compare view of the files you changed (right time to double check everything, before pressing "Create pull request")



Once you validated everything, you can press “Create pull request”.

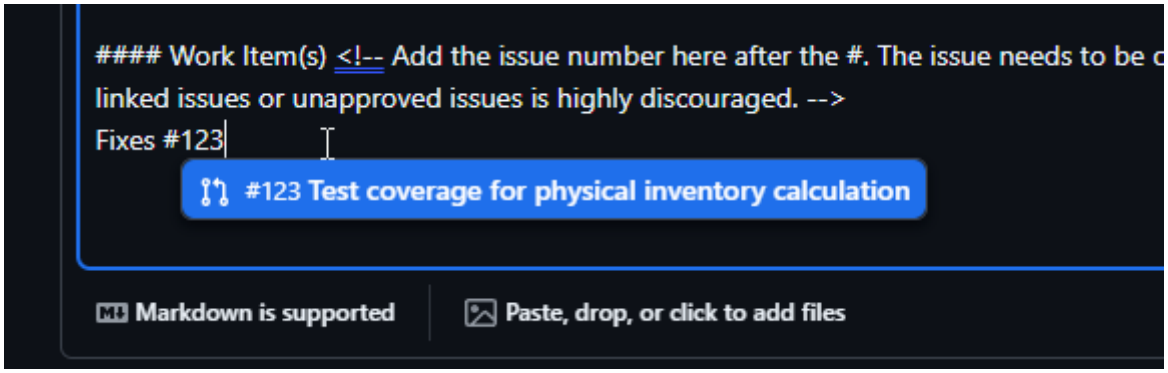


Here you would need to fill out Title and Description.

In the Title would be nice to include your issue ID.

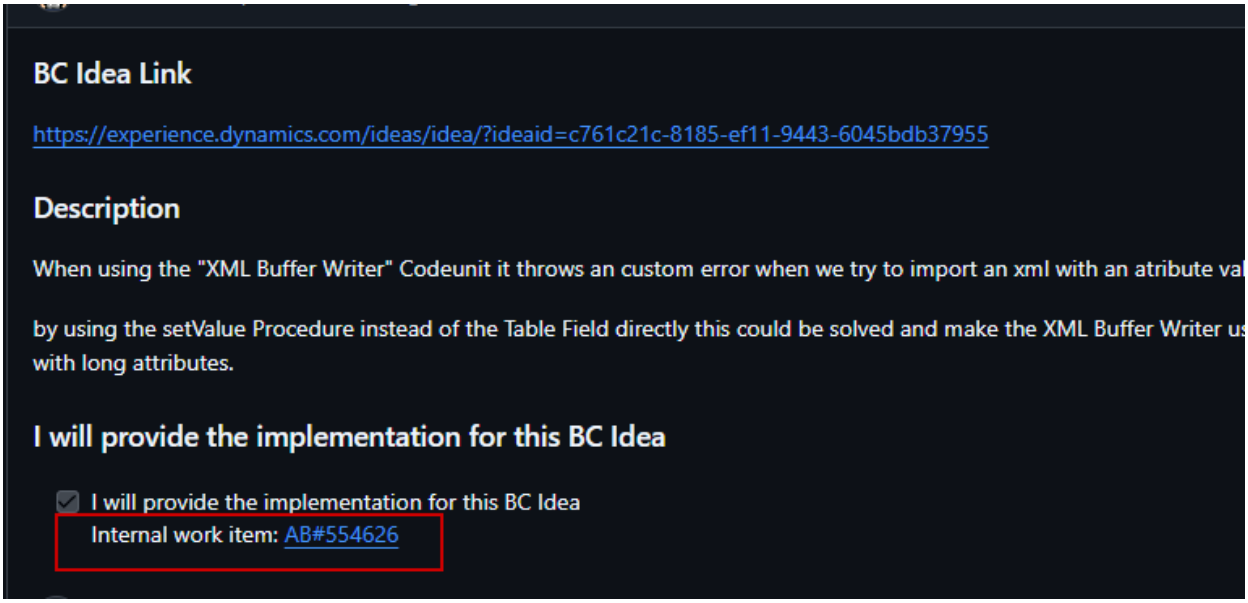
In the description, if you have all described the issue, copy a short description and paste it on the Pull Request description.

In the Work Item’s part, you need to put the issue ID:

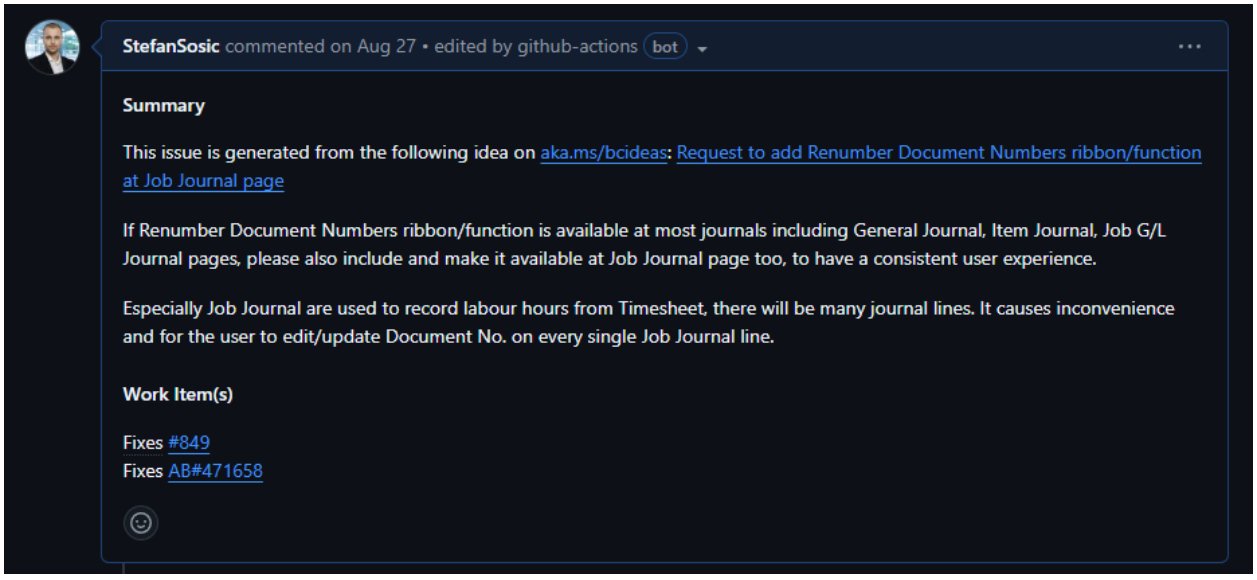


Fill in and press “Enter”.

If you didn’t start the issue immediately upon approval, on the issue you will most probably also have Microsoft Internal work item number:



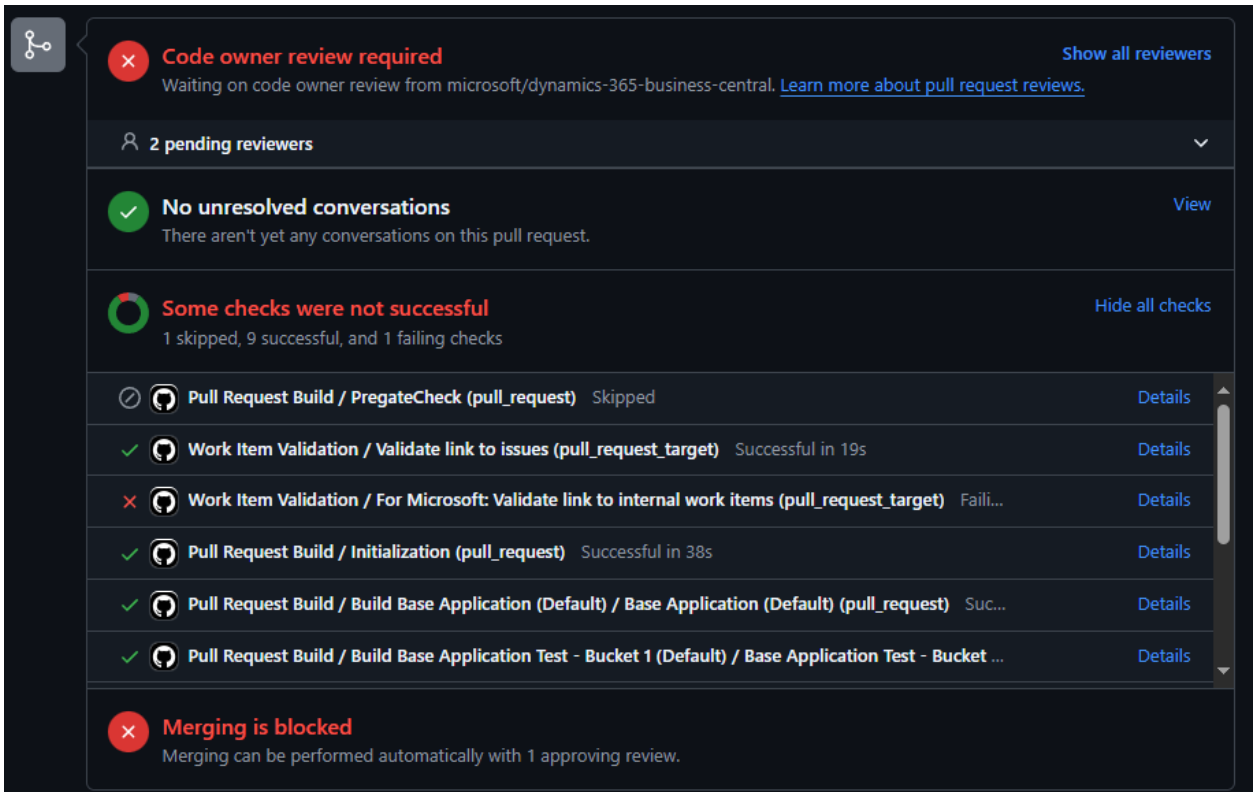
If existing, make sure you put that one also below your issue ID in the Pull Request description since it's required.



It should look like this.

If you don't have that number yet, no worries, it will be added by Microsoft afterward.

Meanwhile, the pipeline validation will fail on that check:



But no reason to worry about it.

Finally, you are ready to click on "Create pull request".

What now?

Wait for comments from other community members or Microsoft.

Check if CI fails (more in Pull Request CI's part of the guide)

Contribution License Agreement

When you are doing your first pull request, you will be asked to accept CLA (Contribution License Agreement).

@microsoft-github-policy-service agree [company="{your company}"]

Options:

1. (default - no company specified) I have sole ownership of intellectual property rights to my Submissions and I am not making Submissions in the course of work for my employer.

@microsoft-github-policy-service agree

2. (when company given) I am making Submissions in the course of work for my employer (or my employer has intellectual property rights in my Submissions by contract or applicable law). I have permission from my employer to make Submissions and enter into this Agreement on behalf of my employer. By signing below, the defined term “You” includes me and my employer.

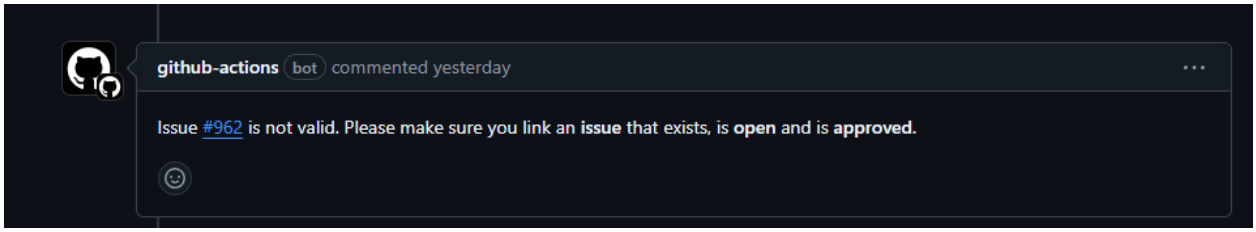
@microsoft-github-policy-service agree company="Microsoft"

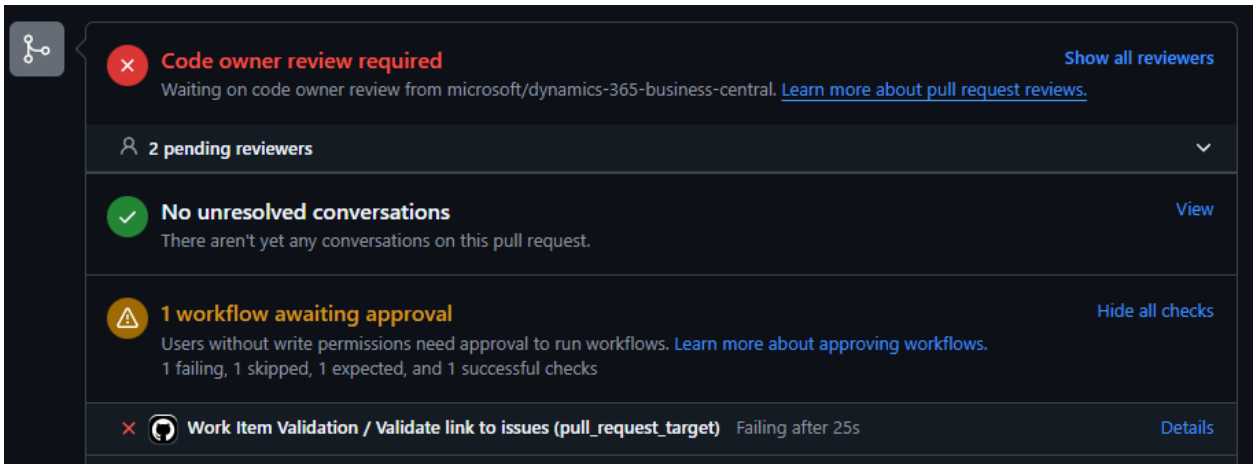
You will get an email regards this, that your Pull Request validation build failed because of this. You can either answer on email, or add as a comment on GitHub.

Pull Request CIs

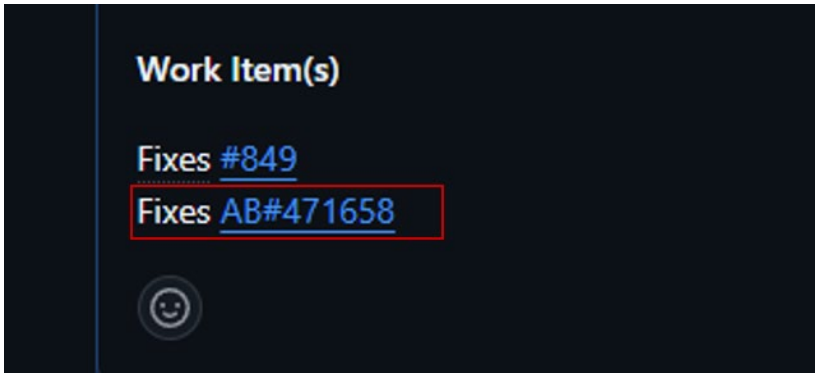
Issue not approved

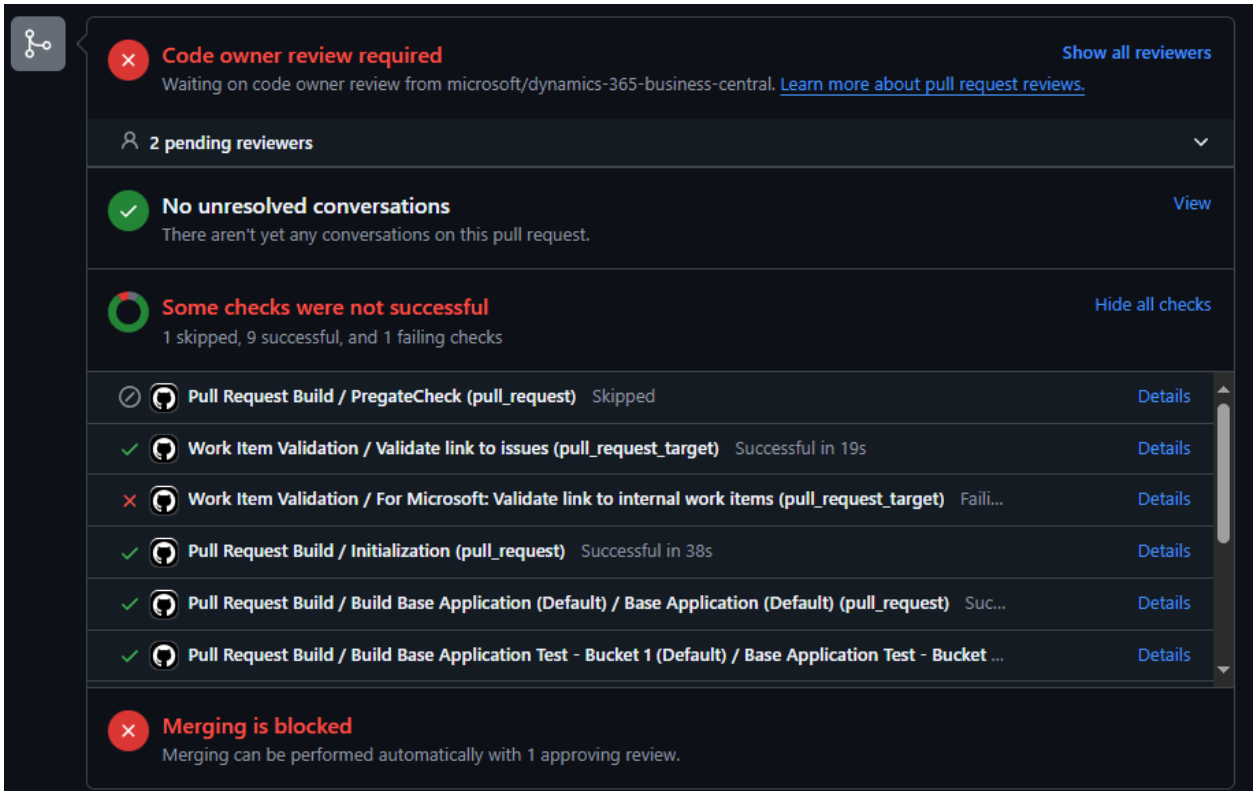
If your issue has not been approved (doesn't contain an "Approved" tag) CI will fail and automatically GitHub will put a comment on your Pull Request:





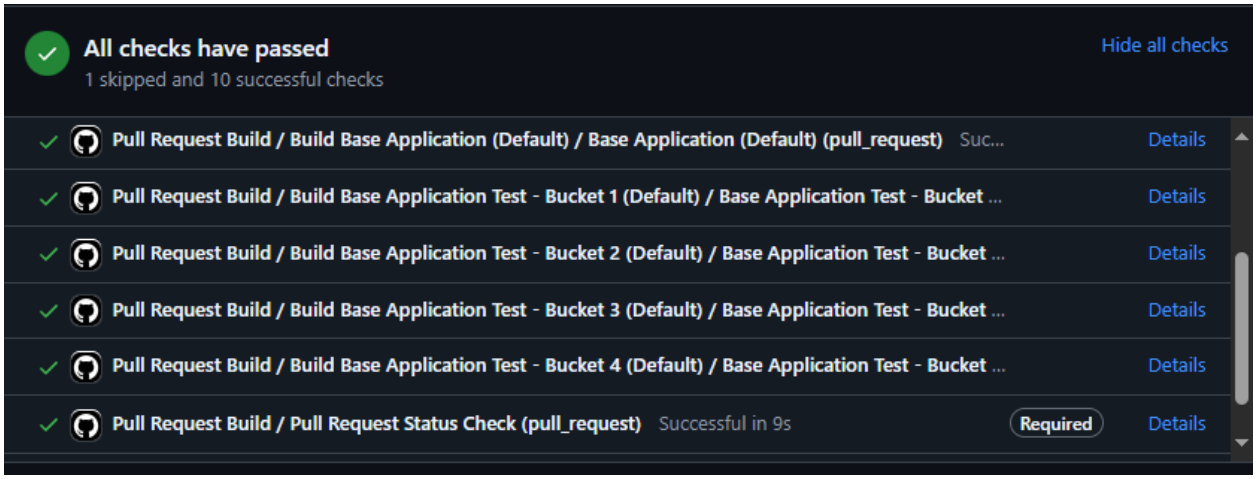
Missing internal work item





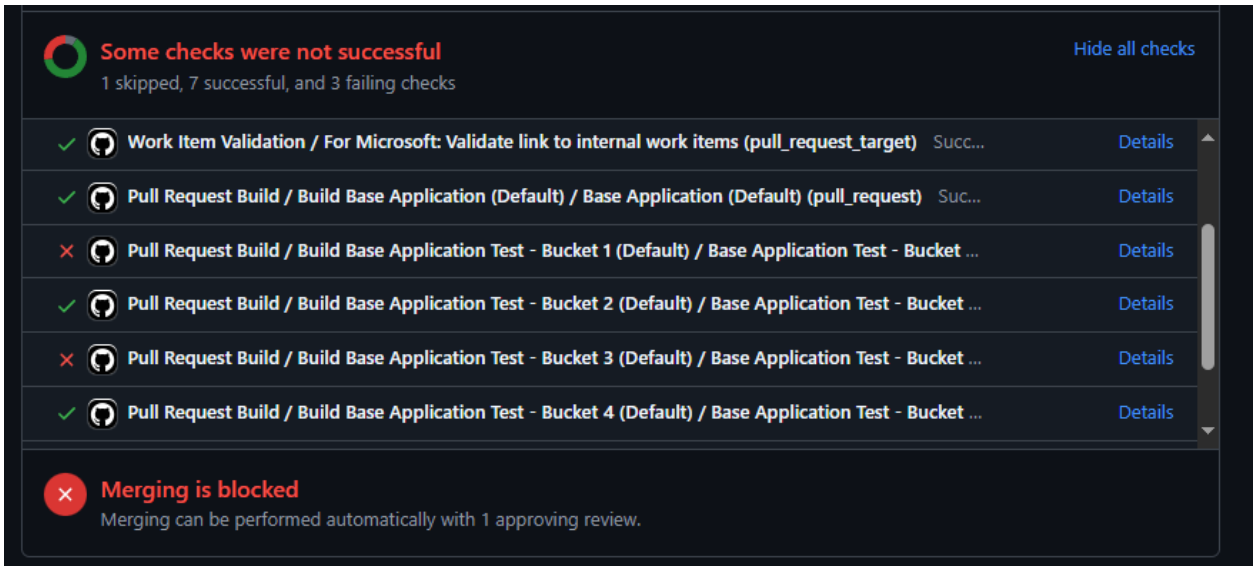
Build validation and tests

How it should be:



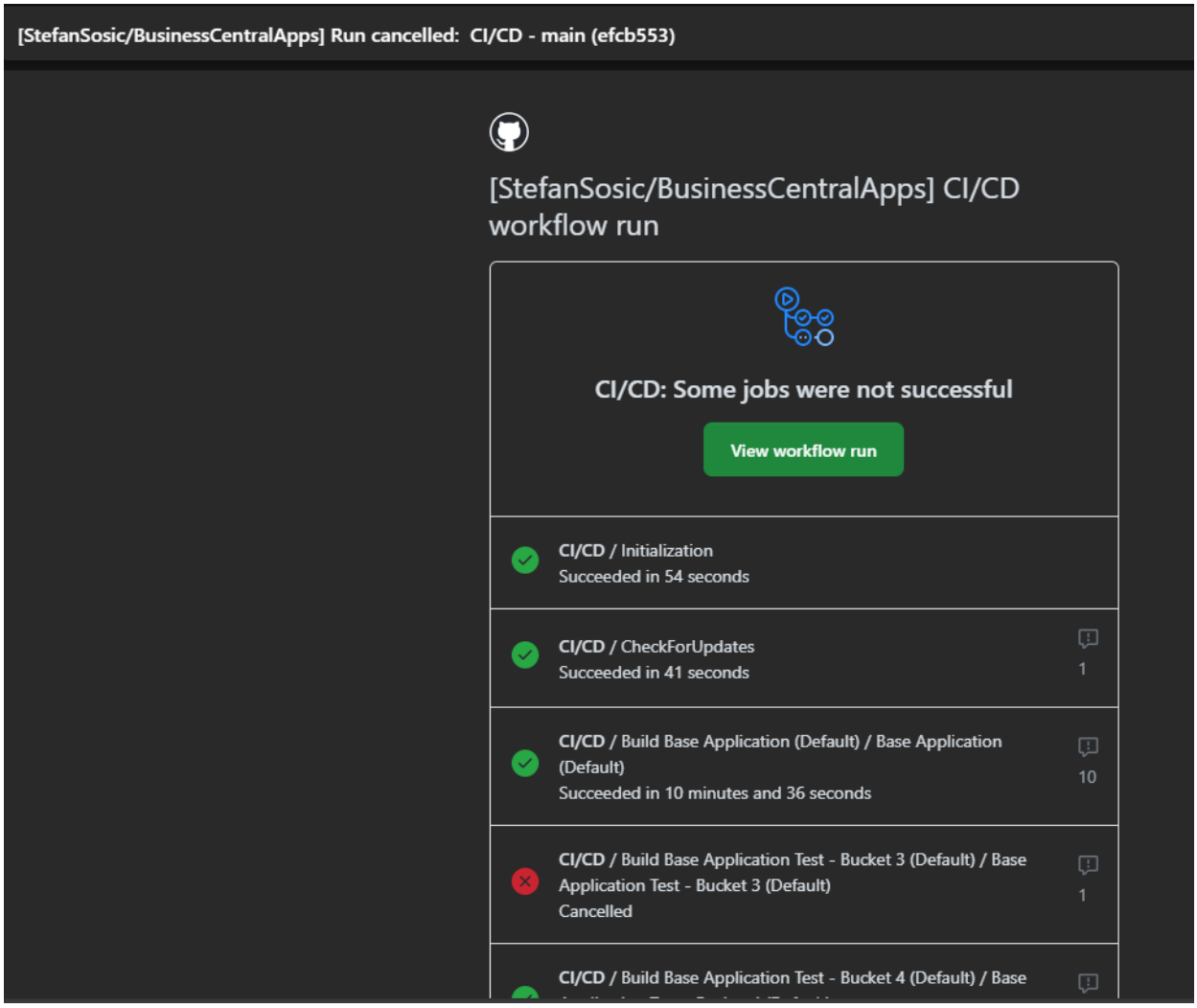
If you have any compilation errors or tests fail.

This is how it should not be, of course:

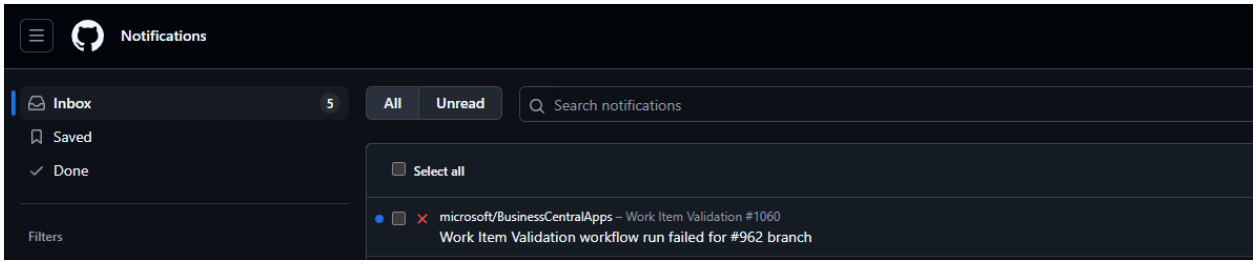


CI failed notifications

You will be informed by Email (if enabled on your GitHub settings):

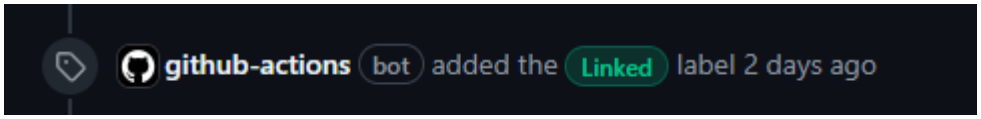


You will be informed by Notification (if enabled on your GitHub settings):



Pull Request Tags

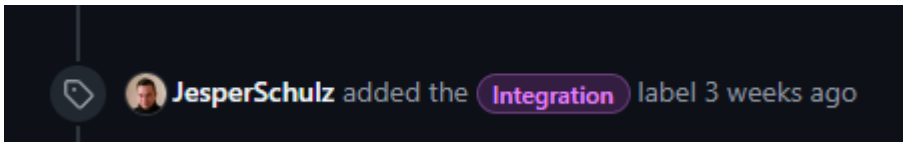
Linked

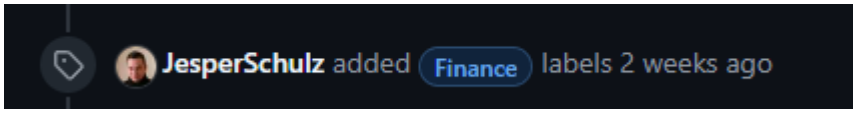


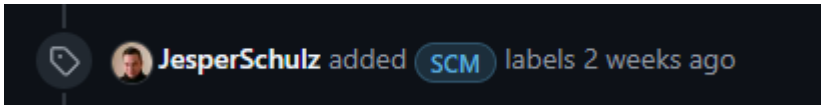
Automatically will be added if you have added an issue that is already approved also an internal Microsoft work item.

Integration/SCM/Finance and so on…

To which area your implementation does belong.



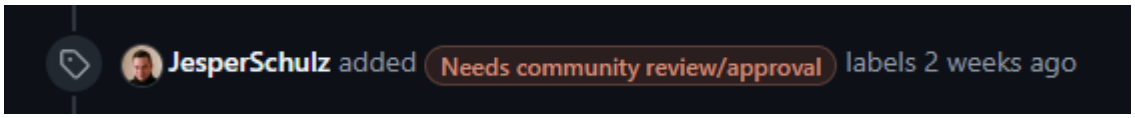




Follow Up

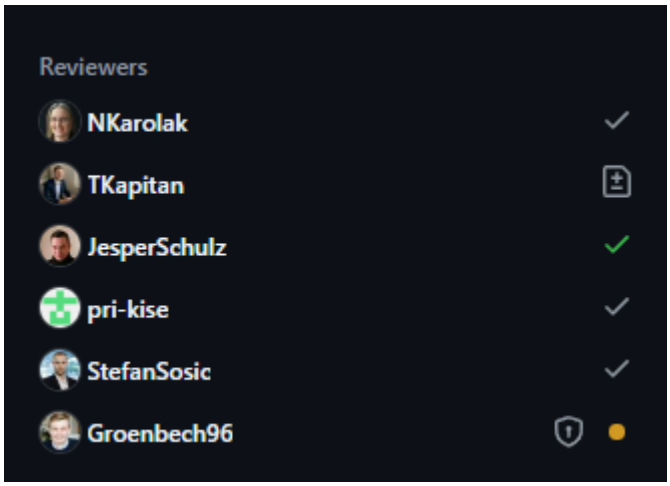
Due to complexity, there is active discussion either with the Pull Request creator or with other internal employees of Microsoft between departments. In order to get additional information and give input to the pull request creator.

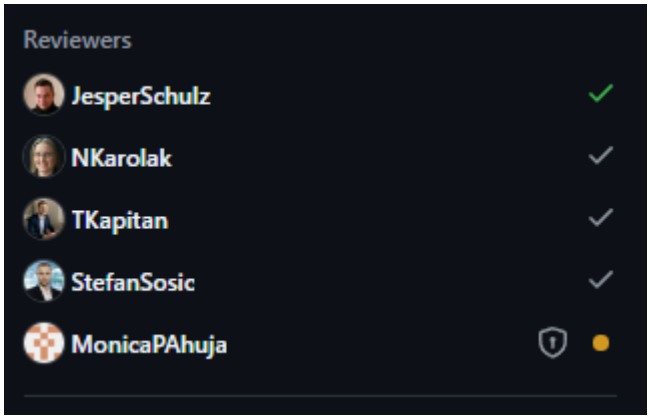
Needs community review/approval



Other community members are required to review your pull request.

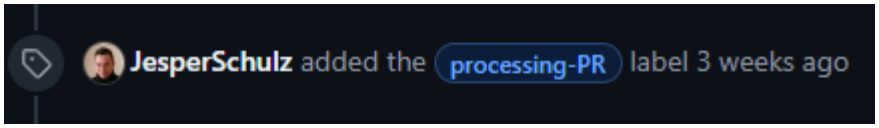
A standard Pull Request approval looks like this:



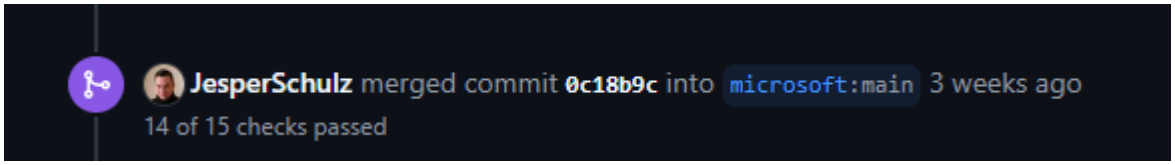


As you can see, a few community members and at least one Microsoft employee.

Processing PR



This means everything is ready so that your Pull Request gets into the “main” branch. Microsoft at this point internally runs all tests and validates your functionality and if all is right, your pull request gets merged.

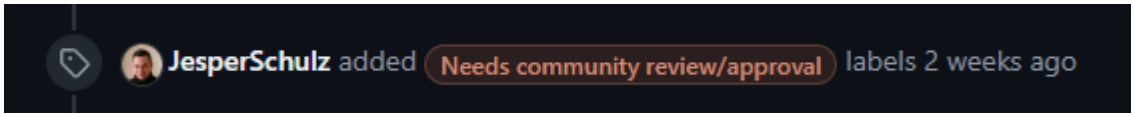


Other ways for contributions - Review of pull requests

Contributions are not only creation of solutions and bug fixes.

As valuable as that are Review of pull requests.

You may already seen before in this guide tag on Pull Request:



Believe it or not, reviews are a very important part of this collaboration. Your input is valuable and as many different people look at some Pull Requests, we will be making sure that the quality is really satisfied and that nothing has slipped through.

System Application

- To contribute to the System Application or Developer Tools, use the BCApps repository.

Processes described for Base Application are mostly the same as for System Application.

ALAppExtensions

- To contribute to Microsoft's 1st party apps, use the ALAppExtensions repository.
    - Use this repository also for any publisher requests, requests to change access modifiers (local/internal->public), and requests to replace option field/variable by enum. The official repository description:
        - Add new integration events – Get the event you need to hook into a process.
        - Change function visibility – For example, make a public function external or a similar change so you can call it from your extension and reuse the business logic.
        - Replace Option with Enum – Replace a specific option with an enum that supports your extension. The new type enum is extensible, but all code was written for non-extensible options.
        - Extensibility enhancements – Request changes in the application code that will improve extensibility.

Processes described for AL: App Extensions are mostly the same as for System Application.