

# Business Central

## Contribution Guide



November, 2024.

Stefan Šošić



## Contents

|  |    |
|--|----|
| How do contributions work? .....                                     | 4  |
| Open source repos.....   | 5  |
| Pitch new BC Idea .....  | 6  |
| Dynamics 365 Business Central Forum .....                            | 6  |
| GitHub Repository .....  | 8  |
| Creating a Bug .....   | 11 |
| Prerequisite to contributing to Base Application .....               | 14 |
| Contributing as developer .....                                      | 15 |
| First steps of preparation for starting developer contribution ..... | 16 |
| 1. Creating a fork on GitHub.....                                    | 16 |
| 1.1 Make sure your fork is up to date.....                           | 17 |
| 2. Cloning the GitHub repository.....                                | 18 |
| 3. Creating a local docker environment .....                         | 22 |
| 3.1. Running script for local docker environment.....                | 22 |
| 3.2. Downloading AL Packages .....                                   | 25 |
| 3.3. Defining assembly probing path for DotNet's .....               | 26 |
| 3.4. Use the correct AL Language extension for VS Code.....          | 28 |
| 3.5. Other possible errors .....                                     | 29 |
| Branch policy on your fork.....                                      | 29 |
| Branch naming .....  | 30 |
| Testing your code.....   | 30 |
| New AL Files.....  | 32 |
| 1. Object range.....   | 32 |
| 2. Object header with copyright information .....                    | 32 |
| 3. Creating a new module .....                                       | 32 |
| Creating your first Pull Request .....                               | 33 |
| Take the issue on GitHub .....                                       | 33 |
| Create a branch .....  | 34 |
| Pre-Pull Request .....   | 34 |
| Create Pull Request .....  | 35 |
| Contribution License Agreement .....                                 | 40 |
| Pull Request CIs .....   | 41 |
| Issue not approved.....  | 41 |
| Missing internal work item .....                                     | 42 |
| Build validation and tests .....                                     | 43 |

|   |    |
|---|----|
| CI failed notifications .....                               | 44 |
| Pull Request Tags.....                                      | 45 |
| Other ways for contributions - Review of pull requests..... | 47 |
| System Application .....                                    | 48 |
| ALAppExtensions .....                                       | 48 |

## 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.



Accelerate Product Growth



Improve Product Quality & Auditability



Enhance Extensibility, Customizability & Compatibility



Foster Collaboration



Develop Community

### 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

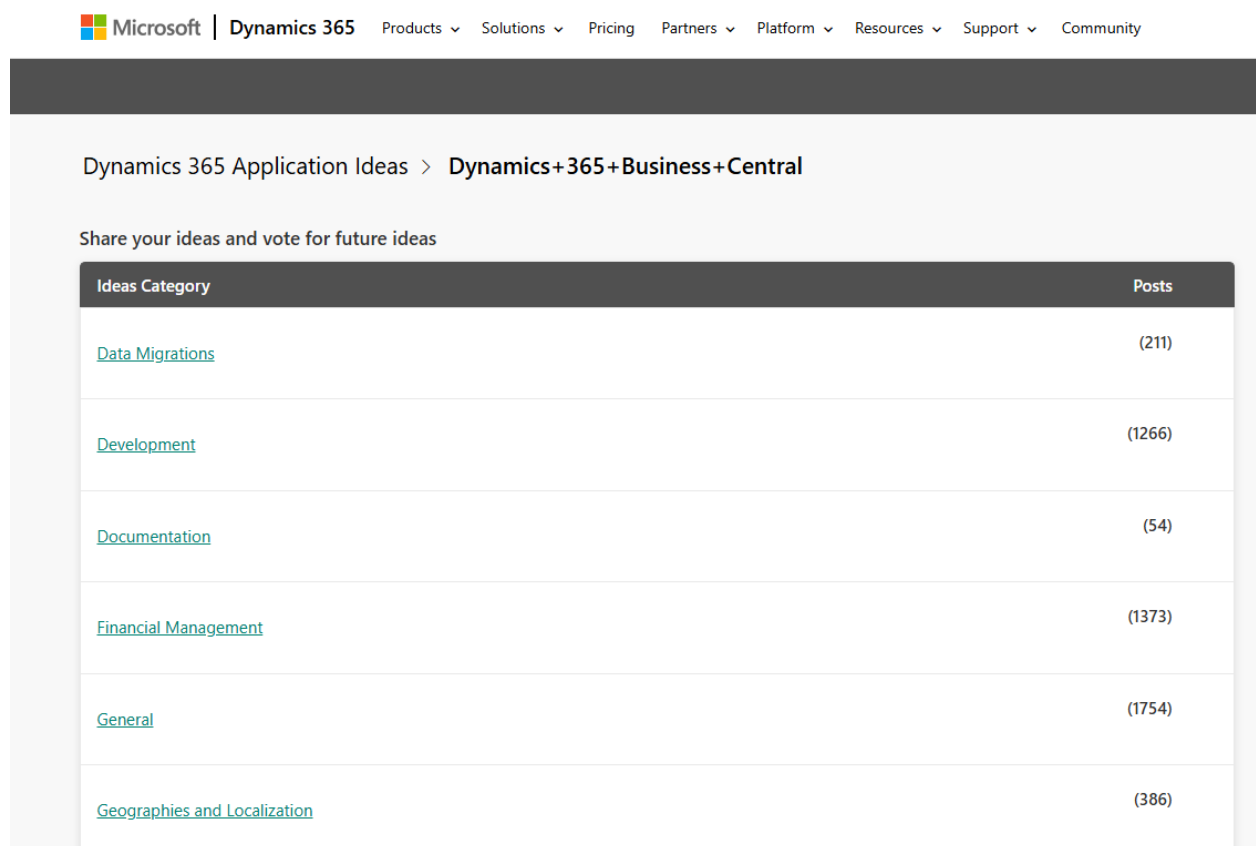
There are two ways to pitch a new BC Idea, through the Dynamics 365 Business Central forum or the GitHub repository.

### 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.



| Ideas Category                               | Posts  |
|--|--------|
| <a href="#">Data Migrations</a>              | (211)  |
| <a href="#">Development</a>                  | (1266) |
| <a href="#">Documentation</a>                | (54)   |
| <a href="#">Financial Management</a>         | (1373) |
| <a href="#">General</a>                      | (1754) |
| <a href="#">Geographies and Localization</a> | (386)  |

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:

STATUS DETAILS

NEW

STATUS DETAILS

UNDER REVIEW



Administrator

Thank you for your feedback. We are considering adding it to our longer term roadmap. Your help is greatly appreciated!

STATUS DETAILS

PLANNED



Administrator

Thank you for your feedback. We are considering adding it to our longer term roadmap.

STATUS DETAILS

NEEDS VOTES



Administrator

Thank you for this suggestion! Currently this is not on our roadmap. We are tracking this idea and if it gathers more votes and comments we will consider it in the future.

STATUS DETAILS

COMPLETED



Administrator

Thank you!

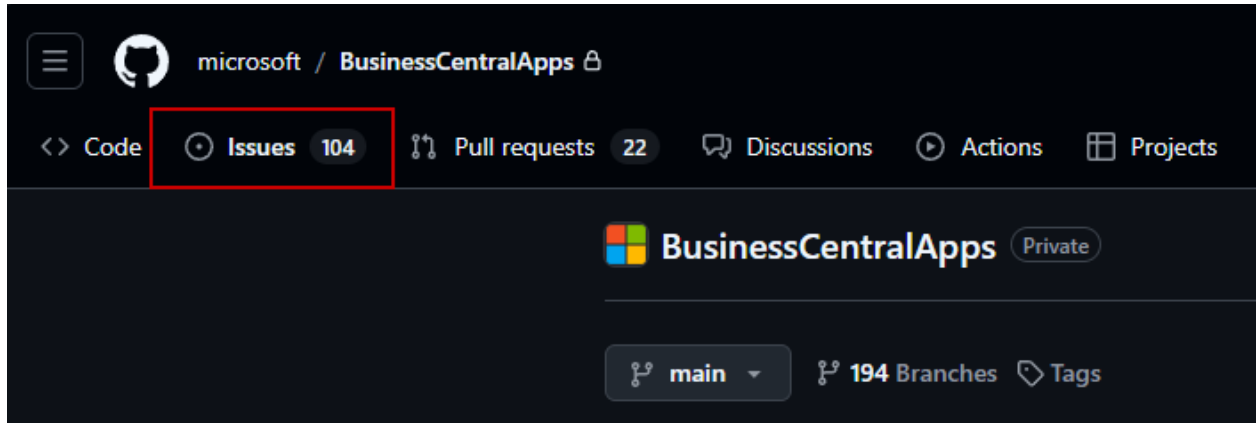
This feature is now released: <https://learn.microsoft.com/dynamics365/release-plan/2023wave2/smb/dynamics365-business-central/scan-barcodes-business-central-mobile-app-ios-android>

## GitHub Repository

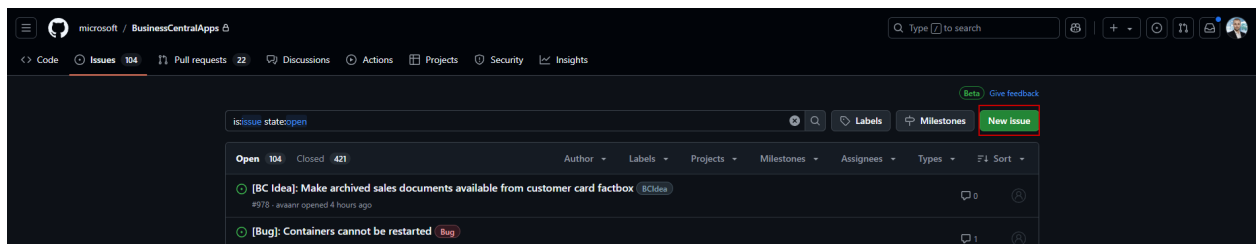
When your idea gains enough support from the community on the forum, you can then create a BC Idea directly in the repository related to that idea.

In order to do that, follow next steps:

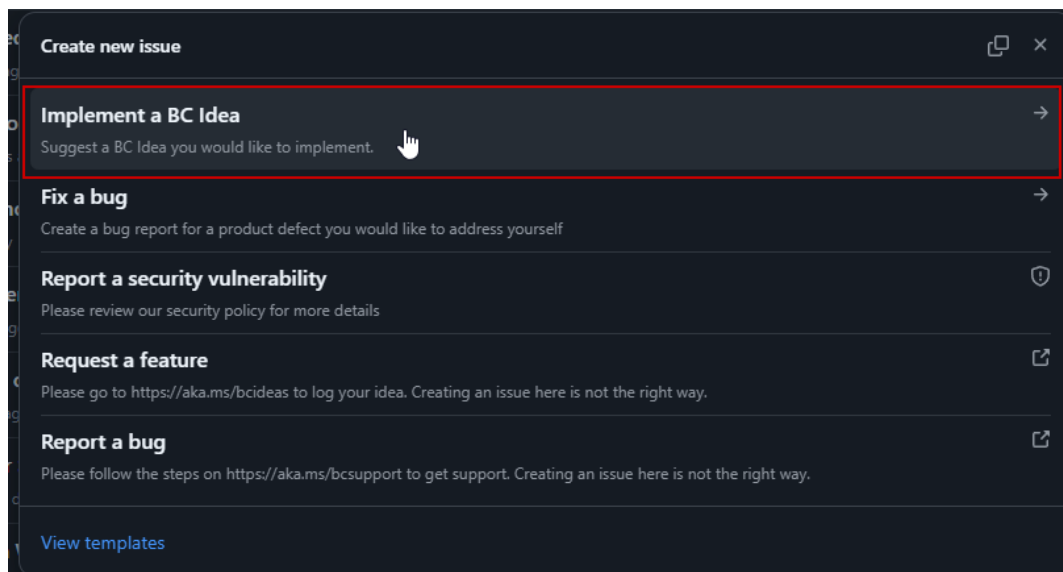
1. Go to GitHub repository



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 \***

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](https://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

### Description

Please include the description from the BC Ideas page

Write Preview

H B I | ☰ <> 🔗 | ☰ ☰ ☰ | @ ↻ ↶ ↷

Description of BC Idea

📎 Paste, drop, or click to add files

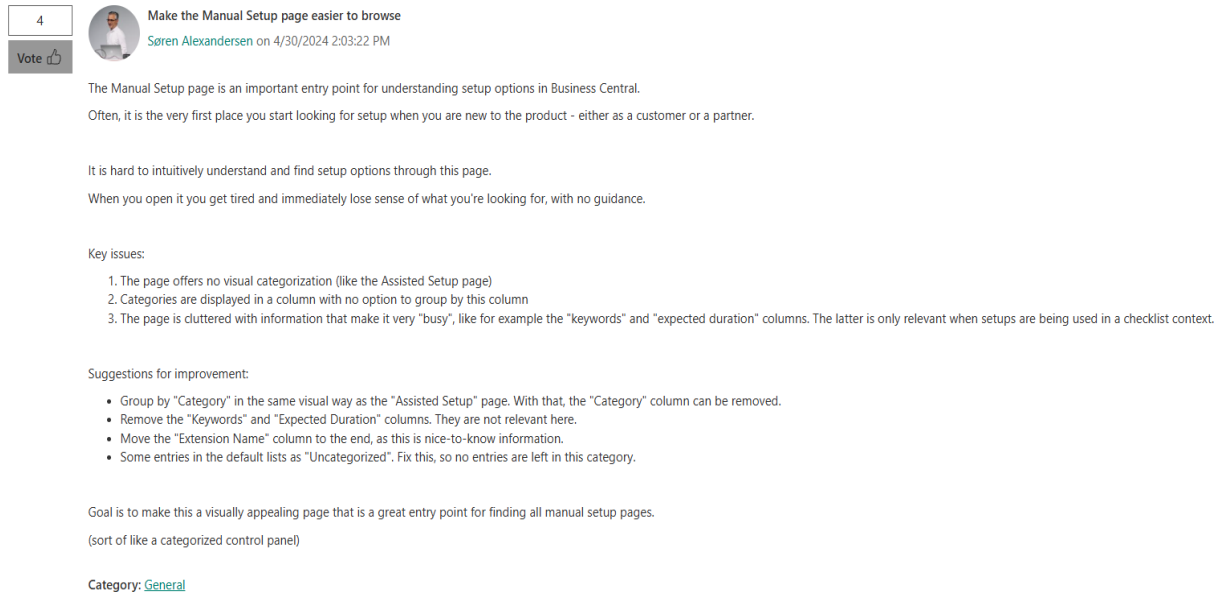
I will provide the implementation for this BC Idea

To provide the implementation for a BC idea, select this checkbox and then Get started. Thanks for contributing.

For example:

On Forum BC Idea is created:

Dynamics 365 Application Ideas > Dynamics 365 Business Central > General > Make the Manual Setup page easier to browse



4  
Vote

**Make the Manual Setup page easier to browse**  
Søren Alexandersen on 4/30/2024 2:03:22 PM

The Manual Setup page is an important entry point for understanding setup options in Business Central. Often, it is the very first place you start looking for setup when you are new to the product - either as a customer or a partner.

It is hard to intuitively understand and find setup options through this page.

When you open it you get tired and immediately lose sense of what you're looking for, with no guidance.

Key issues:

1. The page offers no visual categorization (like the Assisted Setup page)
2. Categories are displayed in a column with no option to group by this column
3. The page is cluttered with information that make it very "busy", like for example the "keywords" and "expected duration" columns. The latter is only relevant when setups are being used in a checklist context.

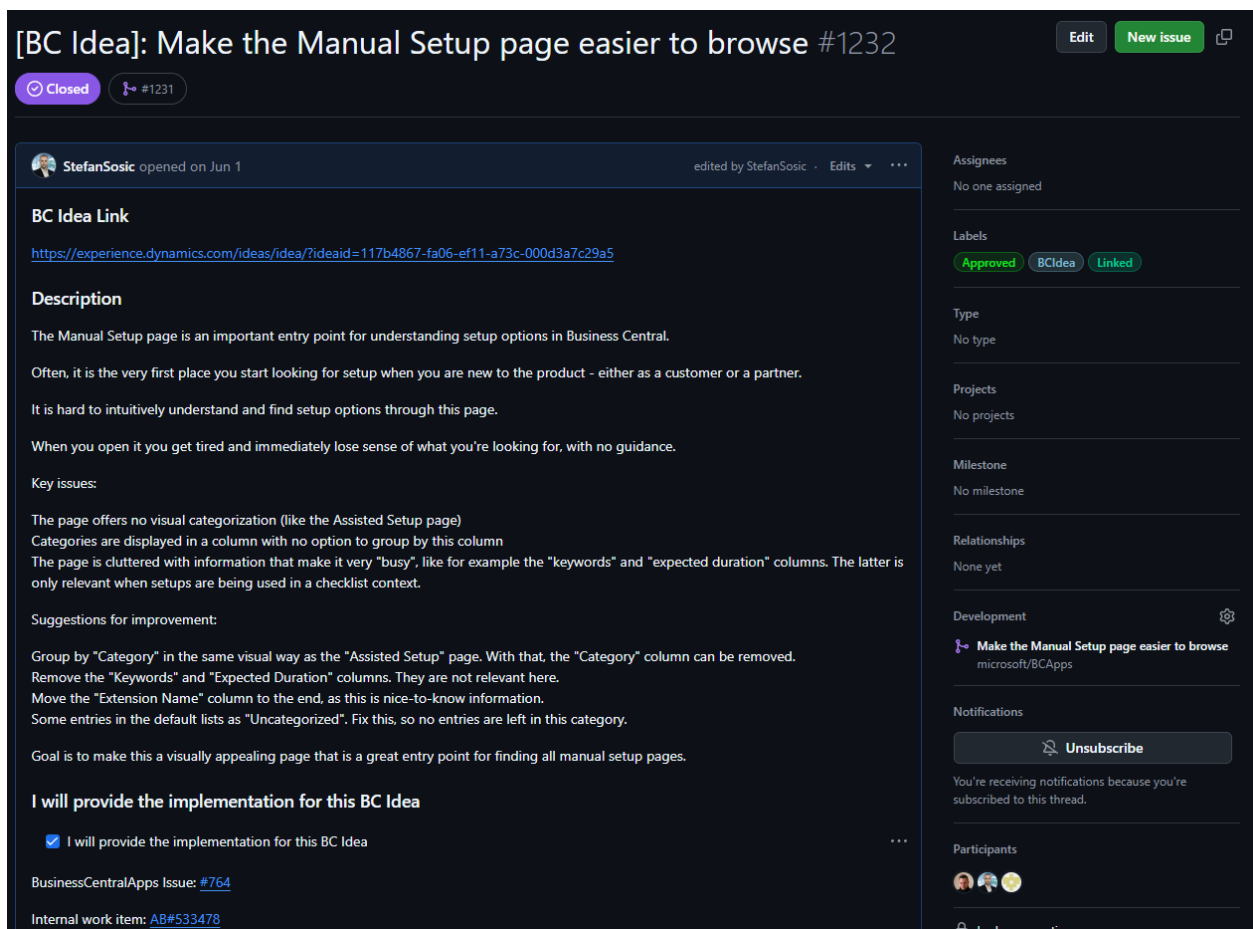
Suggestions for improvement:

- Group by "Category" in the same visual way as the "Assisted Setup" page. With that, the "Category" column can be removed.
- Remove the "Keywords" and "Expected Duration" columns. They are not relevant here.
- Move the "Extension Name" column to the end, as this is nice-to-know information.
- Some entries in the default lists as "Uncategorized". Fix this, so no entries are left in this category.

Goal is to make this a visually appealing page that is a great entry point for finding all manual setup pages. (sort of like a categorized control panel)

Category: [General](#)

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



**[BC Idea]: Make the Manual Setup page easier to browse #1232** Edit New issue

Closed #1231

StefanSosic opened on Jun 1 edited by StefanSosic · Edits · · ·

**BC Idea Link**  
<https://experience.dynamics.com/ideas/idea/?ideaId=117b4867-fa06-ef11-a73c-000d3a7c29a5>

**Description**

The Manual Setup page is an important entry point for understanding setup options in Business Central.

Often, it is the very first place you start looking for setup when you are new to the product - either as a customer or a partner.

It is hard to intuitively understand and find setup options through this page.

When you open it you get tired and immediately lose sense of what you're looking for, with no guidance.

Key issues:

The page offers no visual categorization (like the Assisted Setup page)  
Categories are displayed in a column with no option to group by this column  
The page is cluttered with information that make it very "busy", like for example the "keywords" and "expected duration" columns. The latter is only relevant when setups are being used in a checklist context.

Suggestions for improvement:

Group by "Category" in the same visual way as the "Assisted Setup" page. With that, the "Category" column can be removed.  
Remove the "Keywords" and "Expected Duration" columns. They are not relevant here.  
Move the "Extension Name" column to the end, as this is nice-to-know information.  
Some entries in the default lists as "Uncategorized". Fix this, so no entries are left in this category.

Goal is to make this a visually appealing page that is a great entry point for finding all manual setup pages.

**I will provide the implementation for this BC Idea**

I will provide the implementation for this BC Idea

BusinessCentralApps Issue: [#764](#)

Internal work item: [AB#533478](#)

Assignees: No one assigned

Labels: **Approved** BCidea Linked

Type: No type

Projects: No projects

Milestone: No milestone

Relationships: None yet

Development: **Make the Manual Setup page easier to browse** microsoft/BCApps

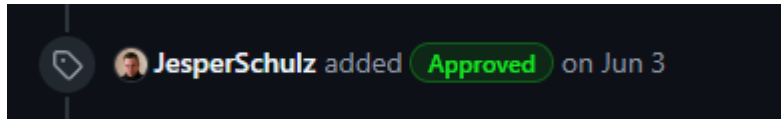
Notifications: **Unsubscribe**

You're receiving notifications because you're subscribed to this thread.

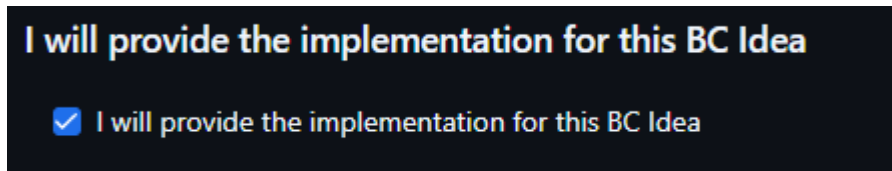
Participants: [User avatars]

Lock conversation

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

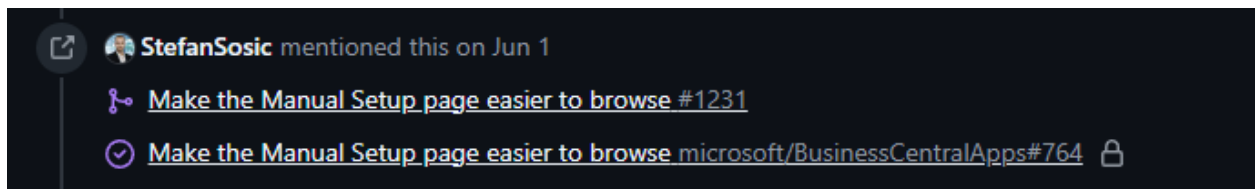


Afterward, if you ticked the checkbox indicating that you will provide the implementation of the BC idea:

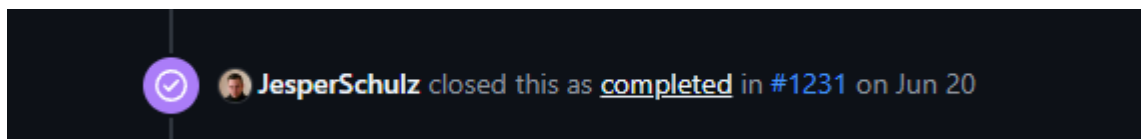


You could start to work on it, if not then some of the other community contributors will take it and provide an implementation for it.

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:

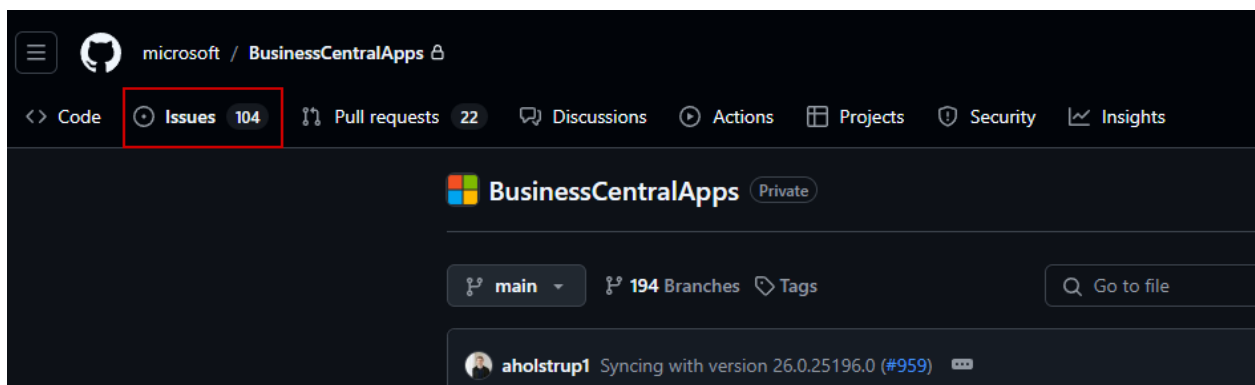


## Creating a Bug

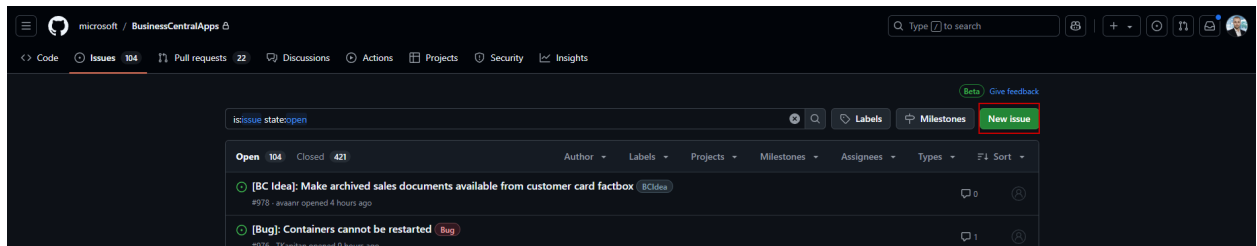
If you find the bug in Business Central lying around for some time, you could report it, get approval, and submit your fix for it.

The process of reporting a bug is quite straightforward.

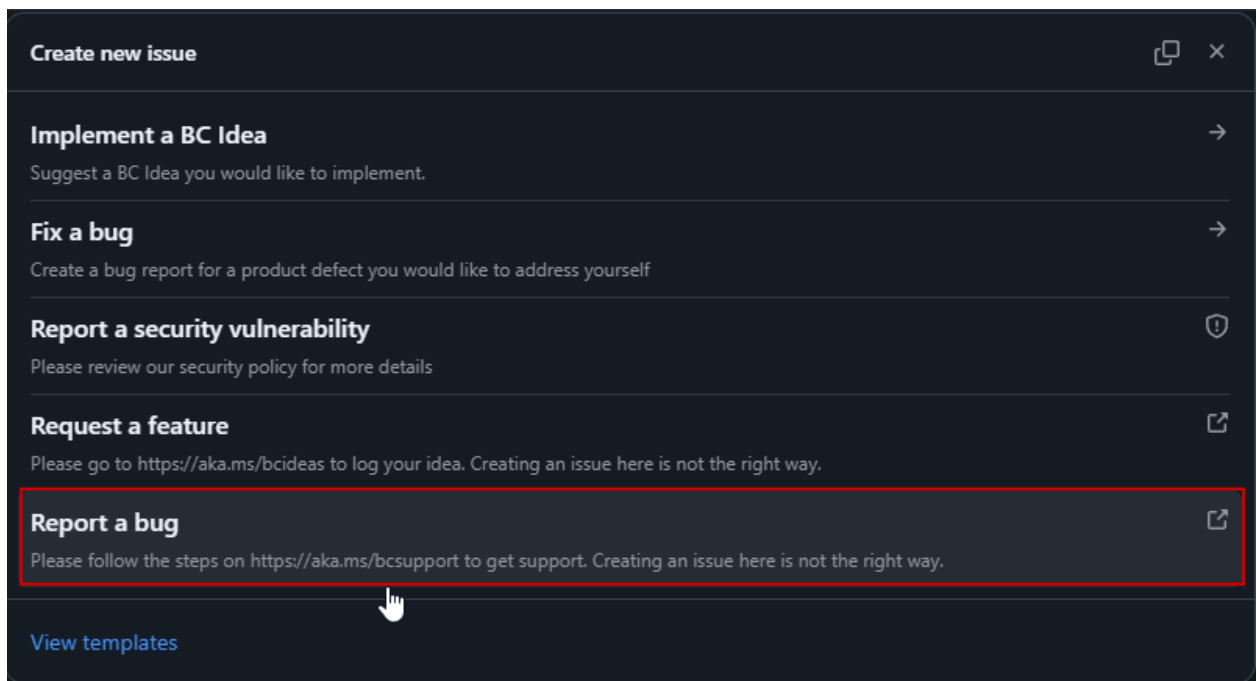
Go to the responsible repository where the bug exists and go to the “Issues” tab:



Then click on the “New Issue” button

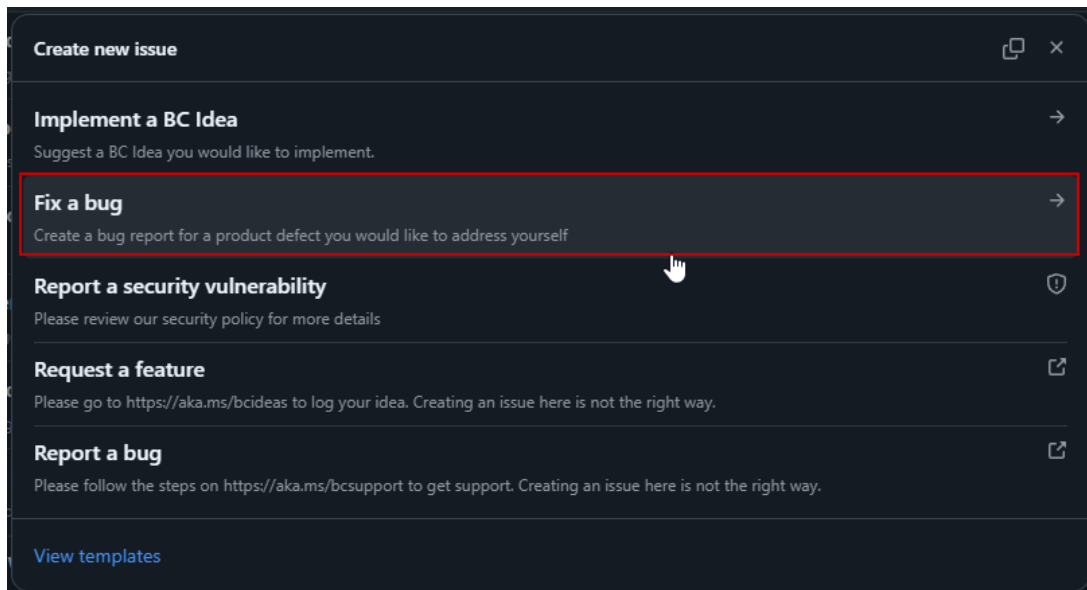


Keep in mind that it's not the right way for all issues to create an issue on the git hub repository:

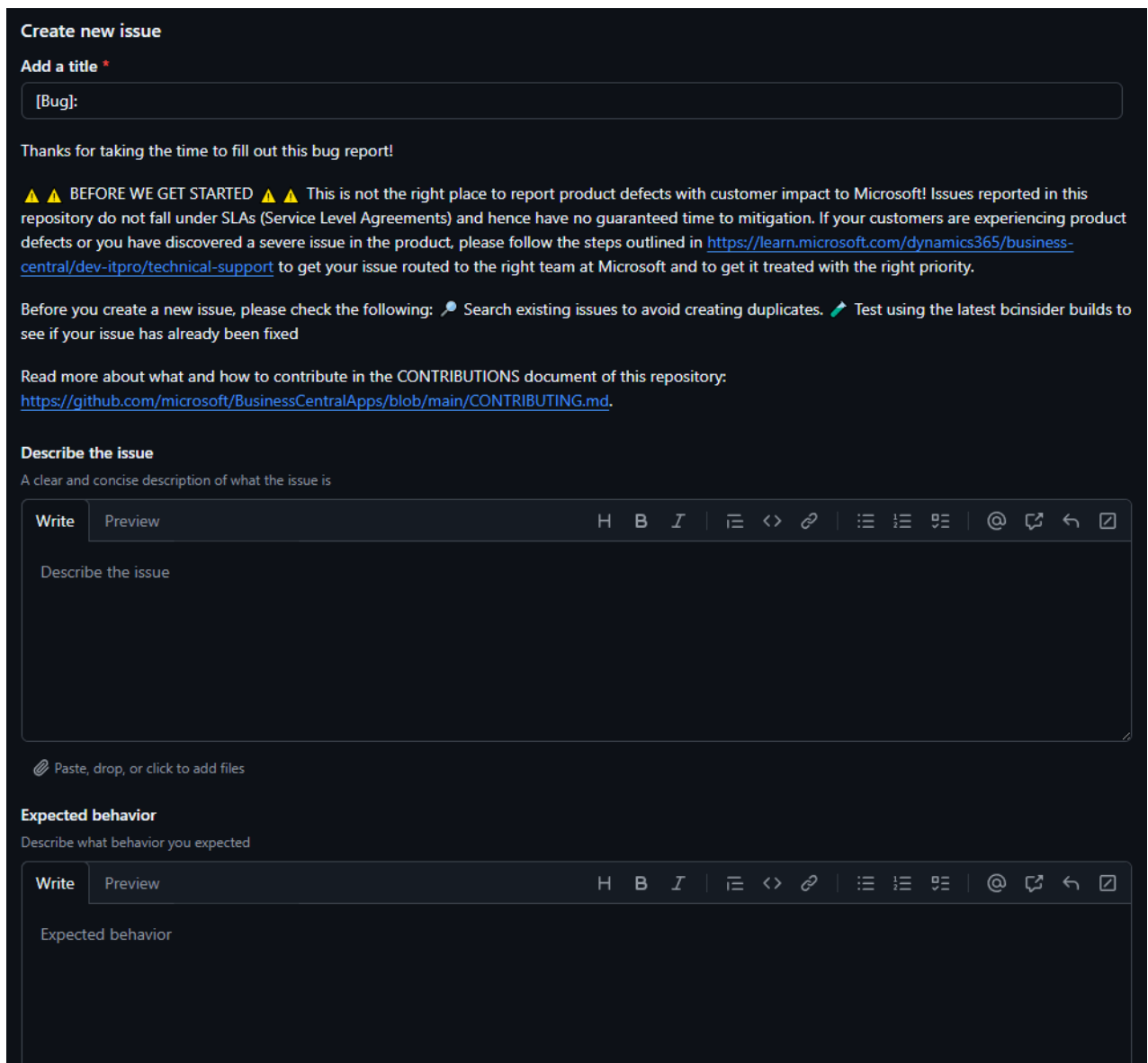


You can navigate to “Report a bug” and read more on the docs page.

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:



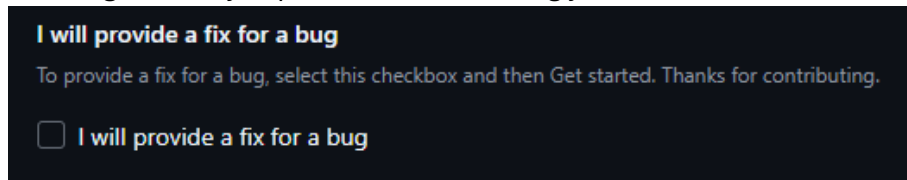
New page will open where you would need to put details for a bug which you are reporting:

A screenshot of the 'Create new issue' form for reporting a bug. The form has a title 'Create new issue' and a sub-header 'Add a title \*'. Below this is a text input field containing '[Bug:'. Underneath, there is a thank you message: 'Thanks for taking the time to fill out this bug report!'. This is followed by a warning: '⚠️ ⚠️ BEFORE WE GET STARTED ⚠️ ⚠️ This is not the right place to report product defects with customer impact to Microsoft! Issues reported in this repository do not fall under SLAs (Service Level Agreements) and hence have no guaranteed time to mitigation. If your customers are experiencing product defects or you have discovered a severe issue in the product, please follow the steps outlined in https://learn.microsoft.com/dynamics365/business-central/dev-itpro/technical-support to get your issue routed to the right team at Microsoft and to get it treated with the right priority.' Below the warning, there are instructions: 'Before you create a new issue, please check the following: 🔍 Search existing issues to avoid creating duplicates. 🚀 Test using the latest bcinsider builds to see if your issue has already been fixed'. Then, there is a link to 'Read more about what and how to contribute in the CONTRIBUTIONS document of this repository: https://github.com/microsoft/BusinessCentralApps/blob/main/CONTRIBUTING.md.'. The form then has two sections: 'Describe the issue' and 'Expected behavior'. Each section has a sub-header and a description of what to write. Below each section is a rich text editor with a 'Write' tab and a 'Preview' tab. The rich text editor for 'Describe the issue' has a toolbar with icons for bold, italic, link, code, list, and other formatting options. The same structure is repeated for the 'Expected behavior' section.



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/JoinBCCcodevelopmentPilot>

A screenshot of a web form titled "Join the Microsoft Dynamics 365 Business Central Base Application contribution pilot". The form includes a header with a person and rocket icon, a paragraph explaining the purpose, a list of required fields, and two input boxes. The first field is labeled "1. Full name \*" and the second is "2. Email address \*". Both input boxes contain the placeholder text "Enter your answer".

**Join the Microsoft Dynamics 365 Business Central Base Application contribution pilot**

Do you want to get access to the <https://github.com/microsoft/BusinessCentralApps> repository and contribute to the development of the Business Central application platform, then simply fill out the form below and we will get you started within a few days!

\* Required

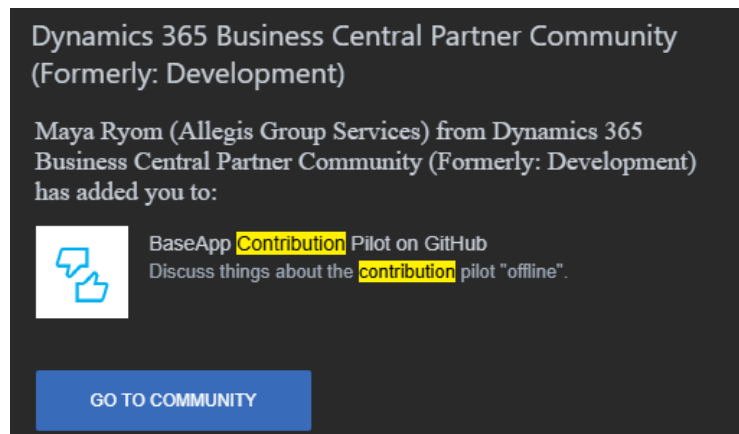
1. Full name \*

Enter your answer

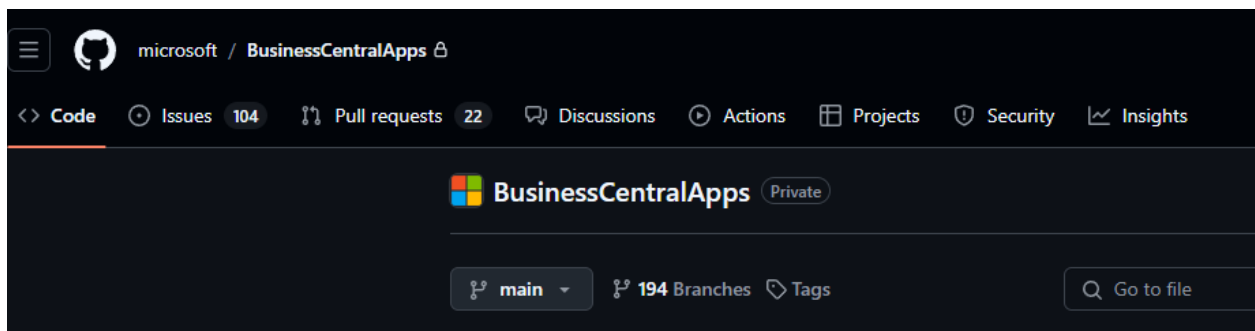
2. Email address \*

Enter your answer

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

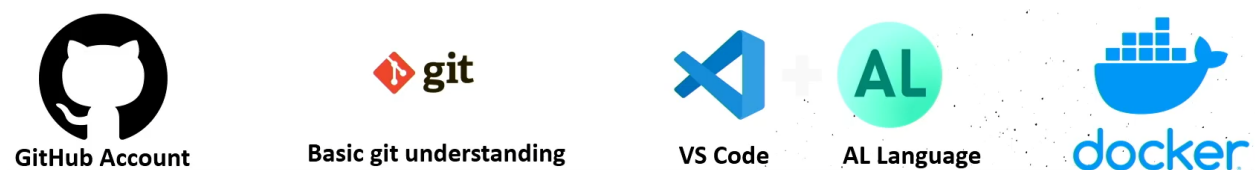


And also the most important, GitHub repository:



## Contributing as developer

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



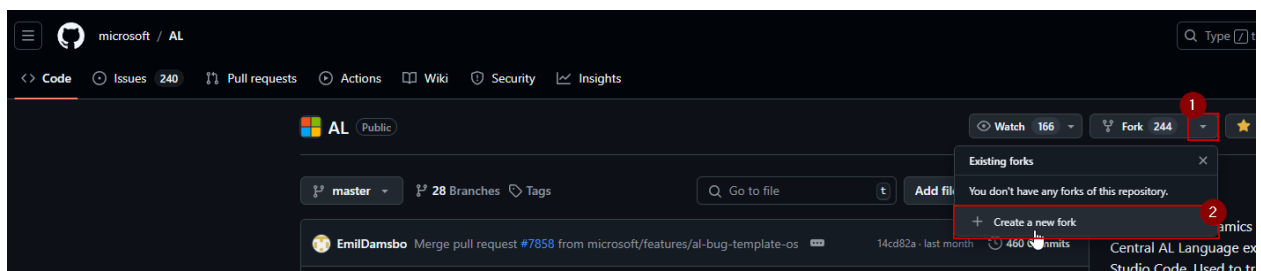
## First steps of preparation for starting developer contribution

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:


## Create a new fork

A *fork* is a copy of a repository. Forking a repository allows you to freely experiment with changes without affecting the original project. [View existing forks.](#)

*Required fields are marked with an asterisk (\*).*

**Owner \*** **Repository name \***

 StefanSosic / AL

 AL is available.


By default, forks are named the same as their upstream repository. You can customize the name to distinguish it further.

**Description** (optional)

Home of the Dynamics 365 Business Central AL Language extension for Visual Studio Code. Used to track issues

**Copy the `master` branch only**

Contribute back to microsoft/AL by adding your own branch. [Learn more.](#)

 You are creating a fork in your personal account.

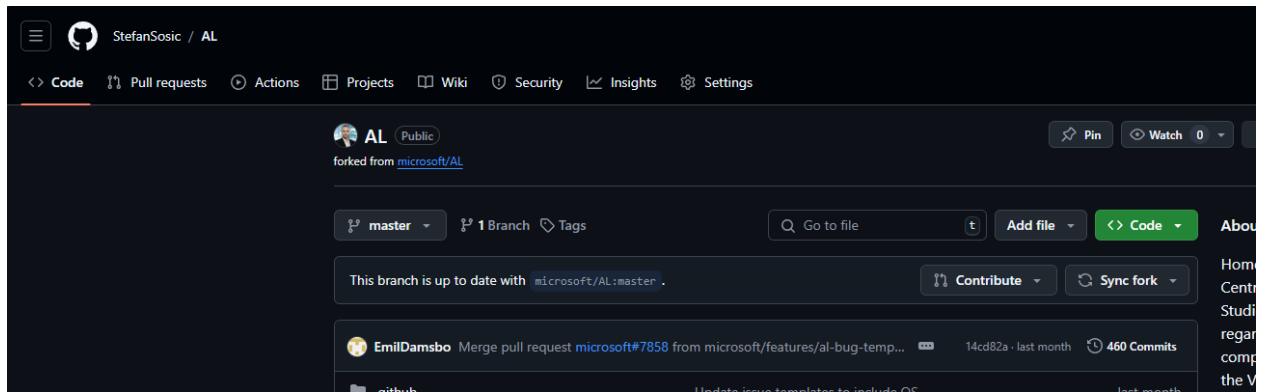
[Create fork](#)

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.

*(this guide is Pre-Release version)*

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

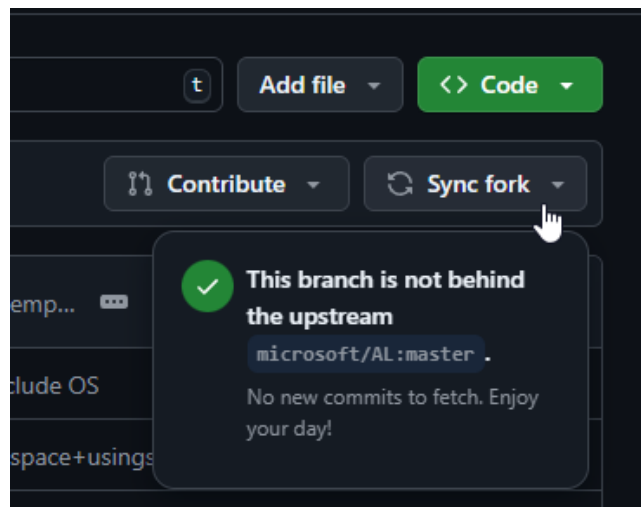


It is basically a copy of the original repository.

### 1.1 Make sure your fork is up to date

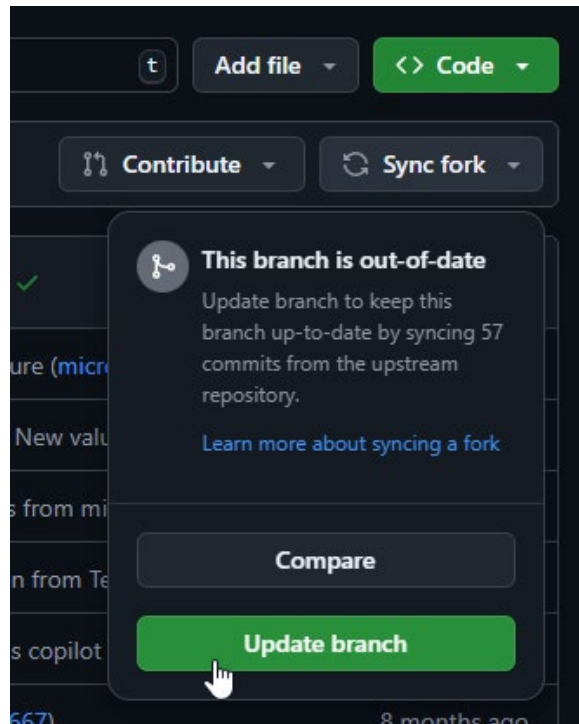
In order to avoid any merge conflict and to work on the most recent code, you will need to keep your fork up to date.

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:

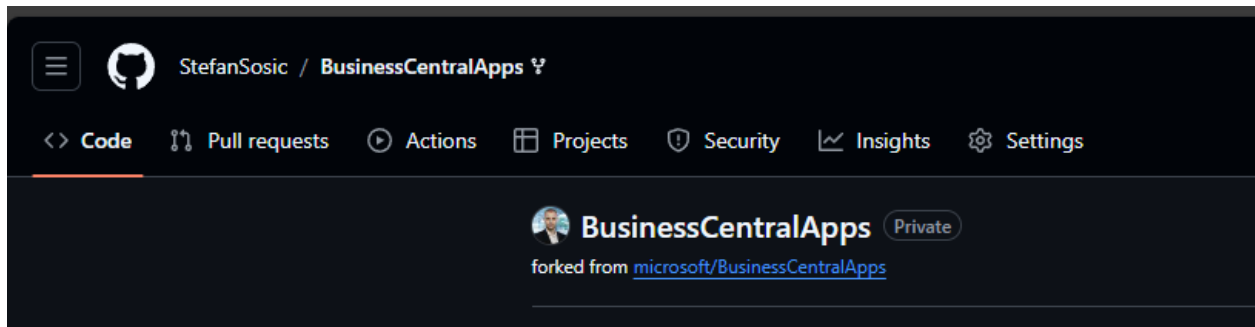


Click on “Update branch” before proceeding any work.

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

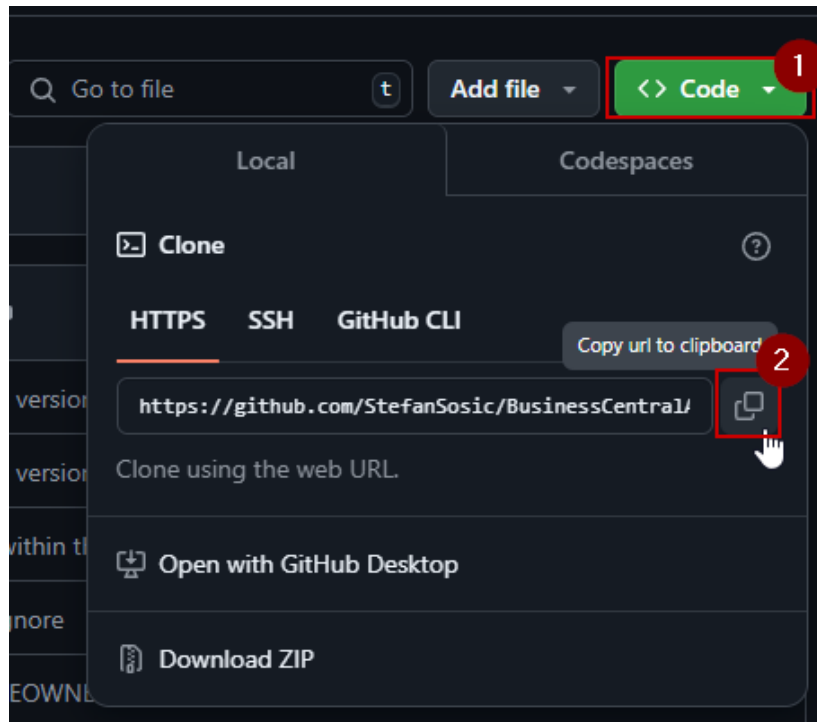
## 2. Cloning the GitHub repository

Make sure you are on your own fork:

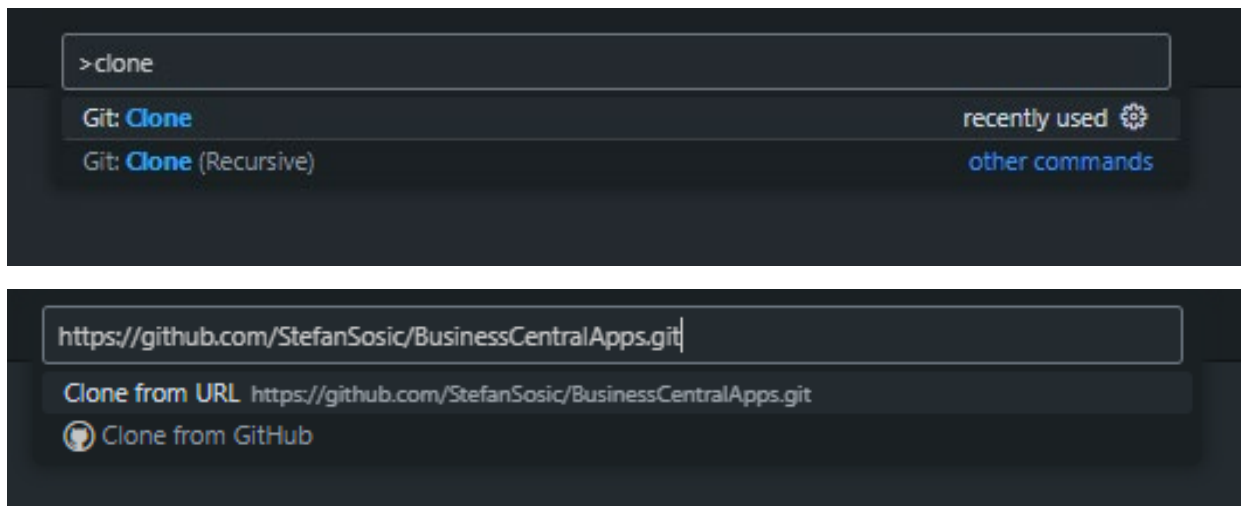




Click on the “Code” button and copy the URL:

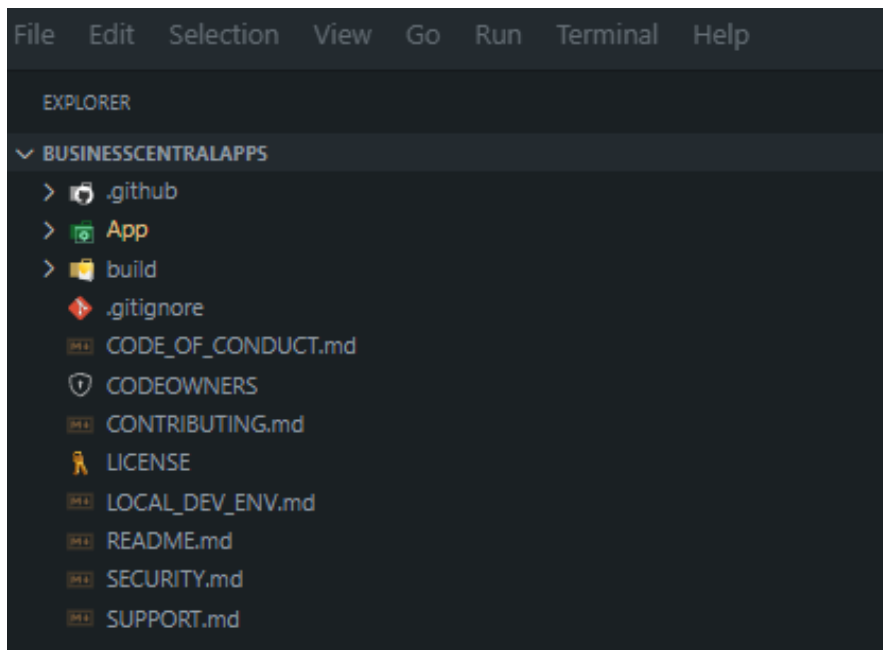


Open Visual Studio Code and open the command prompt. There you type “clone” and press “Enter”:

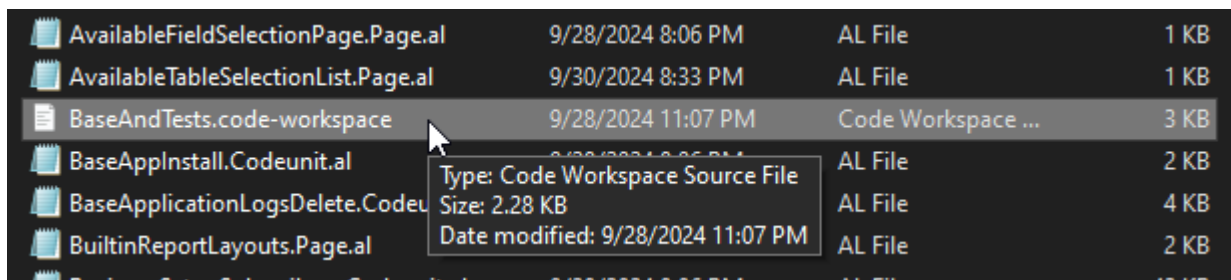


Paste your fork URL copied before and press “Enter”.

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



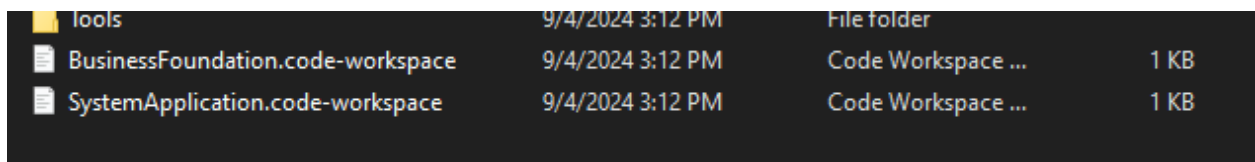
This is not the right way to open your development environment. Each repository has its own Visual Studio workspace file.



**For Base Application it's under path:**

BusinessCentralApps\App\Layers\W1\BaseApp

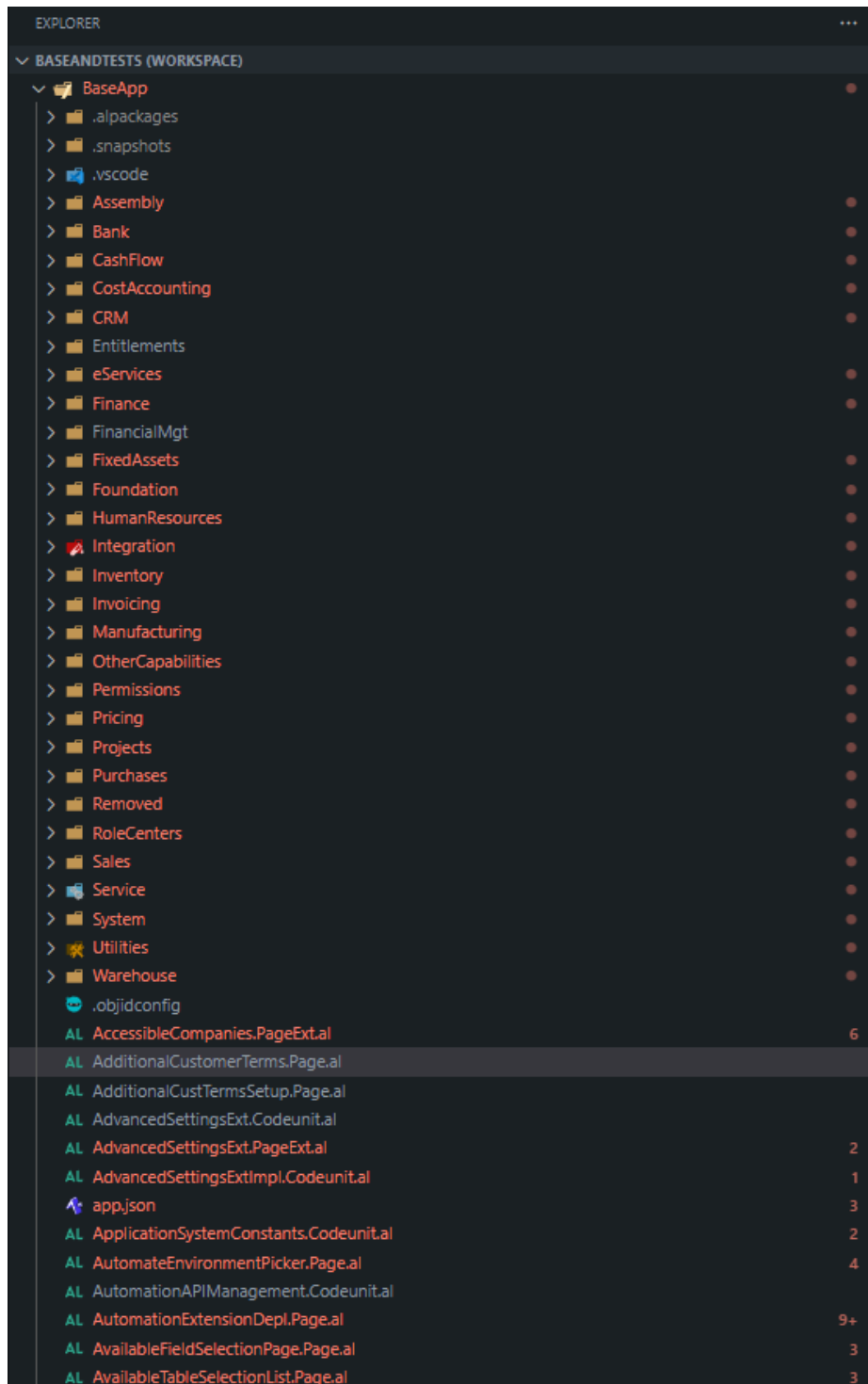
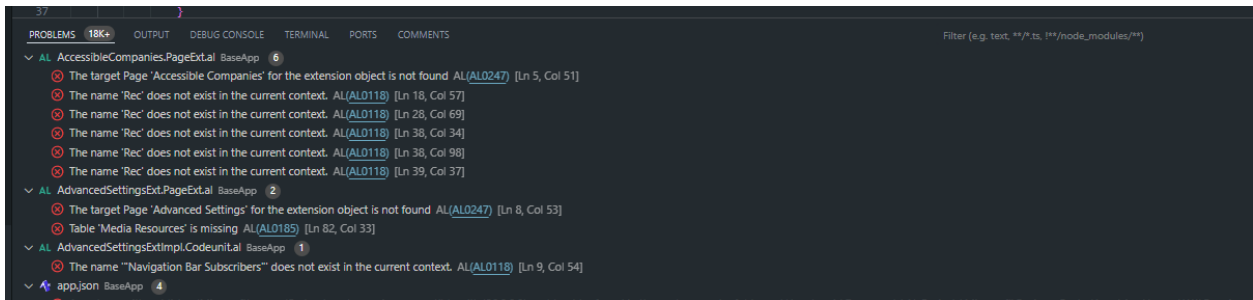
**For System Application, BCApps\src:**



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

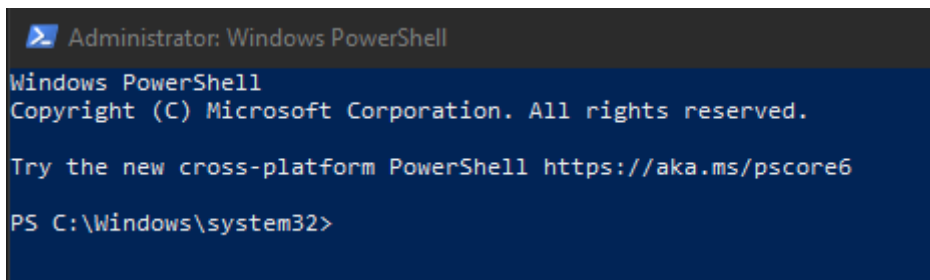
I will focus as before on Base Application. Basically, the process is the same on all repositories. Since our focus is on .AL-Go folder which is basically created out of a template and then adjusted for each repository.

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:



```
Administrator: Windows PowerShell
Windows PowerShell
Copyright (C) Microsoft Corporation. All rights reserved.

Try the new cross-platform PowerShell https://aka.ms/powershell

PS C:\Windows\system32>
```

Go to the path and execute the script:

```

Administrator: Windows PowerShell
Windows PowerShell
Copyright (C) Microsoft Corporation. All rights reserved.

Try the new cross-platform PowerShell https://aka.ms/pscore6

PS C:\Windows\system32> cd "C:\Users\sosic\Documents\AL Projects\Microsoft\BusinessCentralApps\App\Projects\Base Application\AL-Go"
PS C:\Users\sosic\Documents\AL Projects\Microsoft\BusinessCentralApps\App\Projects\Base Application\AL-Go> dir

    Directory: C:\Users\sosic\Documents\AL Projects\Microsoft\BusinessCentralApps\App\Projects\Base Application\AL-Go

Mode                LastWriteTime         Length Name
----                -
-a----            9/28/2024   8:06 PM           4030 cloudDevEnv.ps1
-a----            9/28/2024   8:06 PM           178 ImportTestDataInBcContainer.ps1
-a----            9/28/2024   8:06 PM           366 ImportTestToolkitToBcContainer.ps1
-a----            9/28/2024   8:06 PM           6527 localDevEnv.ps1
-a----            9/28/2024   8:06 PM          2070 localDevEnv.settings.json
-a----            9/28/2024   8:06 PM           165 NewBcContainer.ps1
-a----            9/28/2024   8:06 PM           223 PreCompileApp.ps1
-a----            9/28/2024   8:06 PM           164 PublishBcContainerApp.ps1
-a----            9/28/2024   8:06 PM           164 RunTestsInBcContainer.ps1
-a----            9/28/2024   8:06 PM           361 settings.json

PS C:\Users\sosic\Documents\AL Projects\Microsoft\BusinessCentralApps\App\Projects\Base Application\AL-Go> .\localDevEnv.ps1

```

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

```

Administrator: Windows PowerShell

LocalDevEnv

Downloading Github-Helper.psm1 from https://raw.githubusercontent.com/microsoft/AL-Go/a2a51be438318a7e355fa9044815cf7890ffed1a/Actions/Github-Helper.psm1
Downloading AL-Go-Helper.ps1 from https://raw.githubusercontent.com/microsoft/AL-Go/a2a51be438318a7e355fa9044815cf7890ffed1a/Actions/AL-Go-Helper.ps1
Downloading Packages.json from https://raw.githubusercontent.com/microsoft/AL-Go/a2a51be438318a7e355fa9044815cf7890ffed1a/Actions/Packages.json

This script will create a docker based local development environment for your project.

NOTE: You need to have Docker installed, configured and be able to create Business Central containers for this to work.
If this fails, you can setup a cloud based development environment by running cloudDevEnv.ps1

All apps and test apps will be compiled and published to the environment in the development scope.
The script will also modify launch.json to have a Local Sandbox configuration point to your environment.

Applying settings from C:\Users\sosic\Documents\AL Projects\Microsoft\BusinessCentralApps\github\AL-Go-Settings.json
Applying settings from C:\Users\sosic\Documents\AL Projects\Microsoft\BusinessCentralApps\App\Projects\Base Application\AL-Go\settings.json
No settings found in C:\Users\sosic\Documents\AL Projects\Microsoft\BusinessCentralApps\github\localDevEnv.settings.json
Applying settings from C:\Users\sosic\Documents\AL Projects\Microsoft\BusinessCentralApps\App\Projects\Base Application\AL-Go\localDevEnv.settings.json
No settings found in C:\Users\sosic\Documents\AL Projects\Microsoft\BusinessCentralApps\App\Projects\Base Application\AL-Go\sosic.settings.json
Checking System Requirements

Container name
-----
Please enter the name of the container to create (default bcserver)

```

I keep it personally by default as “bcserver”

```

Container name
-----
Please enter the name of the container to create (default bcserver)
bcserver selected
Authentication mechanism for container
-----
a Windows Authentication
b Username/Password authentication
Select authentication mechanism for container (default b)

```

Next Authentication:

```

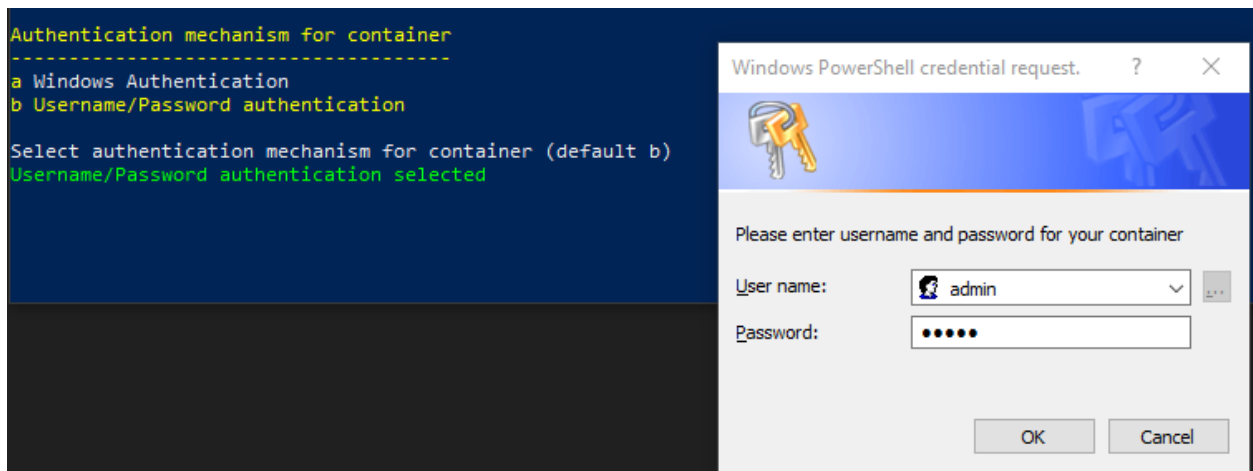
Authentication mechanism for container
-----
a Windows Authentication
b Username/Password authentication
Select authentication mechanism for container (default b)

```

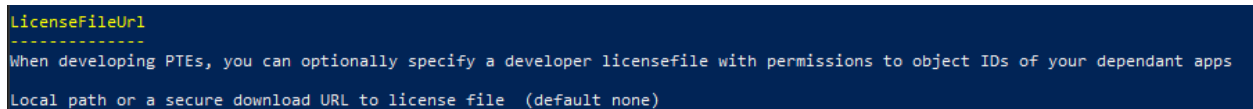


Choose the one that you got used to, we will stick to the default (User/Pass).

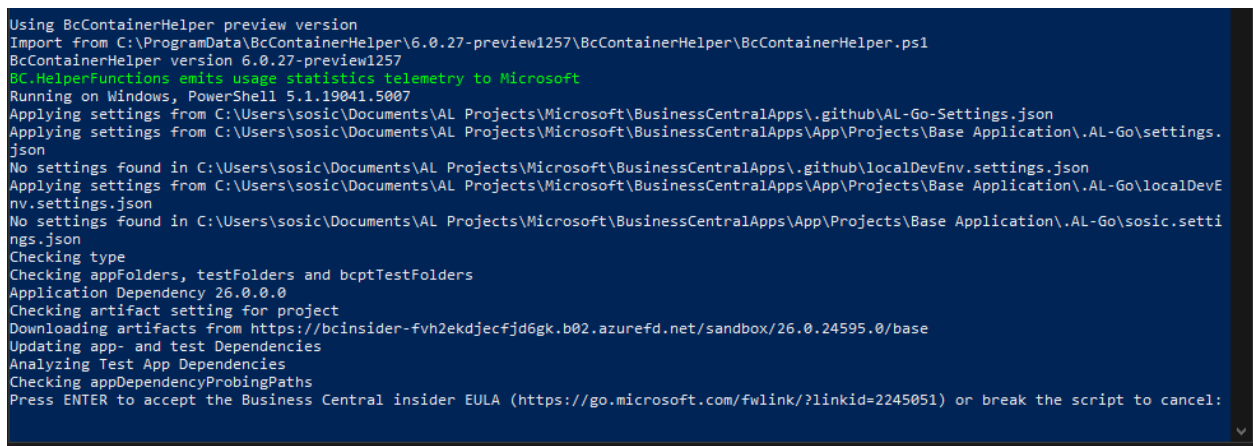
When selected, a credential prompt will show up where you can input the Password of your choice:



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:



```

Administrator: Windows PowerShell

$script = Join-Path $PSScriptRoot "../../../build/scripts/NewBcContainer.ps1" -Resolve
$script -parameters $parameters
Import-TestToolkitToBcContainer override
Diagnostics.CodeAnalysis.SuppressMessageAttribute('PSReviewUnusedParameter', 'parameters', Justification = 'The parameter is not used, but it's script needs to match this format')
Param(
    [hashtable] $parameters
)

$script = Join-Path $PSScriptRoot "../../../build/scripts/ImportTestToolkitToBcContainer.ps1" -Resolve
$script -parameters $parameters
Custom pre-compilation script defined.
Param(
    [string] $appType,
    [ref] $compilationParams
)

$scriptPath = Join-Path $PSScriptRoot "../../../build/scripts/PreCompileApp.ps1" -Resolve
$scriptPath -parameters $compilationParams -appType $appType
PublishBcContainerApp override
Param([Hashtable]$parameters)

$script = Join-Path $PSScriptRoot "../../../build/scripts/PublishBcContainerApp.ps1" -Resolve
$script -parameters $parameters
ImportTestDataInBcContainer override
Param(
    [Hashtable]$parameters
)

$script = Join-Path $PSScriptRoot "../../../build/scripts/ImportTestDataInBcContainer.ps1" -Resolve
$script -parameters $parameters
RunTestsInBcContainer override
Param([Hashtable]$parameters)

$script = Join-Path $PSScriptRoot "../../../build/scripts/RunTestsInBcContainer.ps1" -Resolve
$script -parameters $parameters
Pulling generic image

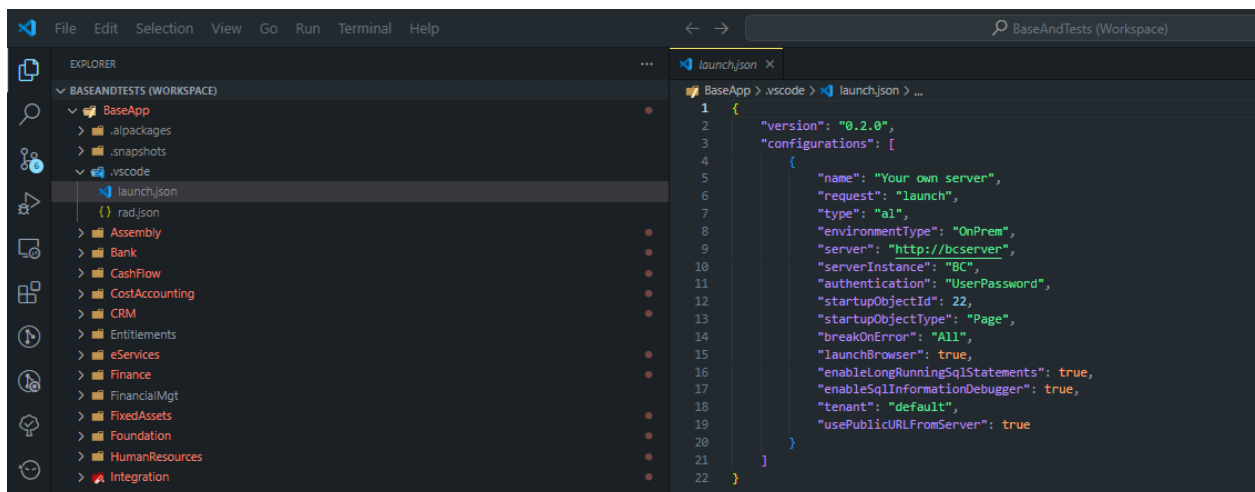
Pulling generic image
pulling mcr.microsoft.com/businesscentral:ltsc2019-dev
  
```

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:

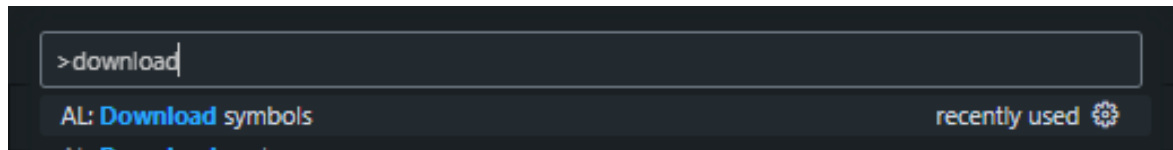


```

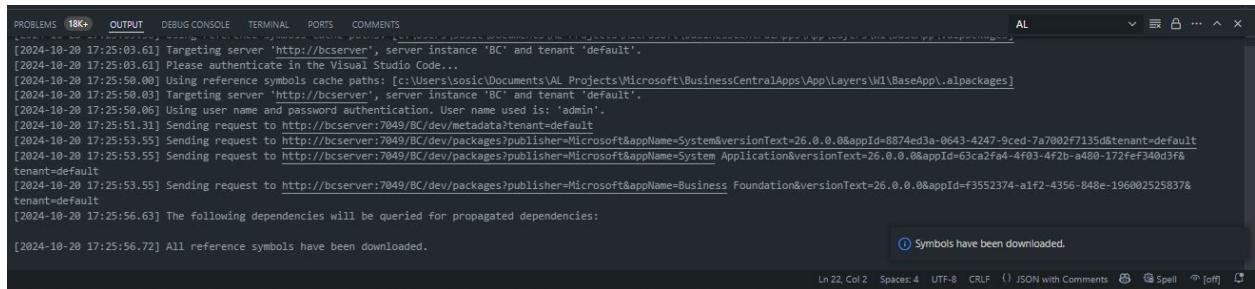
{
  "version": "0.2.0",
  "configurations": [
    {
      "name": "Your own server",
      "request": "launch",
      "type": "al",
      "environmentType": "OnPrem",
      "server": "http://bcserver",
      "serverInstance": "BC",
      "authentication": "UserPassword",
      "startupObjectId": 22,
      "startupObjectType": "Page",
      "breakOnError": "All",
      "launchBrowser": true,
      "enableLongRunningSqlStatements": true,
      "enableSqlInformationDebugger": true,
      "tenant": "default",
      "usePublicURLFromServer": true
    }
  ]
}
  
```

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

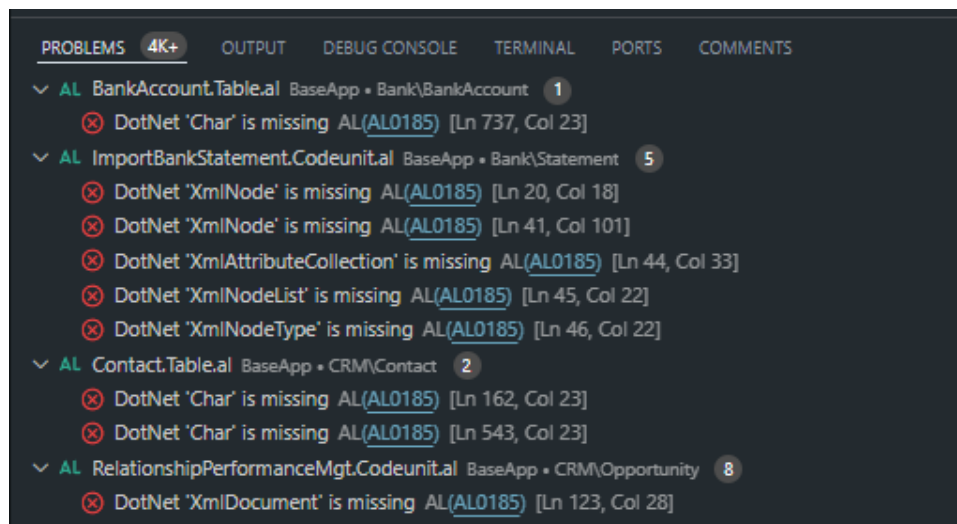
Execute the “Download symbols” command:



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

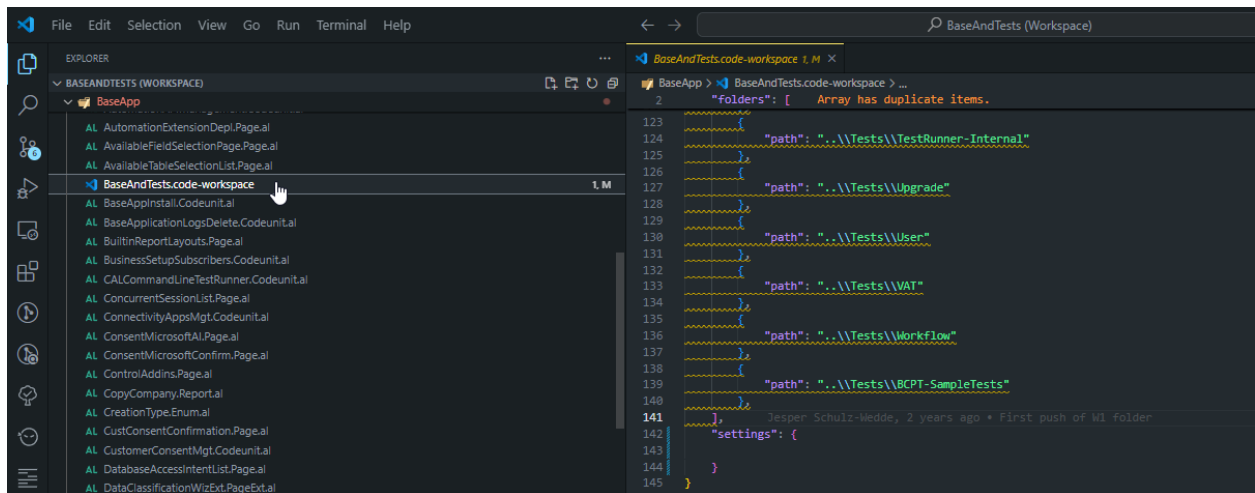
In order to resolve issues related to missing DotNet files you will need to define the assembly probing path.

#### 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.

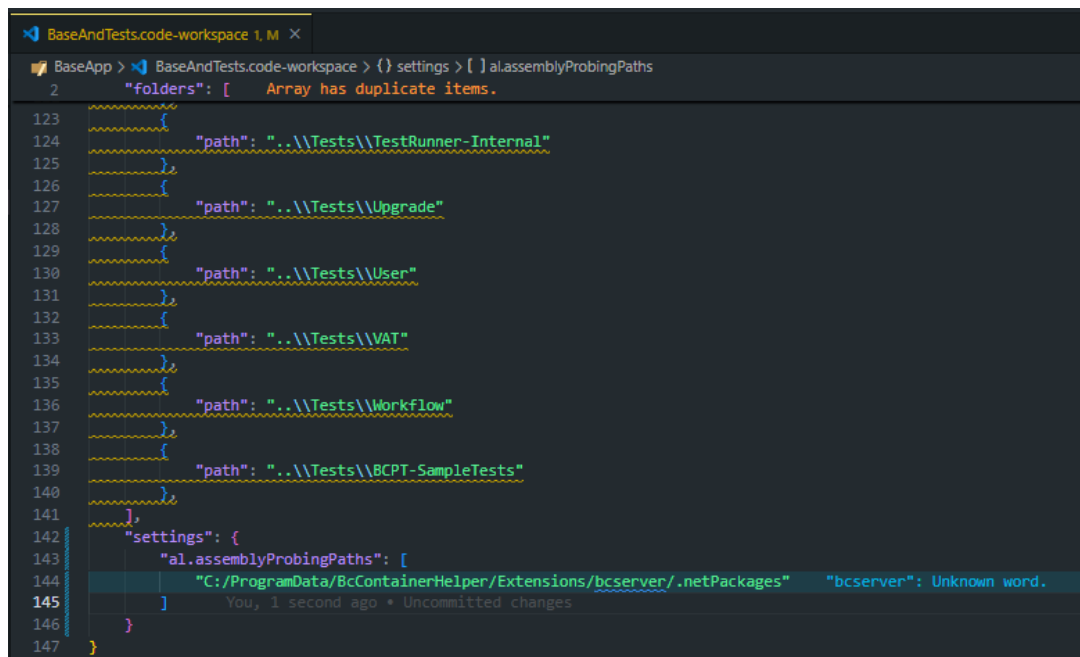
Inside “settings” part, you will need to define the following:

```

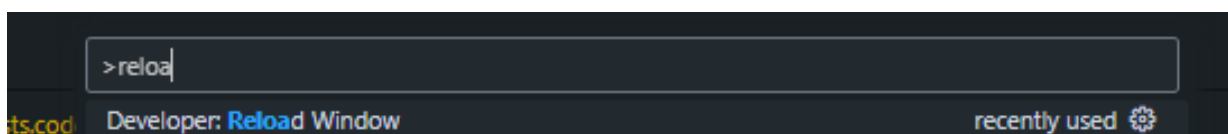
"al.assemblyProbingPaths": [
    "C:/ProgramData/BcContainerHelper/Extensions/bcserver/.netPackages"
]

```

It will look like:

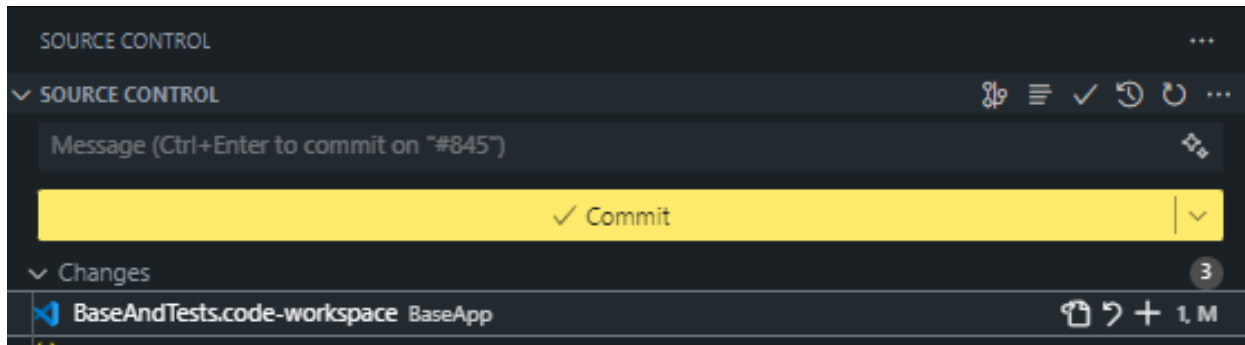


Wait a bit and errors will be gone. If not, restart your VS Code:



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

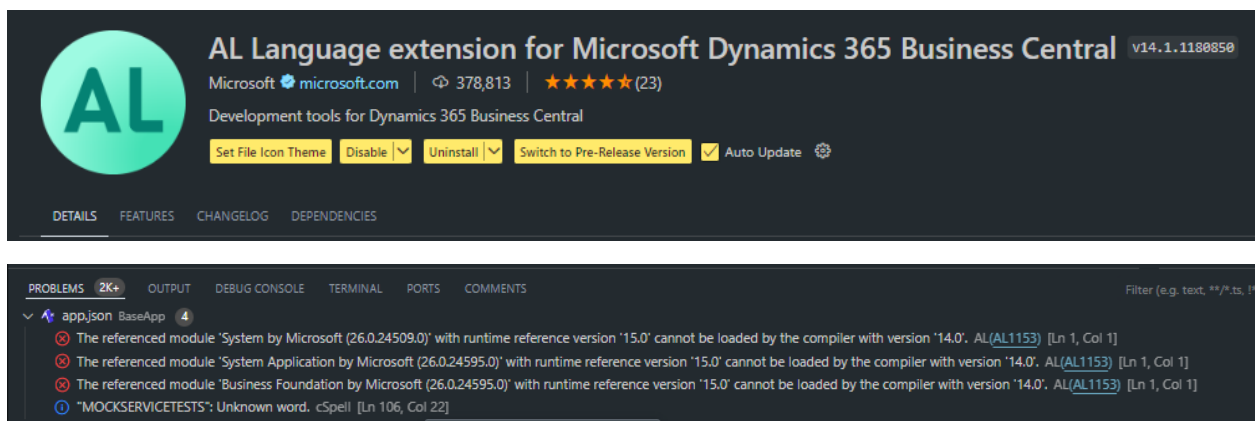
When you did this, this file will show up in Source Control:



**Make sure you don't commit this file!**

### 3.4. Use the correct AL Language extension for VS Code

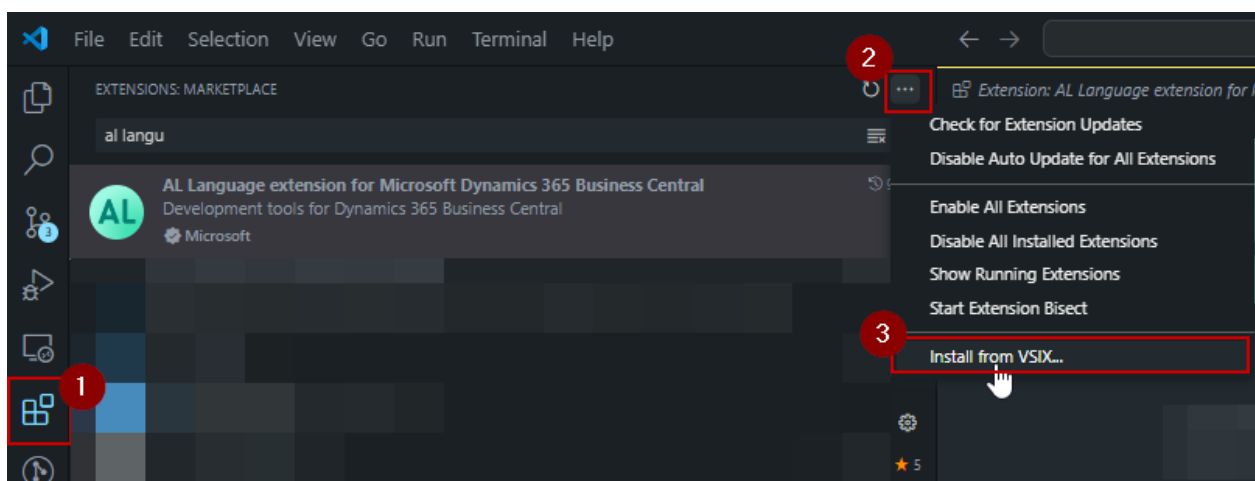
When working with the insider version you can't use the Release version of AL Language:



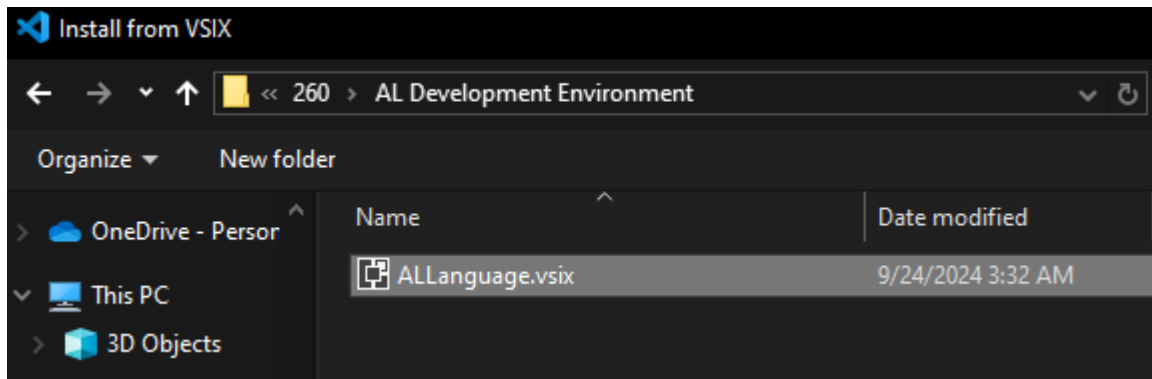
And if not just runtime version error, you will get more errors.

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:



*(this guide is Pre-Release version)*



### Where is my VSIX file?

Example path is:

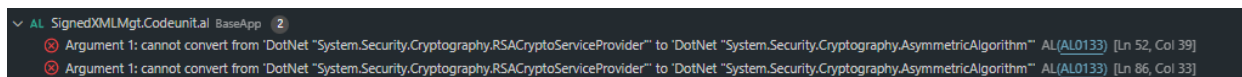
C:\bcartifacts.cache\sandbox\26.0.24595.0\platform\ModernDev\program files\Microsoft Dynamics NAV\260\AL Development Environment

### 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 then 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:

```
StefanSosic:#962
```

```
TKapitan:Issue#965
```

```
pri-kise:feature/861-ext-texts-job-fixes
```

## 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.

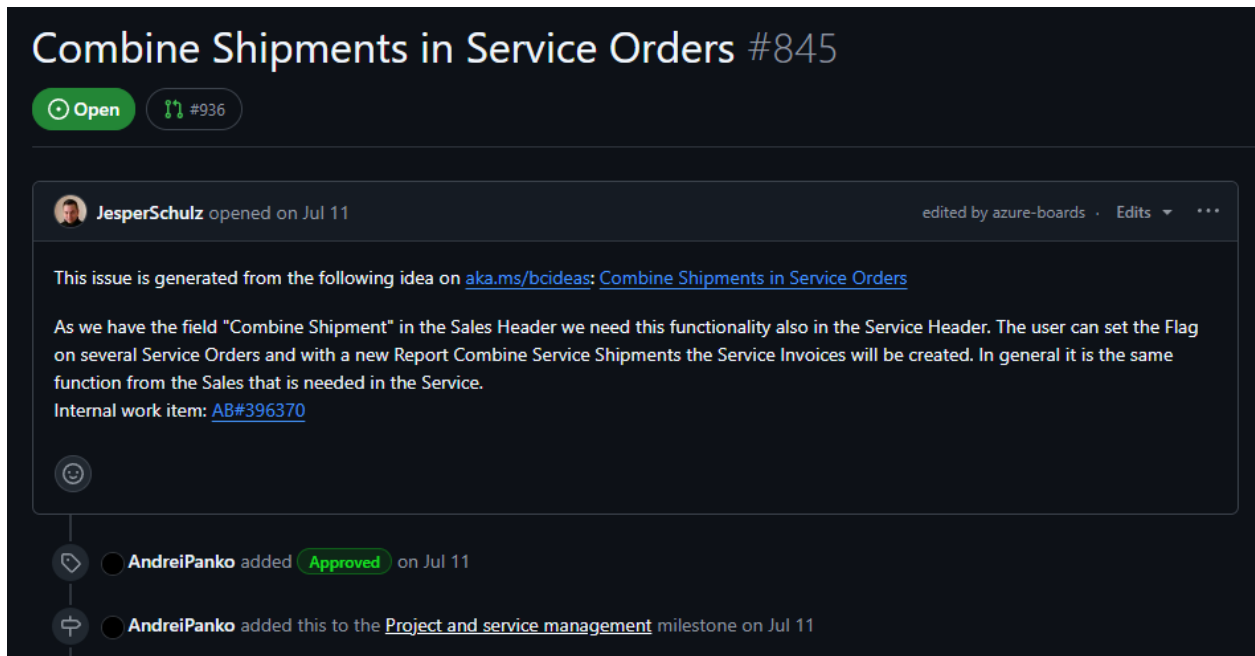
Describe a new module inside the README.md file.

And of course, create tests inside your new module.

## Creating your first Pull Request

### Take the issue on GitHub

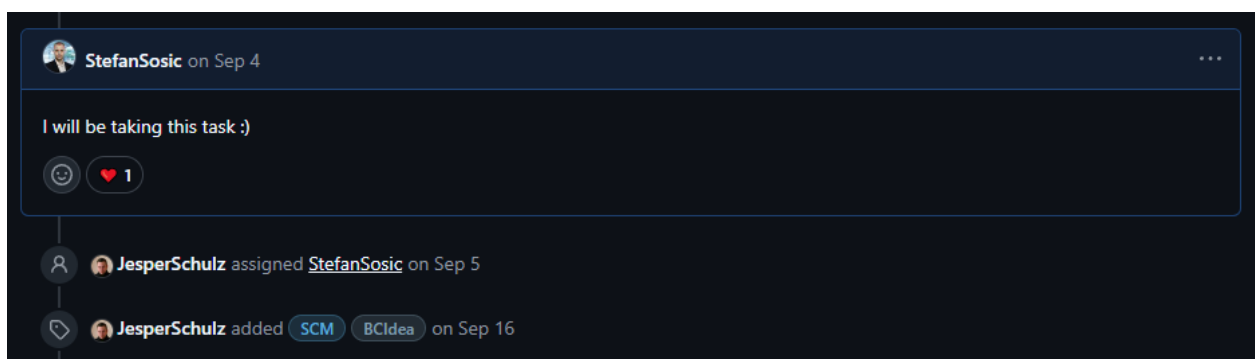
Find the issue which you find interesting and would like to provide a solution for it.



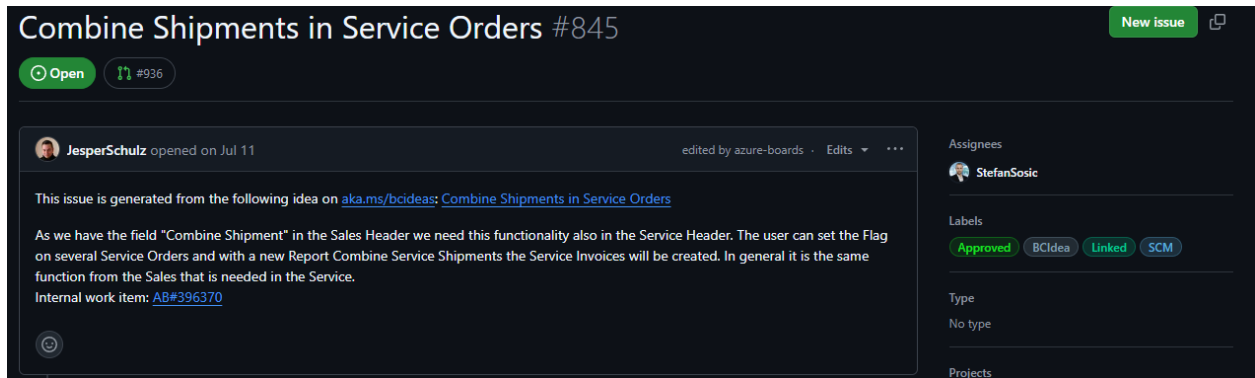
Make sure it has an “Approved” tag on it.

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:

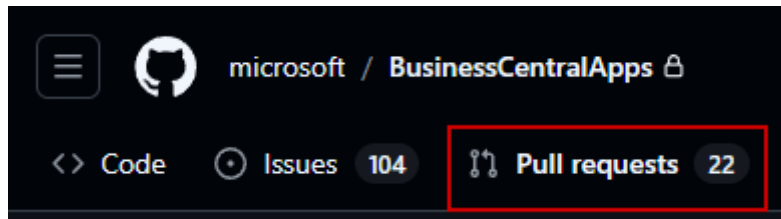
- Variables are sorted
- Usings are sorted
- You don’t leave unused variables
- Check if you are making breaking changes (if you are doing breaking changes, make sure you put obsolete property and CLEAN tags)
- Check spelling in your Labels/Texts/Procedures/Descriptions
- Casing of methods, variables, 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.

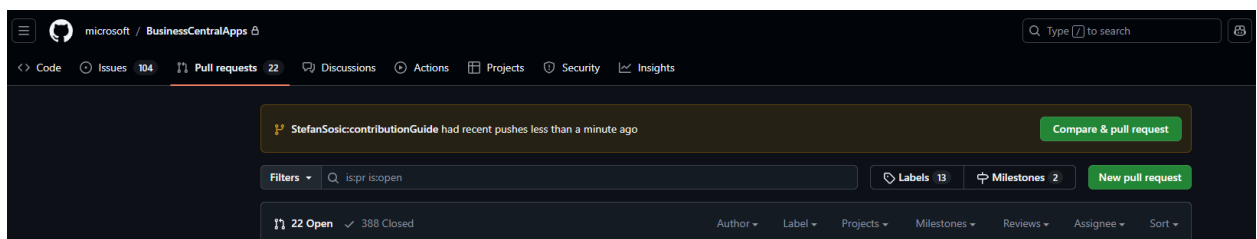
## Create Pull Request

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

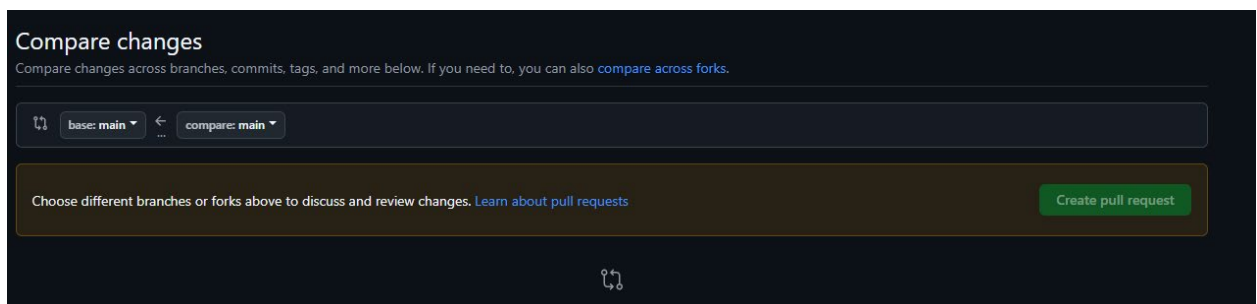
Open the GitHub Microsoft repository and go to the “Pull Requests” tab:



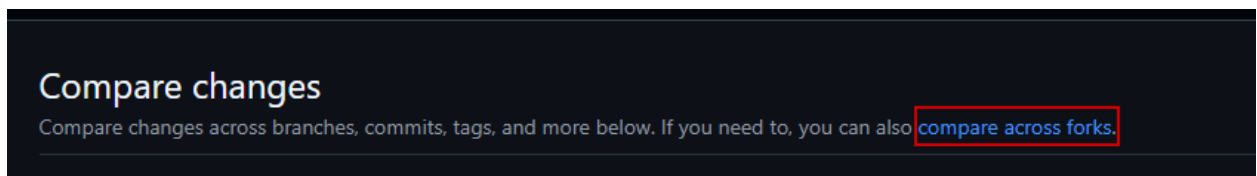
If you have done all right you will see a screen like this:



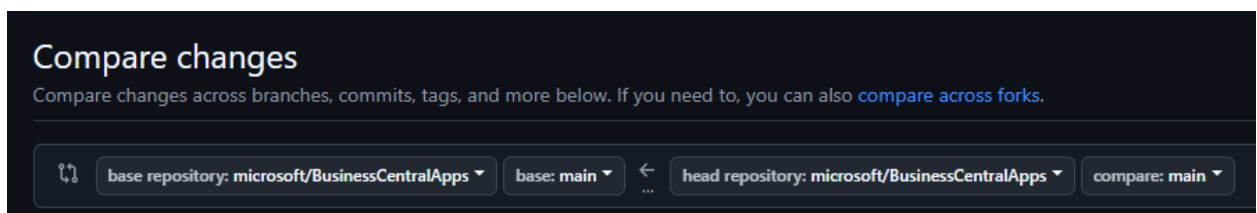
If you want to create a Pull Request for some issue that you resolved earlier but haven't created a Pull Request yet, you can go to “New pull request”:



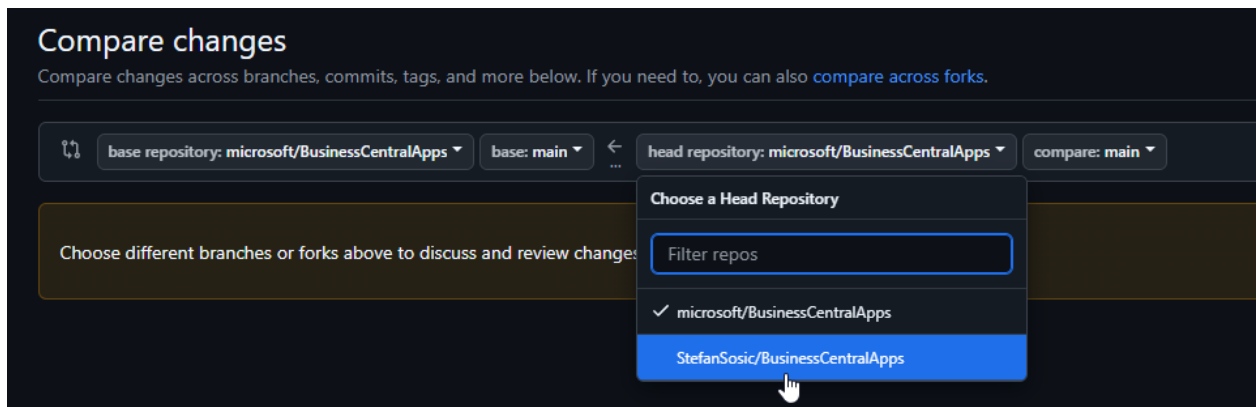
If you don't see your branch, no worries. You will need to select “compare across forks”.



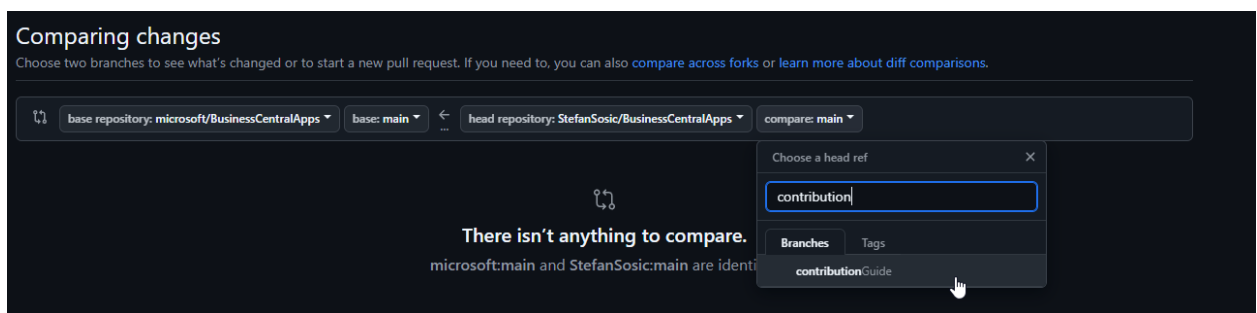
Different compare appear:



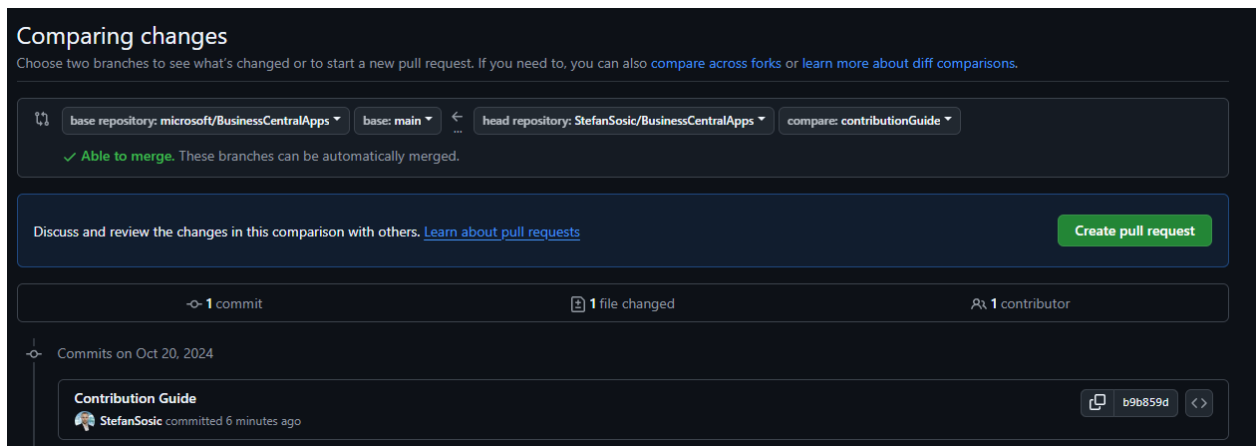
On the head repository drill down and select your fork:



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”.

**Open a pull request**  
Create a new pull request by comparing changes across two branches. If you need to, you can also [compare across forks](#). [Learn more about diff comparisons here](#).

base repository: microsoft/BusinessCentralApps base: main head repository: StefanSotic/BusinessCentralApps compare: contributionGuide

✓ Able to merge. These branches can be automatically merged.

**Add a title**  
#123 Contribution Guide

**Add a description**

Write Preview

<!-- Thank you for submitting a Pull Request. If you're new to contributing to BusinessCentralApps please read our pull request guideline below  
\* <https://github.com/microsoft/BusinessCentralApps/blob/main/CONTRIBUTING.md>  
-->  
#### Summary <!-- Provide a general summary of your changes -->  
  
#### Work Item(s) <!-- Add the issue number here after the #. The issue needs to be open and approved. Submitting PRs with no linked issues or unapproved issues is highly discouraged. -->  
Fixes #

Markdown is supported Paste, drop, or click to add files

Allow edits and access to secrets by maintainers **Create pull request**

Remember, contributions to this repository should follow its [contributing guidelines](#), [security policy](#), and [code of conduct](#).

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:

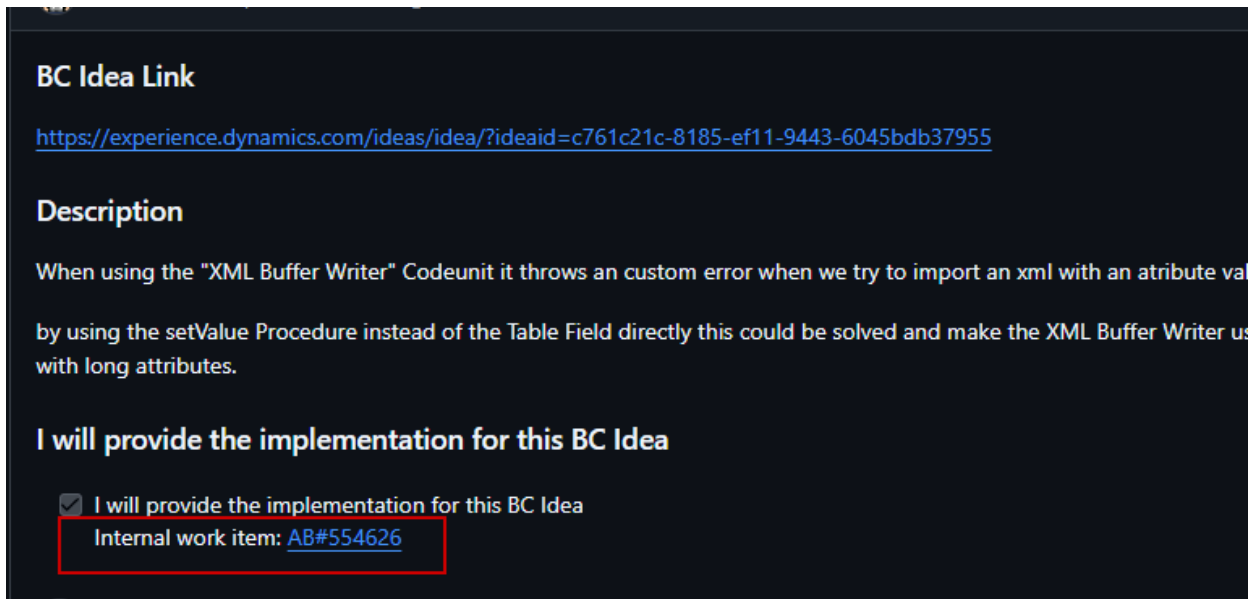
#### Work Item(s) <!-- Add the issue number here after the #. The issue needs to be linked issues or unapproved issues is highly discouraged. -->  
Fixes #123

#123 Test coverage for physical inventory calculation

Markdown is supported Paste, drop, or click to add files

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:



**BC Idea Link**

<https://experience.dynamics.com/ideas/idea/?ideaid=c761c21c-8185-ef11-9443-6045bdb37955>

**Description**

When using the "XML Buffer Writer" Codeunit it throws an custom error when we try to import an xml with an attribute value with long attributes.

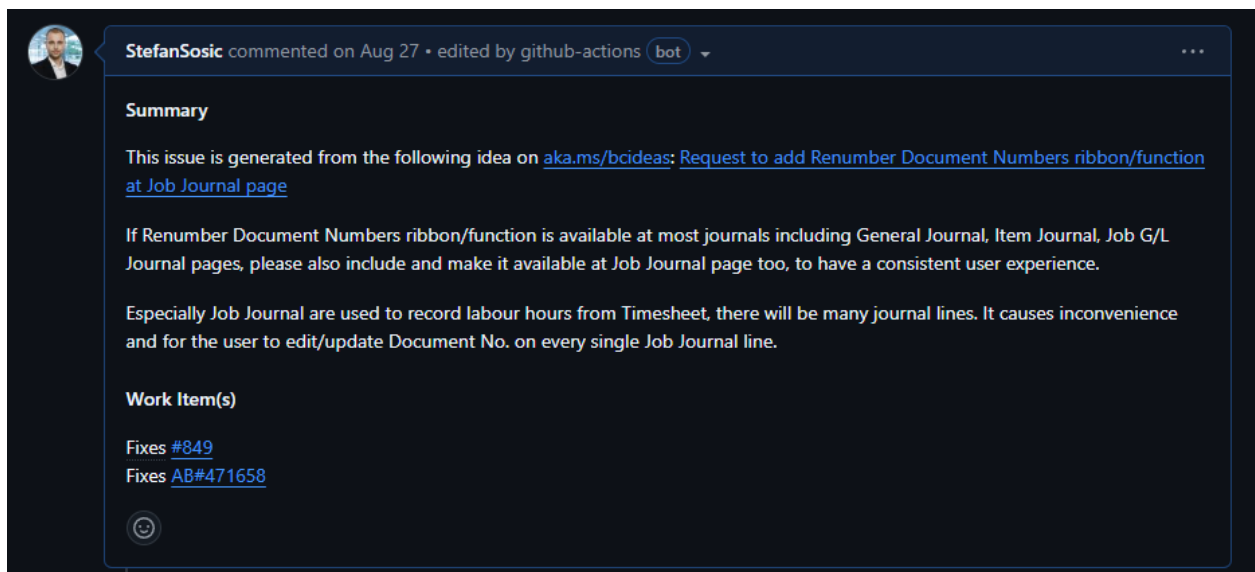
by using the setValue Procedure instead of the Table Field directly this could be solved and make the XML Buffer Writer usable with long attributes.

**I will provide the implementation for this BC Idea**

I will provide the implementation for this BC Idea

Internal work item: [AB#554626](#)

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



StefanSusic commented on Aug 27 • edited by github-actions (bot) ▾

**Summary**

This issue is generated from the following idea on [aka.ms/bcideas](https://aka.ms/bcideas): [Request to add Renumber Document Numbers ribbon/function at Job Journal page](#)

If Renumber Document Numbers ribbon/function is available at most journals including General Journal, Item Journal, Job G/L Journal pages, please also include and make it available at Job Journal page too, to have a consistent user experience.

Especially Job Journal are used to record labour hours from Timesheet, there will be many journal lines. It causes inconvenience and for the user to edit/update Document No. on every single Job Journal line.

**Work Item(s)**

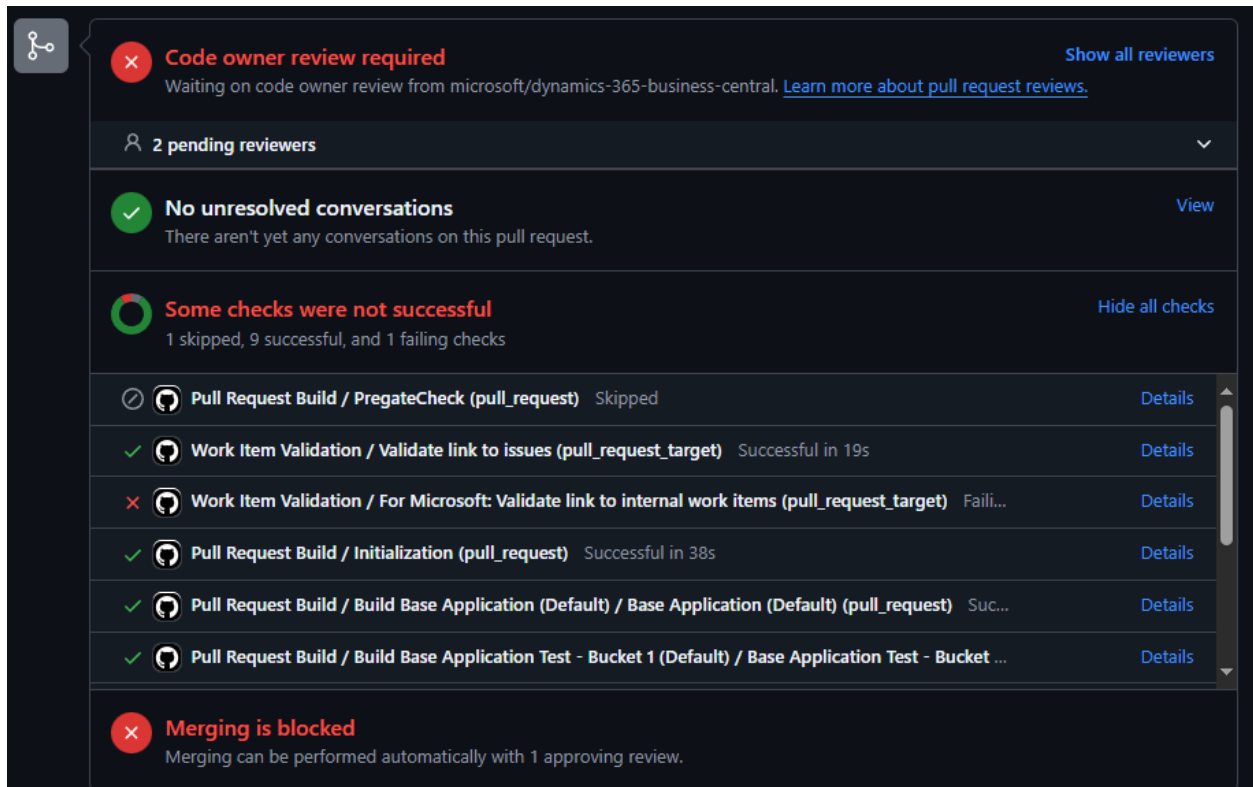
Fixes [#849](#)

Fixes [AB#471658](#)

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.

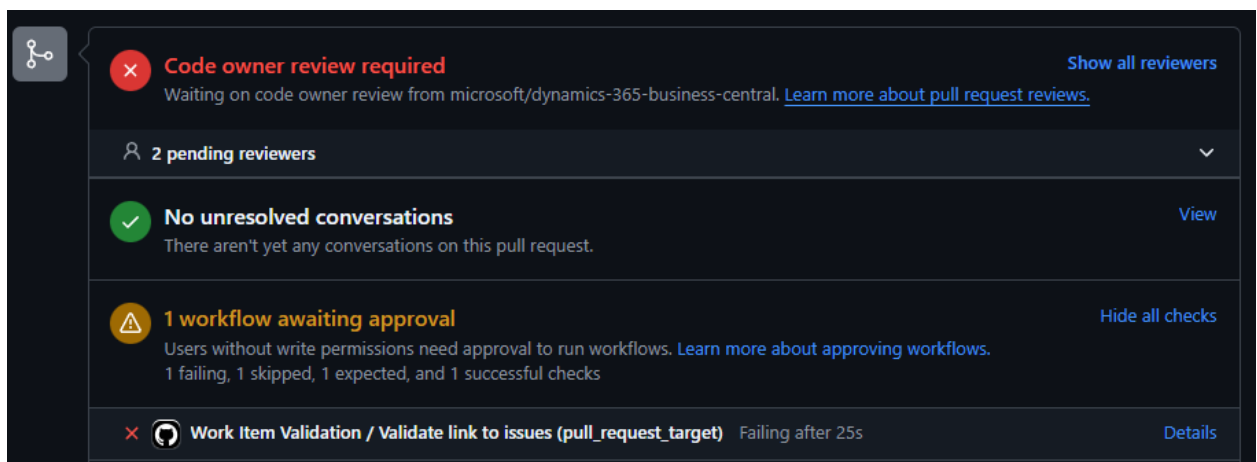
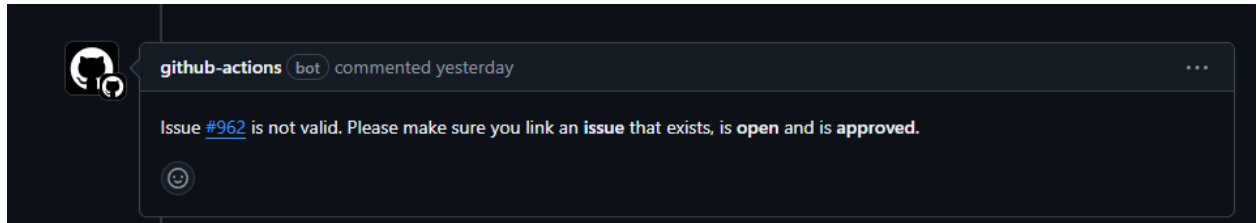
**@microsoft-github-policy-service agree company="Microsoft"**

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.

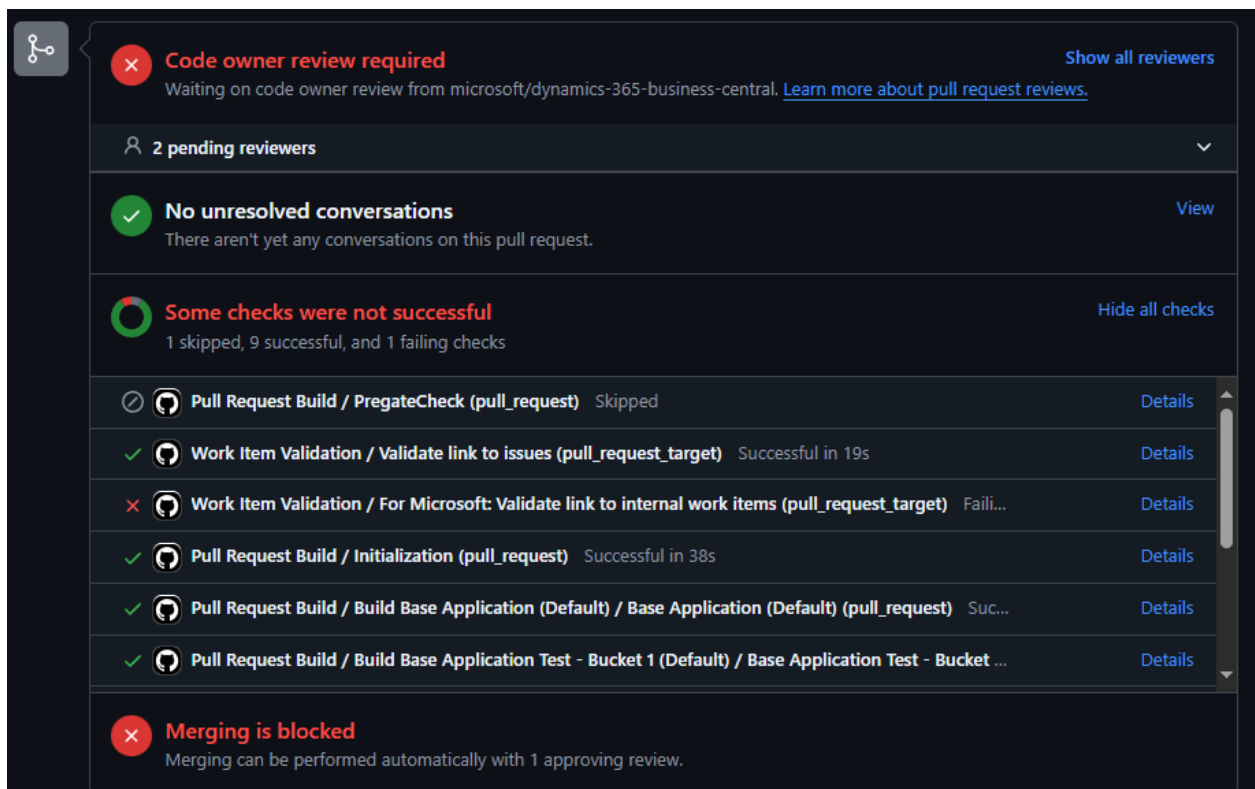
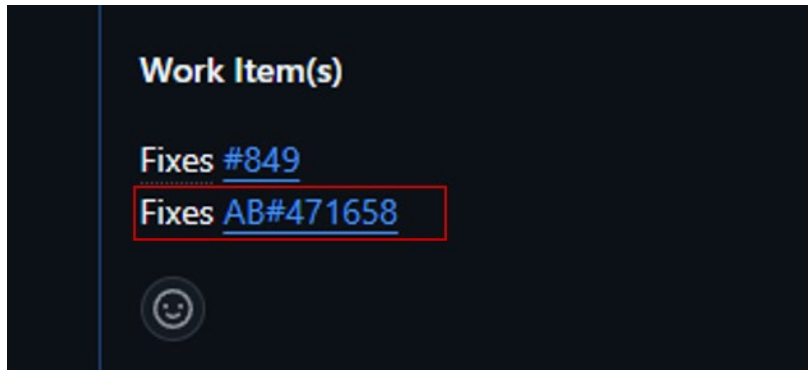
## Pull Request CIs

### 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:

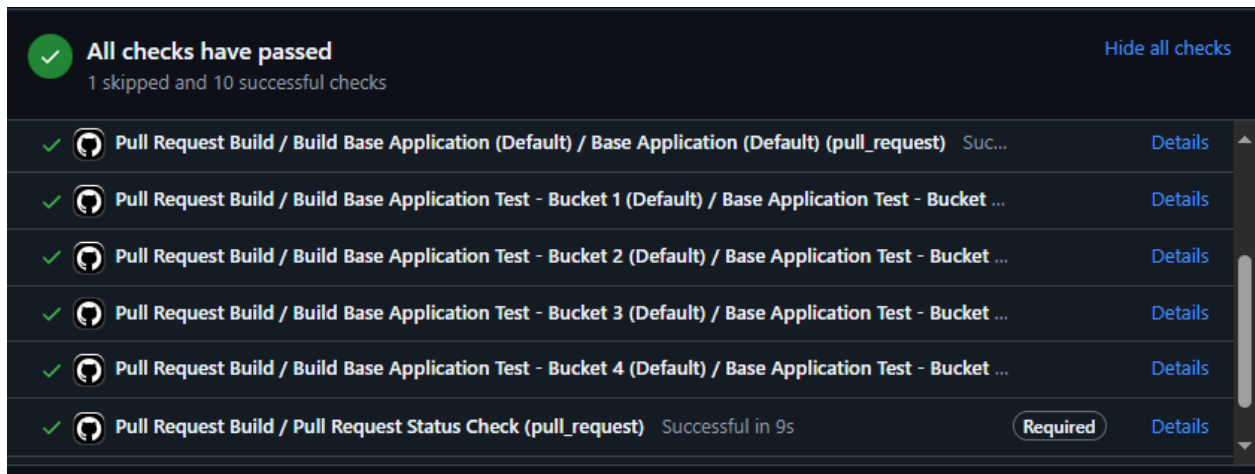


## Missing internal work item



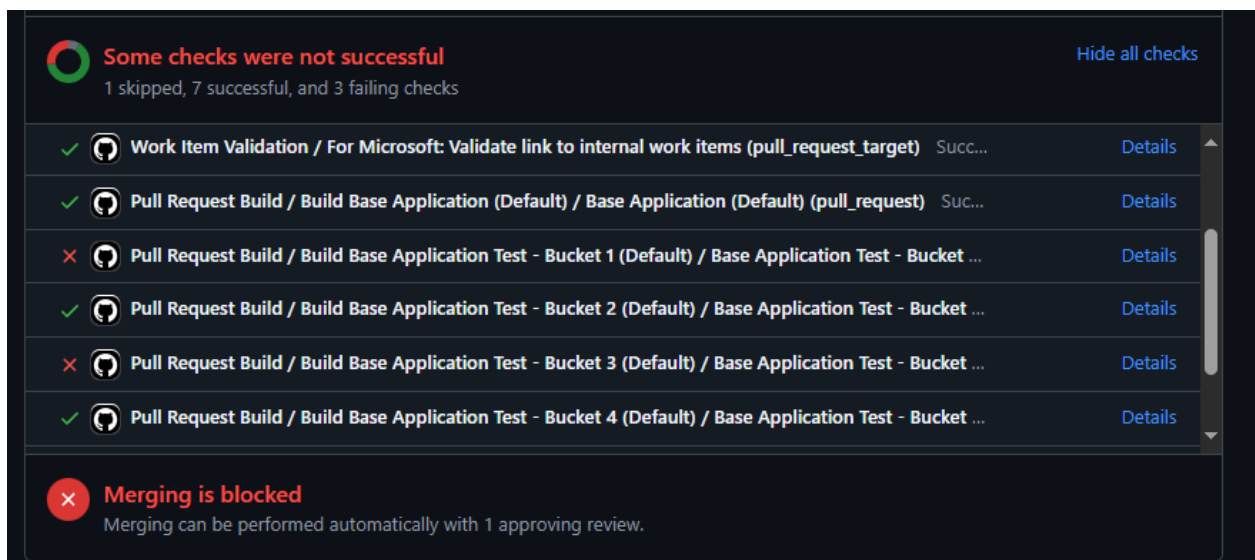
## Build validation and tests

How it should be:



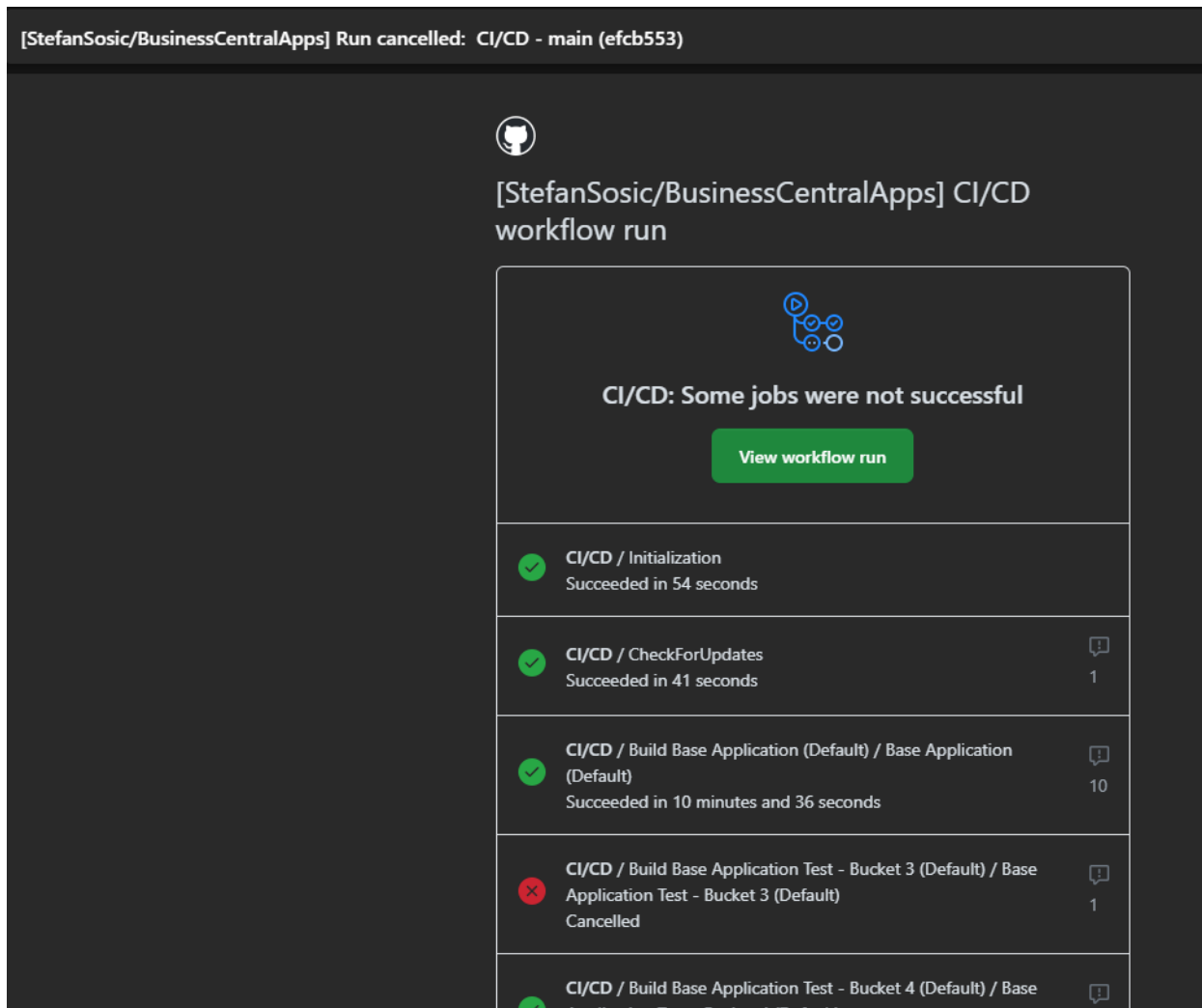
If you have any compilation errors or tests fail.

This is how it should not be, of course:

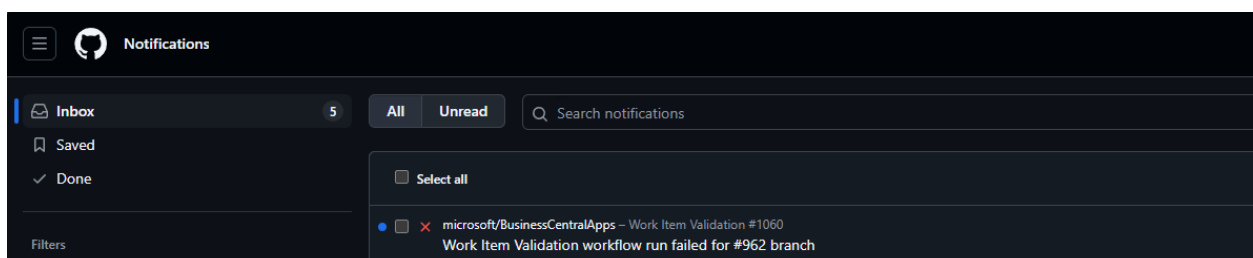


## CI failed notifications

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

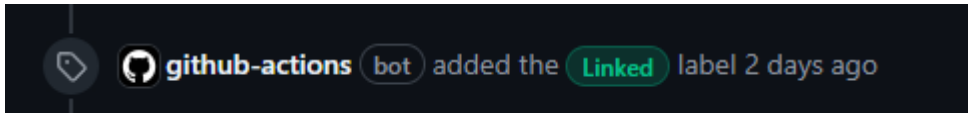


You will be informed by Notification (if enabled on your GitHub settings):



## Pull Request Tags

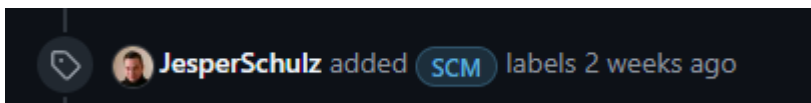
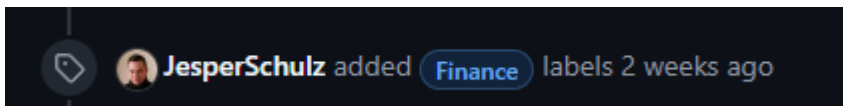
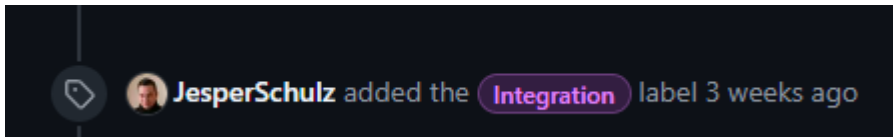
### Linked



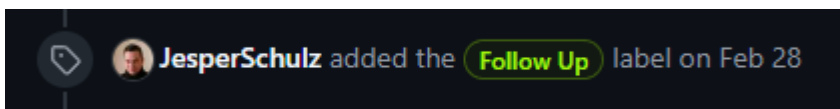
Automatically will be added if you have added an issue that is already approved also an internal Microsoft work item.

### Integration/SCM/Finance and so on...

To which area your implementation does belong.

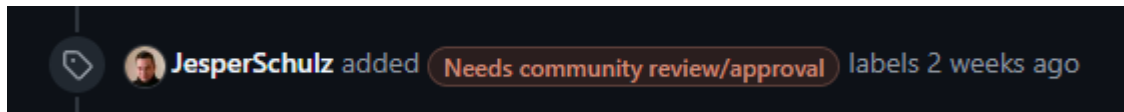


### Follow Up



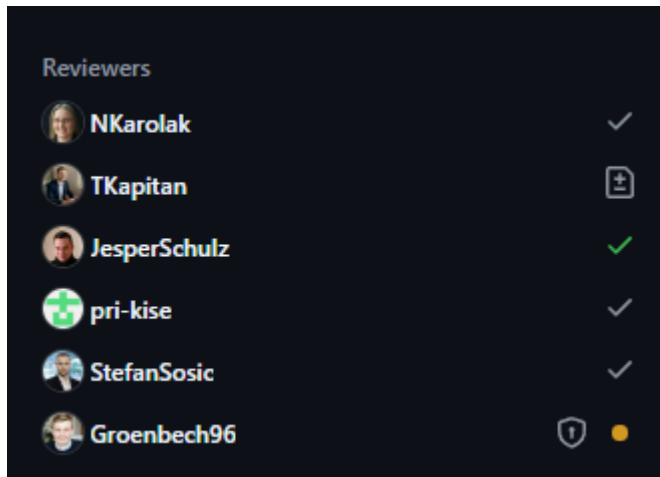
Due to complexity, there is active discussion either with the Pull Request creator or with other internal employees of Microsoft between departments. In order to get additional information and give input to the pull request creator.

### Needs community review/approval



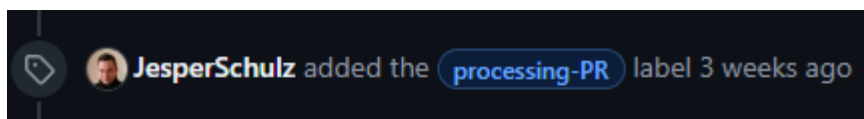
Other community members are required to review your pull request.

A standard Pull Request approval looks like this:

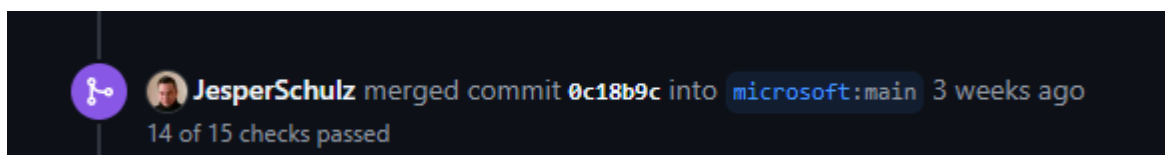


As you can see, a few community members and at least one Microsoft employee.

### 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.

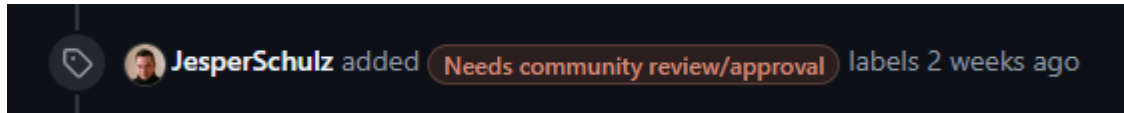


## Other ways for contributions - Review of pull requests

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.*