

Directions ASIA 2023

The Power of Open-Source D365 (not only) base app



Tomas Kapitan (Fusion5)

Fusion5 AU/NZ

 @KeptyCZ


 @tomaskapitan

<https://www.kepty.cz>

Jesper Schulz-Wedde

Microsoft

 @JesperSchulz

 @jesper-schulz-wedde

#makingpotentialreality



Agenda

- The contribution process:
 - Why Open Source?
 - Get your contribution suggestion approved
 - Get your development environment setup
 - Create your PR and get it to the finish line
- Hands on experiences from a partner's perspective
 - Code review process - it's give and take!
 - A developer's notes
- What's next? Which changes are to come?
- Q&A

Why Open Source?

Why have a contribution model?



Accelerate Product Growth



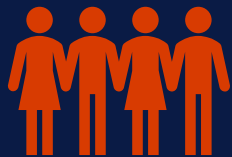
Improve Product Quality & Auditability



Enhance Extensibility, Customizability & Compatibility



Foster Collaboration

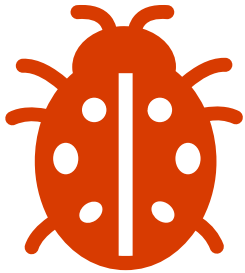


Develop Community

The contribution process

Phase I: Get your idea approved for contribution

How does Microsoft work internally?



Customer reported product defects (support cases, repair items and hotfix requests)

Procedural issues (e.g., blocking work processes)

Data issues (e.g., generating faulty data)

Partner reported product defects

Papercuts (typically fixed by themselves in a code customization model)

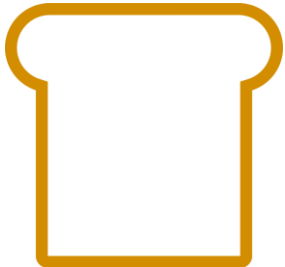
Design change requests (added to prioritized backlog)

Collaboration requests (helping partner resolve an issue on their end)

Extensibility requests and issues (currently processed by separate team)

Internally reported product defects

Come in all shapes and sizes



Part of internal epics (e.g., fundamentals, onboarding, manufacturing)

Large stories which improve the product all up

Partner reported ideas (aka.ms/bcideas)

Ideas of various nature and various sizes

!!! EXIT CRITERIA !!!

How do things look on GitHub?

One work item to rule them all: issues!

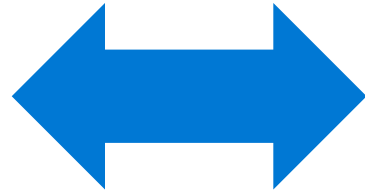
“GitHub’s issue tracking is unique because of our focus on simplicity, references, and elegant formatting.”

<https://github.com/features/issues>

Challenge

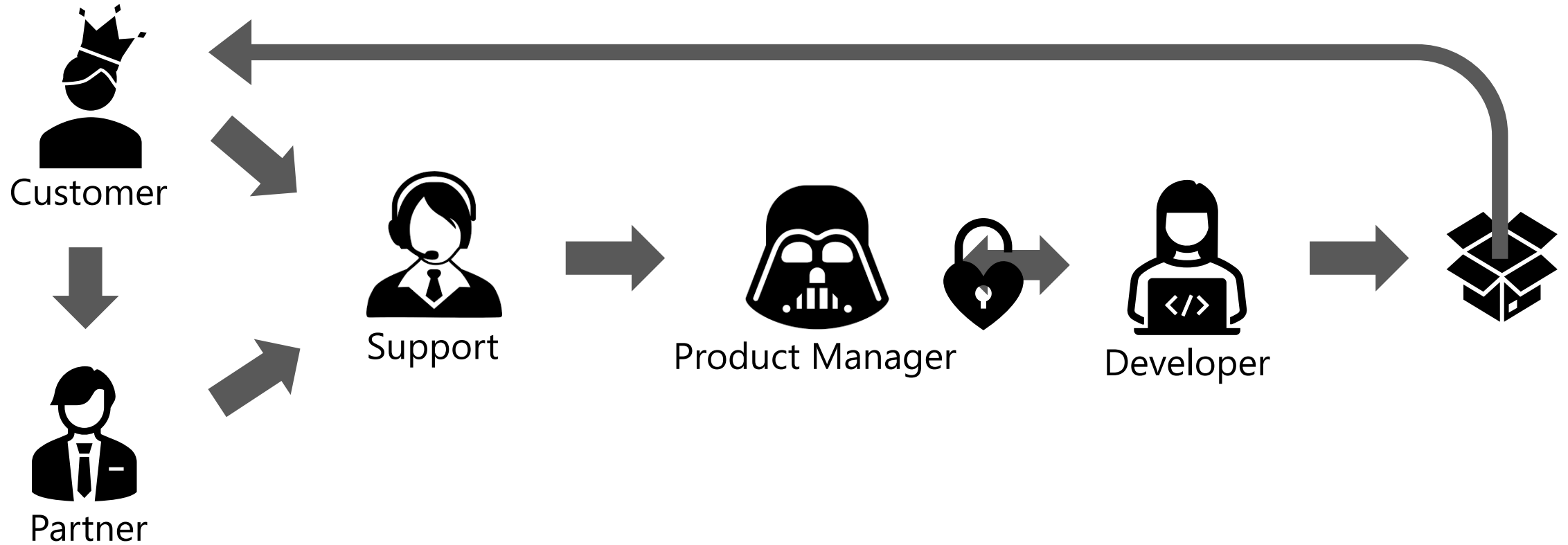


GitHub

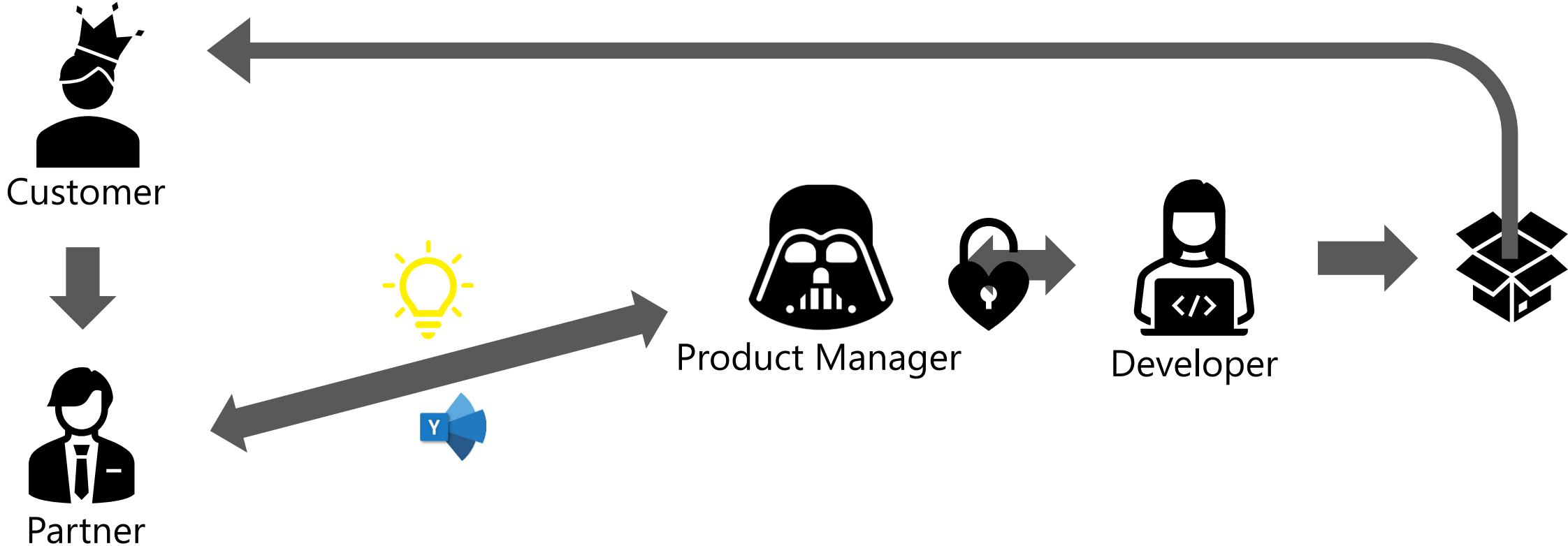


Azure **DevOps**

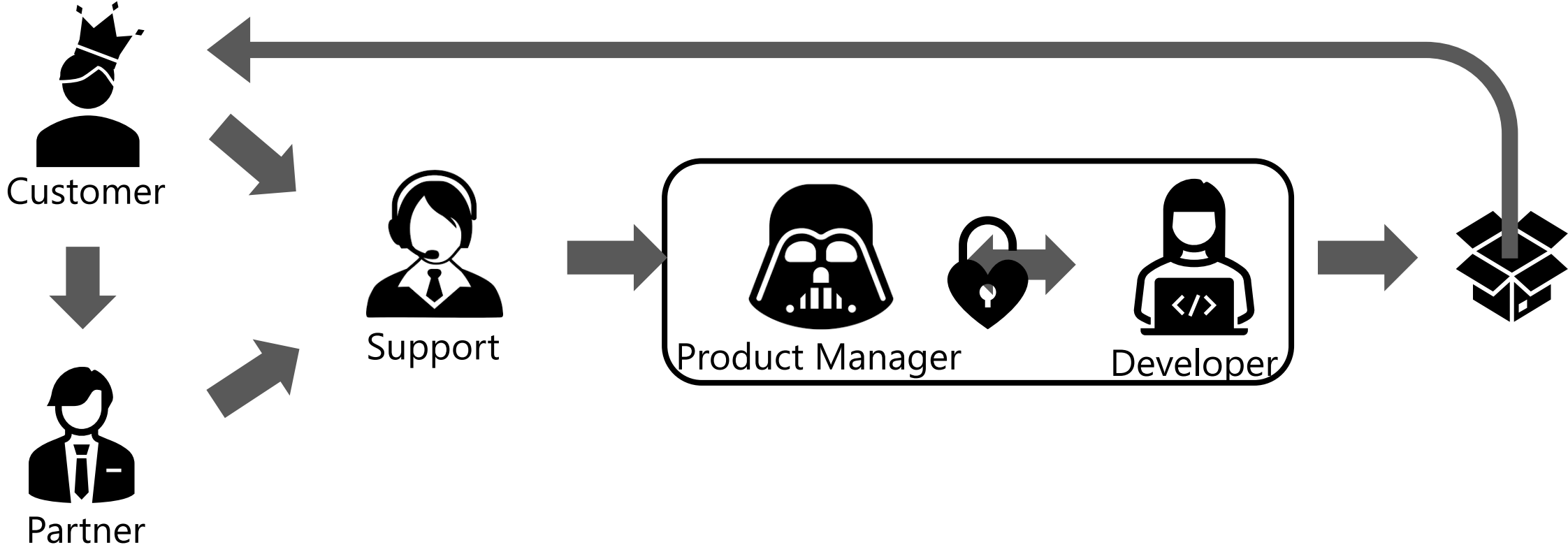
The normal flow of work



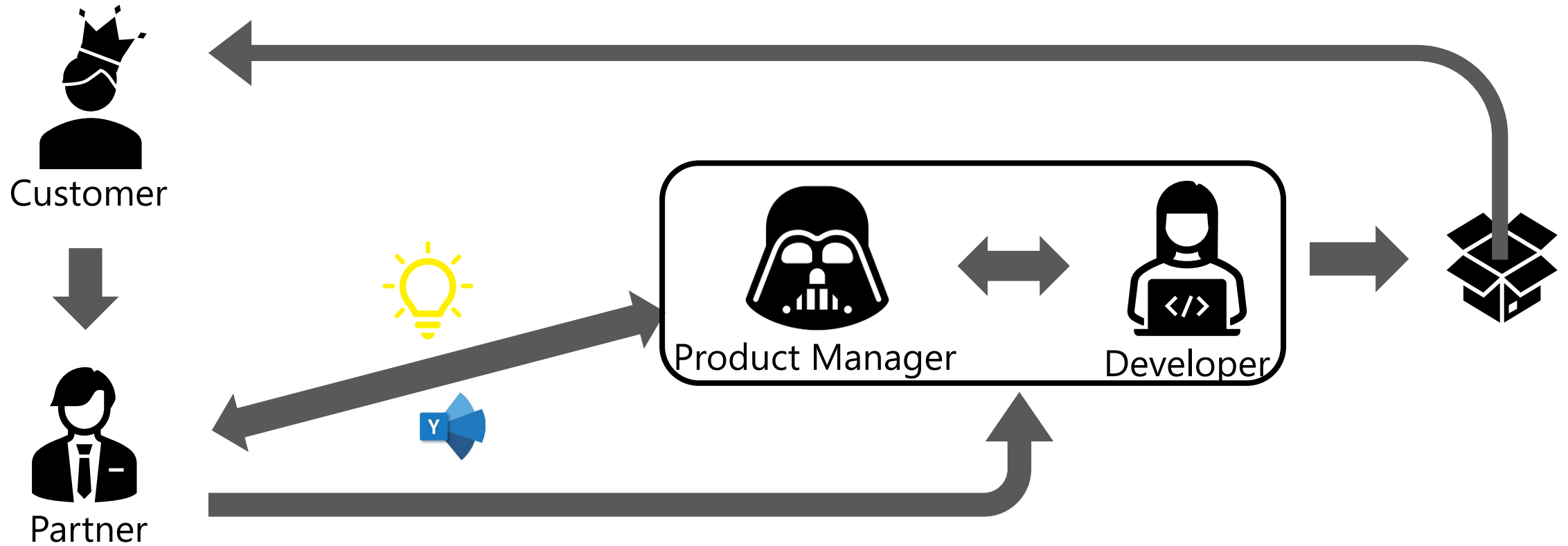
The normal flow of work



How does GitHub fit into this?



How does GitHub fit into this?



What will the Product Managers approve?

Always

- Papercuts
 - Missing fields on pages
 - Missing fields in data sets
 - Typos or missing tooltips
 - Missing actions found elsewhere
 - Alignment of experience
 - Improvements to tests
- New horizontal building blocks
- Bug-fixes without direct impact on production.

Sometimes

- Changes to existing business logic
- Very large and complex contributions

Never

- Product Defects with customer impact
- Work on localizations
- Currently: Event Requests

Conclusion



Small issues correspond to bugs (internally in Microsoft). They can get created as issues on GitHub and will very likely get approved by the Microsoft product group.



Larger issues correspond to slices. They need to start their life as BCIdeas and need to get approved by a Microsoft product manager to fit the overall development strategy.



While Microsoft is working on moving the internal development of all apps (incl. the Base Application) out on GitHub, some “bridging the gap” is needed. Defining that bridge was/is the goal of the pilot.

Create your GitHub issue

And get it "ready for implementation"

Demo 1

<https://kepty.cz/OS-Bangkog-Demo1.mp4>

(demos are created by Jesper, Microsoft)

The contribution process

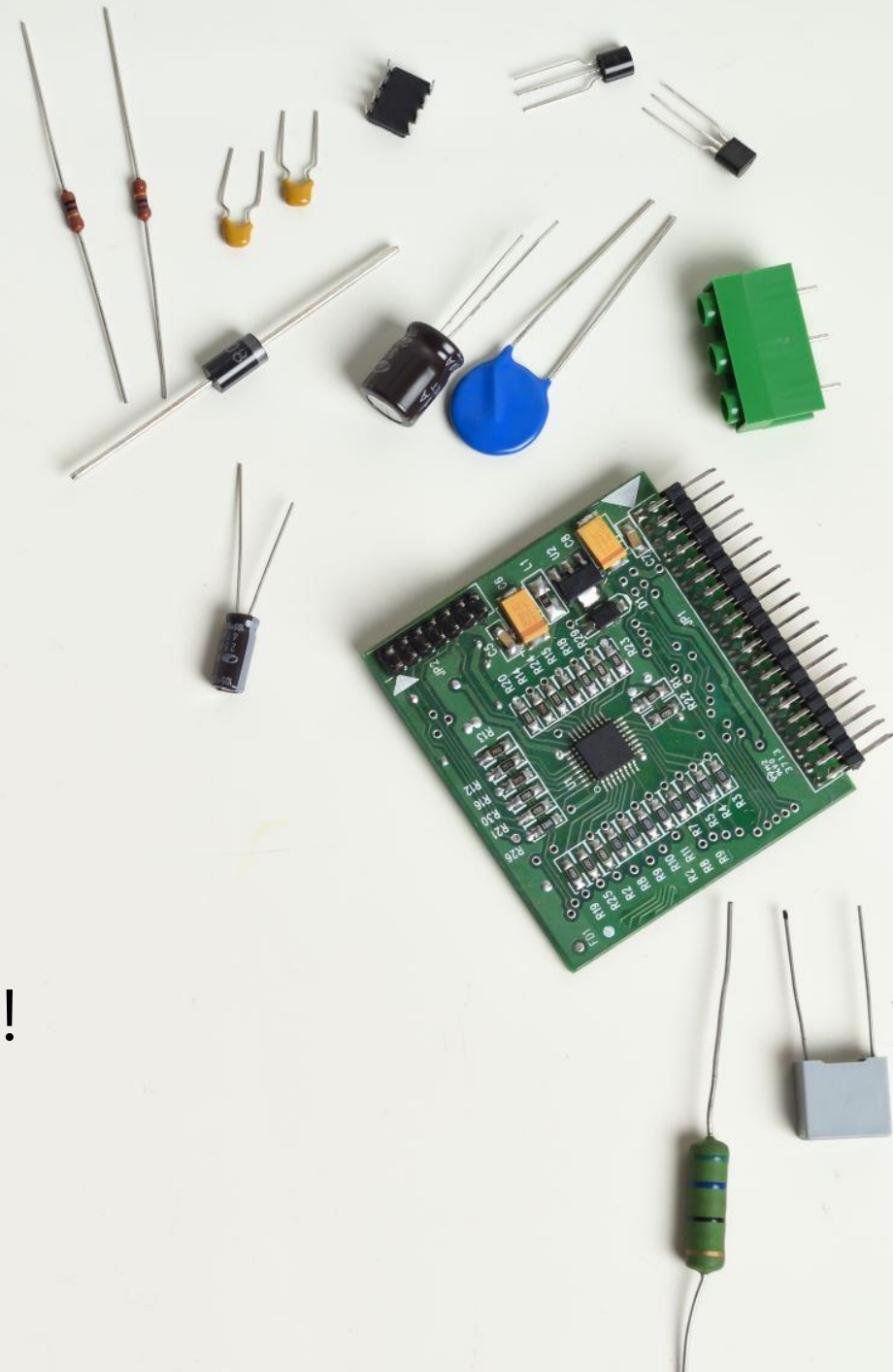
Phase II: Get your development environment set up

It's easy – really...

Once you know how!

So, let's take 10 minutes to walk you through the setup process of a Business Central development machine.

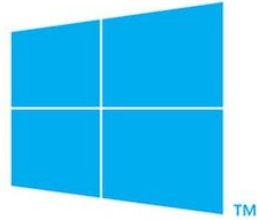
We'll be with you every single step of the way!



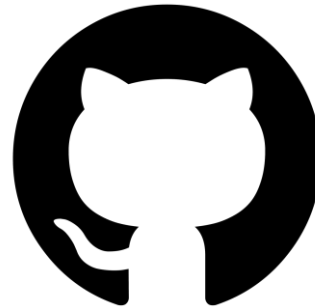
First things first: You need a PC



Requirements for getting started



Windows[®]



**GitHub
Account**



git

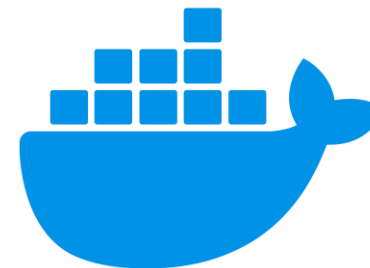
**Basic git
understanding**



**Visual
Studio Code**



**AL Language
Extension**



docker[®]

Get your development environment set up

From 0 to contribution hero in 30 minutes

Demo 2

<https://kepty.cz/OS-Bangkog-Demo2.mp4>

(demos are created by Jesper, Microsoft)

**What other settings
should one make?**

Enable RAD!



The contribution process

Phase III: Create a PR and get it released

Create your Pull Request

And get it released

Demo 3

<https://kepty.cz/OS-Bangkog-Demo3.mp4>

(demos are created by Jesper, Microsoft)

Create a PR and get it released

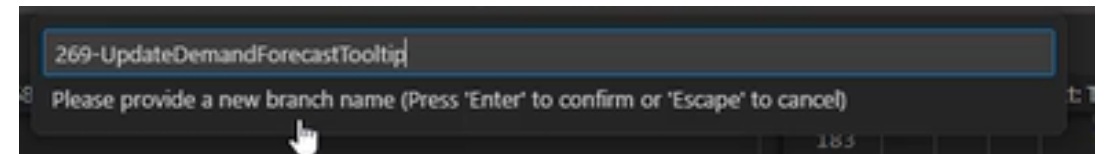
Prerequisite: Have a local repository based on the forked repository

1) Create a new branch locally

Be sure you are in the main branch and that this branch is up-to-date

Create a new branch in VS Code

Use the issue ID as the first part of the branch name



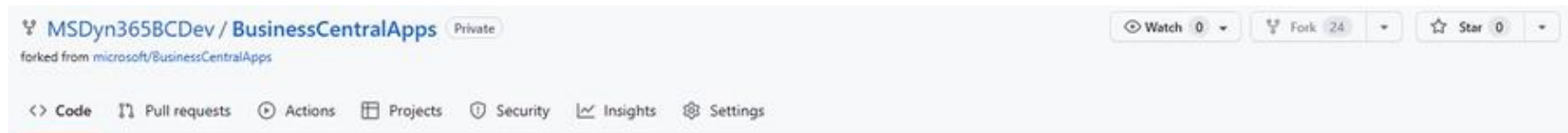
2) Do your development

3) Commit your changes and publish the branch

Create a PR and get it released

4) Create a new pull request

Once a new branch with changes is pushed to the remote repository, GitHub automatically suggest creating a pull request

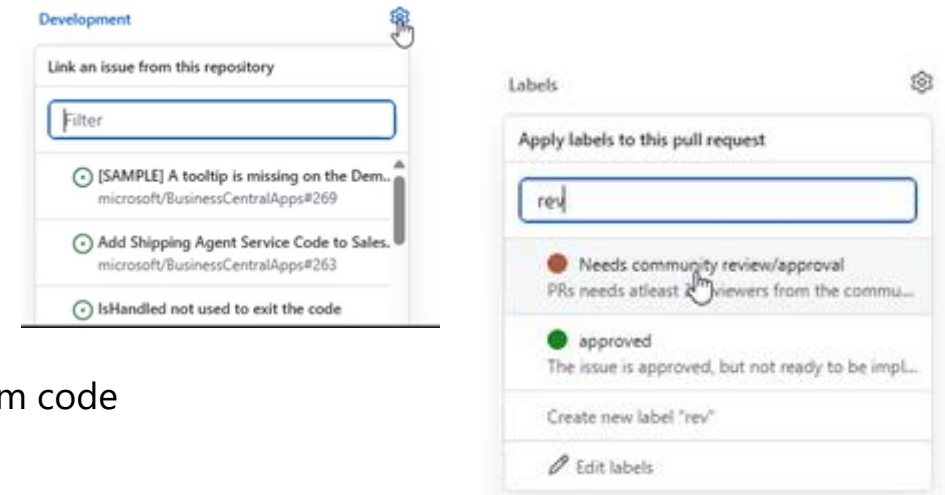


5) Link newly created pull request to the original issue

6) Assign the "Needs community review/approval" label

7) Respond to all suggestions/change requests

It is not just about creating PR, but we need to ensure that suggestions from code reviewers are resolved. Otherwise, our changes will not be deployed.



You already know how to become the contribution hero in 30 minutes.

But true heroes are born by helping others...

So, you can not be the true hero without **reviewing Pull Requests**

Why should we do code review?

You can learn a lot from others.

You can teach others.

You can shape the change and suggest other smaller improvements.

And without code review & code reviewers, our changes will not be approved by Microsoft 😊

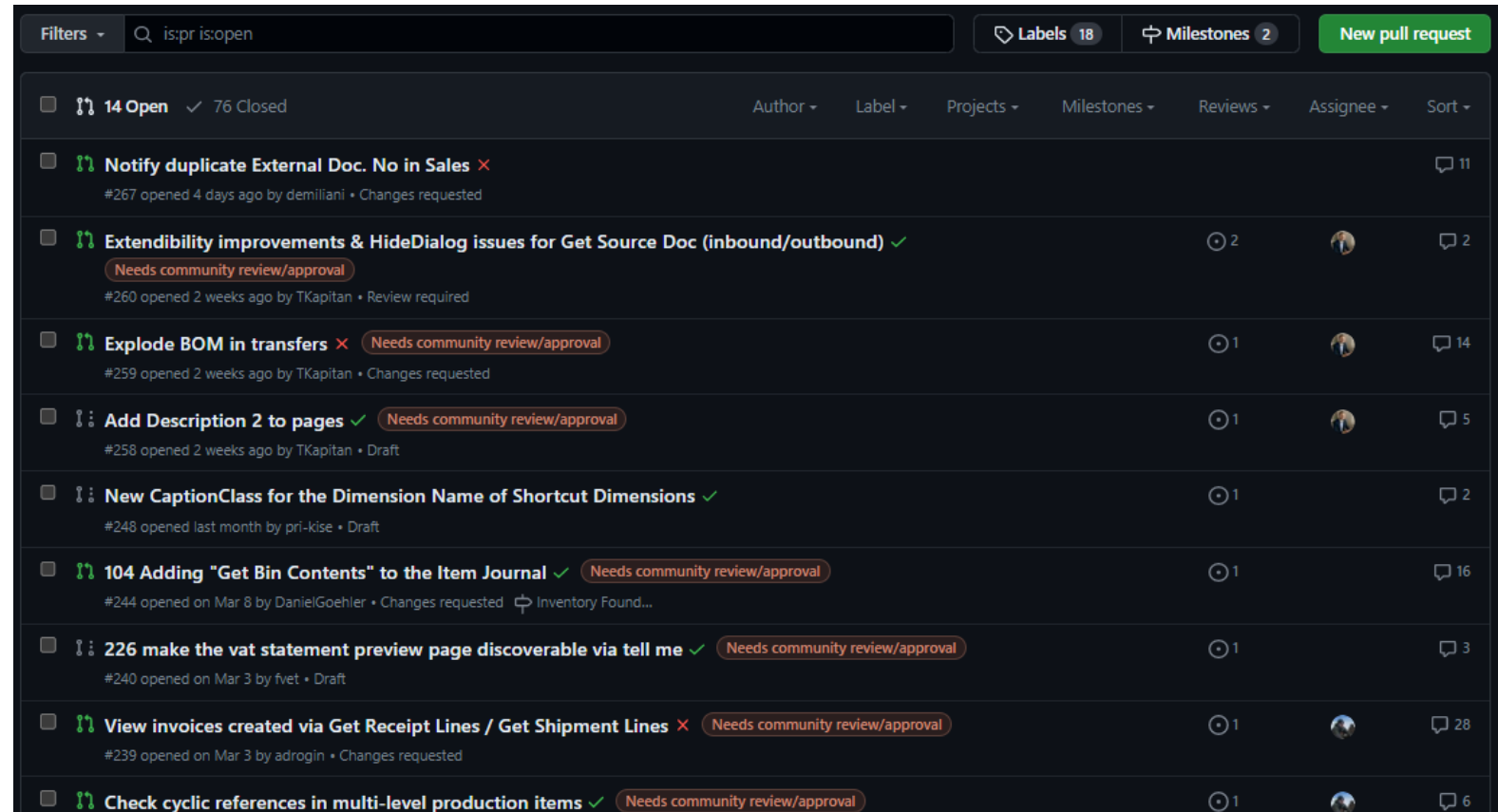
Code Review is one of the most important processes in which any developer should be involved. By reading and trying to understand code made by other developers (and the ideas behind the code), any developer can improve their skills much more quickly and efficiently than by any different approach.

Why should we do code review?

Some stats from existing PRs

- 14 open PRs as of now
- 51 closed PRs in total
 - 9 rejected PRs
- 15 authors of closed PRs
 - 9 with 3+ approved PRs
 - 3 with rejected PRs only
- 2 new authors of open PRs

But only +/- 6 active community code reviewers!



The screenshot shows a GitHub pull request list with the following details:

- Filters: is:pr is:open
- Labels: 18
- Milestones: 2
- New pull request button
- Summary: 14 Open, 76 Closed
- Columns: Author, Label, Projects, Milestones, Reviews, Assignee, Sort
- PRs listed (from top to bottom):
 - #267: Notify duplicate External Doc. No in Sales (Changes requested)
 - #260: Extendibility improvements & HideDialog issues for Get Source Doc (inbound/outbound) (Needs community review/approval)
 - #259: Explode BOM in transfers (Needs community review/approval)
 - #258: Add Description 2 to pages (Needs community review/approval)
 - #248: New CaptionClass for the Dimension Name of Shortcut Dimensions
 - #244: 104 Adding "Get Bin Contents" to the Item Journal (Needs community review/approval)
 - #240: 226 make the vat statement preview page discoverable via tell me (Needs community review/approval)
 - #239: View invoices created via Get Receipt Lines / Get Shipment Lines (Needs community review/approval)
 - Check cyclic references in multi-level production items (Needs community review/approval)

A Developer's Notes

What should I know?

Developer's notes

Always follow best practices and design patterns.

- [Best Practices for AL code - Business Central | Microsoft Learn](#)
- [alguidelines.dev - Business Central Design Patterns](#)

Try to follow the existing code structure ... but do not follow old, non-optimal, practices!

Fix existing code warnings only if they are in the code you are "touching". Otherwise, following what was changed during the code review is much more complicated.

Discuss the planned change with others if you are not entirely sure.

Try to react to requested changes from the code review as soon as possible. It is always complicated (for you as a developer) to respond to suggestions after a few months.

Tests are mandatory for all (almost) changes.

Developer's notes – tests

Sometimes it could be hard to figure out where the new tests belong to. There are more than 1400 test Codeunits, and the structure is not always clear.

You can search for the table/functionality in “\App\Layers\W1\Tests”. Usually, similar tests are split across more Codeunits, so it's up to you to figure out the best place.

Many tests for standard functionality are in the ERM folder.

If you need a new procedures to mock some data, add them to the TestLibraries app folder for better reusability.

Our experience with the pilot program

Bert Verbeek

...it is a pilot. There can be mistakes.
But when I look back it is really a great thing
and together we make the pilot better and better.

There is a lot of progress on it!



Frédéric Vercaemst

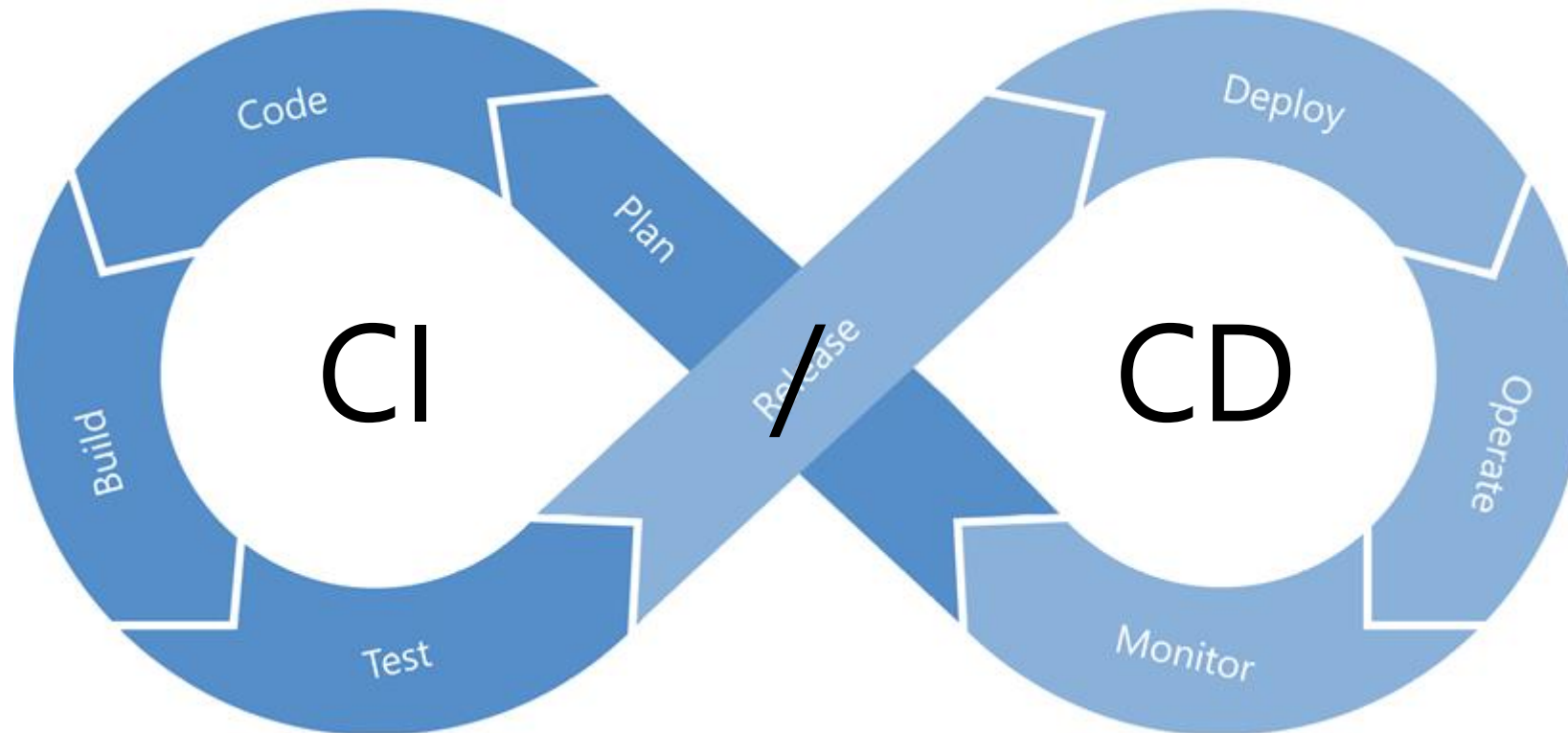
Besides creating my own PR's, I'm a big fan of
code reviewing now and then as well...

just to try to learn from the pro's ;)



What's Next?

What's Next?



"AL:Go for GitHub" powered!

What's Next for the pilot?

- CI/CD advancements
- Code review improvements
- Issue categorization improvements



Time to wrap up...

Do you have any questions?

Next steps

Learn, educate, activate



What's new in 2023 release wave 1
aka.ms/BCReleasePlan



Join the conversation
aka.ms/BCYammer



Leverage learning courses
aka.ms/BCLearn



Follow us on Twitter
<https://twitter.com/MSDYN365BC>



Submit your ideas
aka.ms/BCideas



Thank you !

Please rate the session in the Conference App !

