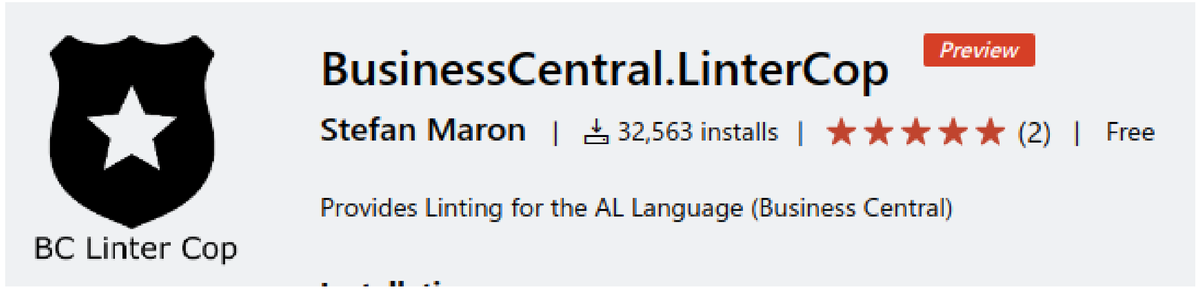
Stefan Maron

- Freelance BC Developer

- github.com/StefanMaron/MSDyn365BC.Code.History
- github.com/StefanMaron/BusinessCentral.LinterCop



- Reach out to me on Twitter/LinkedIn/GitHub via: StefanMaron

Tobias Fenster

- Business
  Managing Partner at

- Community

- Reach out via

  tobiasfenster on Twitter and LinkedIn

  tobias.fenster@hachyderm.io on Mastodon

  tfenster on Github

  tobiasfenster.io: Blog and Podcast « Window on Technology »

Disclaimer

A lot of our content is either very early days and experimental or relying on implementation details which might change without notice. It works today, but it might not work tomorrow and you need to make a conscious decision for which scenarios to use it.

But it should be fun
and give you a glimpse into the future

Agenda

Wasm

- What is Wasm, where does it come from?
- Why does it matter in general and especially for BC developers?
- Demos!

Linux, devcontainers, Codespaces

- AL on Linux, why and how?
- Demo!
- Devcontainers and Codespaces, why and how?
- Even more demos!

Bonus topic if we have time in the end:
AL Language Linux Patcher behind the scenes

What is WebAssembly (Wasm)?

- Safe, portable, low-level code format
  - Near-native performance in a sandboxed environment
  - Hardware-, platform- and language-independent
- Small, efficiently executable files
  - Modular, binary format
  - Streamable and parallelizable
- Open standard: https://webassembly.github.io/spec/core/
- First implementation in browsers
  - Safe, portable, small, efficient, streamable, …
- Examples:
  - BananaBread, built in 2012 as one of the first starting points: kripken.github.io/BananaBread
  - Photoshop, Figma, AutoCAD on the web
  - Google Earth

How to create and run Wasm “things”?

- Systems programming languages (e.g. C/C++, Rust, Go)
  - Relatively easy as compilation to various targets were already in place
- Bytecode languages (e.g. Java, Kotlin, C#)
  - More difficult as they used to compile only into their own intermediate language
- Scripting languages (e.g. Python, JavaScript, Ruby)
  - Only interpreter needs to be ported
- Can run e.g. in all major browser
- Otherwise needs execution environment
  - By default no access to anything local (network, files, …)
  - WebAssembly System Interface (WASI) makes it possbile

Demo

- Hello world in Wasm with WASI using Rust
  - Security sandbox for file access
  - Portability

Codespace

Inspired by https://github.com/bytecodealliance/wasmtime/blob/main/docs/WASI-tutorial.md

Why does it matter in general?

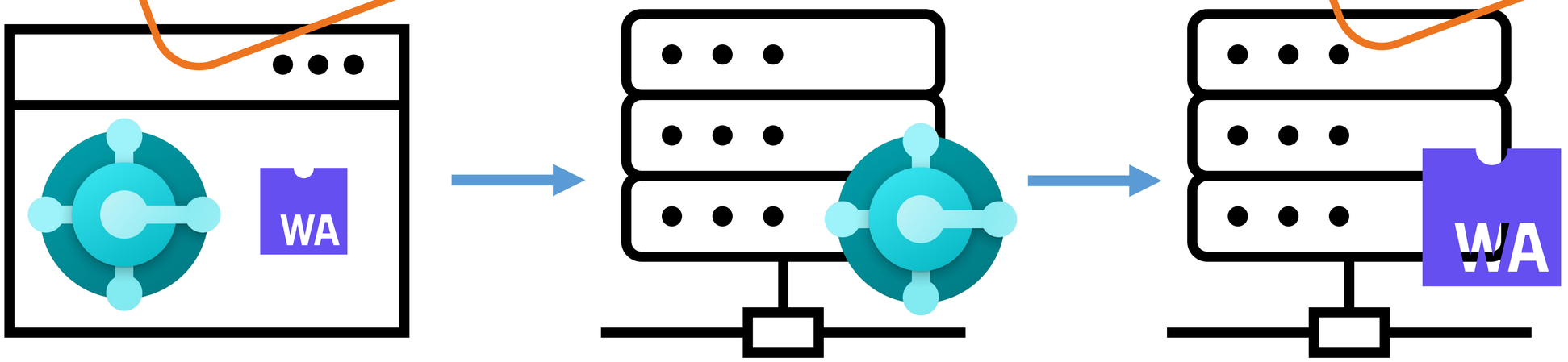
- Already widely used mechanism to make “anything” available in a browser
- Great fit for IoT / edge devices → small, secure, portable, efficient
- Allows “use right tool for the job” approach → one Wasm module can interact with another Wasm module even if created in totally different languages
- Cloud usage for serverless functions has picked up recently

Why does it matter for BC?

- Client side
  - Use best (or only) library / tool / product for a task
- Server side
  - Lightning fast, secure, scalable way to do things BC can't

DEMO

DEMO



Demo client-side

- Live image manipulation for customer pictures in BC
  - Image manipulation using Go, delivered as Wasm module to the browser
  - BC integration with a control addin
  - Wasm module called via JavaScript

Codespace

Inspired by https://github.com/agnivade/shimmer

Demo server-side

- Fast check of a backend service on open page
  - Implemented in Go with Fermyon Spin: “Spin is a framework for building and running event-driven microservice applications with WebAssembly (Wasm) components”
  - Walkthrough on how to create a Spin module
  - See performance
  - Use pre-built storage example with BC integration via HttpClient

[Codespace]

Inspired by https://developer.fermyon.com/spin/kv-store-tutorial

Agenda

Wasm

- What is Wasm, where does it come from?
- Why does it matter in general and especially for BC developers?
- Demos!

Linux, devcontainers, Codespaces

- AL on Linux, why and how?
- Demo!
- Devcontainers and Codespaces, why and how?
- Even more demos!

AL on Linux

Why Linux?

- Full control, trans
  configurability
- Huge developer-c
  ecosystem
- Less demanding c
  more performanc
- No license cost ty
- It's fun ☺

Why isn't everyone using it?

- es with hardware
- ed options
- might need for your
  able – Office, report
  container, C/SIDE, BC
  ssion?), AL
- Less gaming

Why AL on Linux?

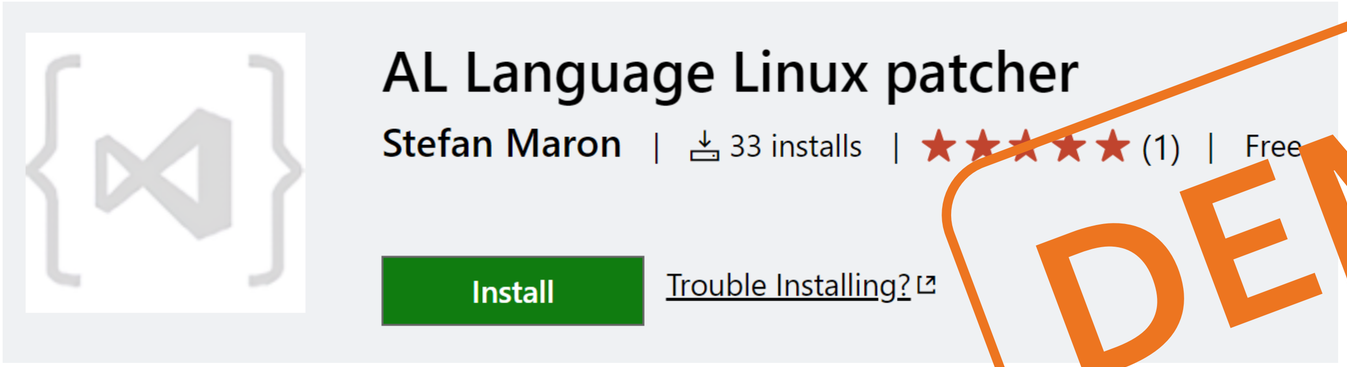
- If you are used to the benefits of Linux especially for development, why not AL?
- Maybe only AL is blocking you…

AL on Linux

Why doesn't it work out of the box?

- Almost there thanks to migration to .NET Core
- Precompiled Windows executable directly called, instead we need to call the generic dll via the dotnet executable
- Small cosmetics to make it work (JSON data types)

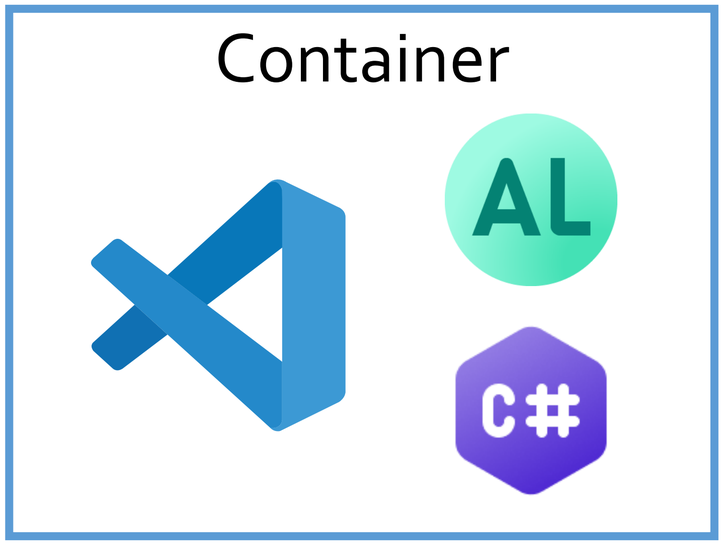
How can we fix it?



DEMO

Devcontainers and Codespaces, why?

Containerized development ftw!



Containerized dev – Why?

- Cleanly separated development systems with better resource utilization than e.g. with VMs
  - No version conflicts and side effects
  - No "littering" → simply discard and rebuild

- All dependencies, tools, etc. including versions described in configuration files in the repo
  - IaC approach for local development environments
  - No configuration drift between different developers
  - Clear and simple rollout of changes in the development stack

- Extremely fast setup of development environments
  - Thus also extremely fast onboarding of new development
  - Easy and clean switching between projects

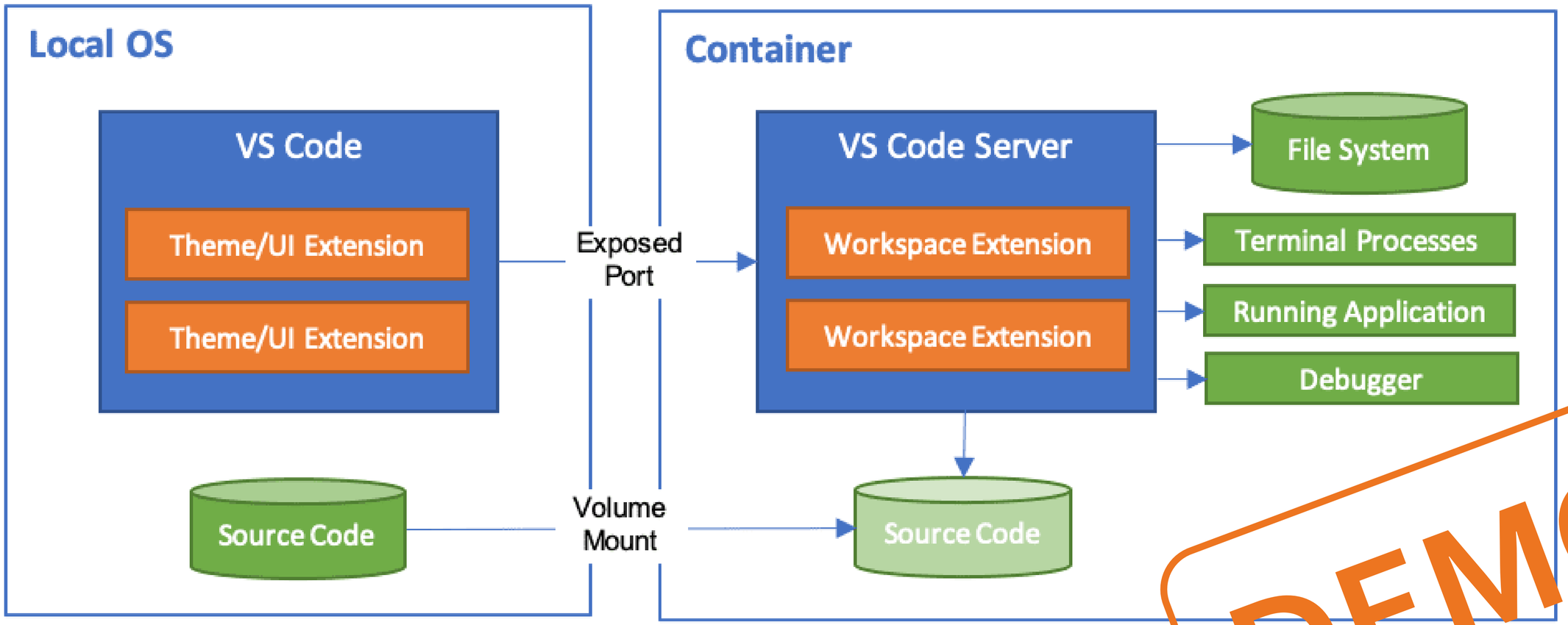
Containerized dev – How?

- Visual Studio Code development container (devcontainer)
  - Containerized, configurable, local development environment
  - Connection via VS Code
  - Full functionality incl. extensions, access to local resources and offline support

→ Very good development environment for all scenarios (except Windows-based development…)

- Ideal starting point for GitHub Codespaces (same technology)

VS Code devcontainers

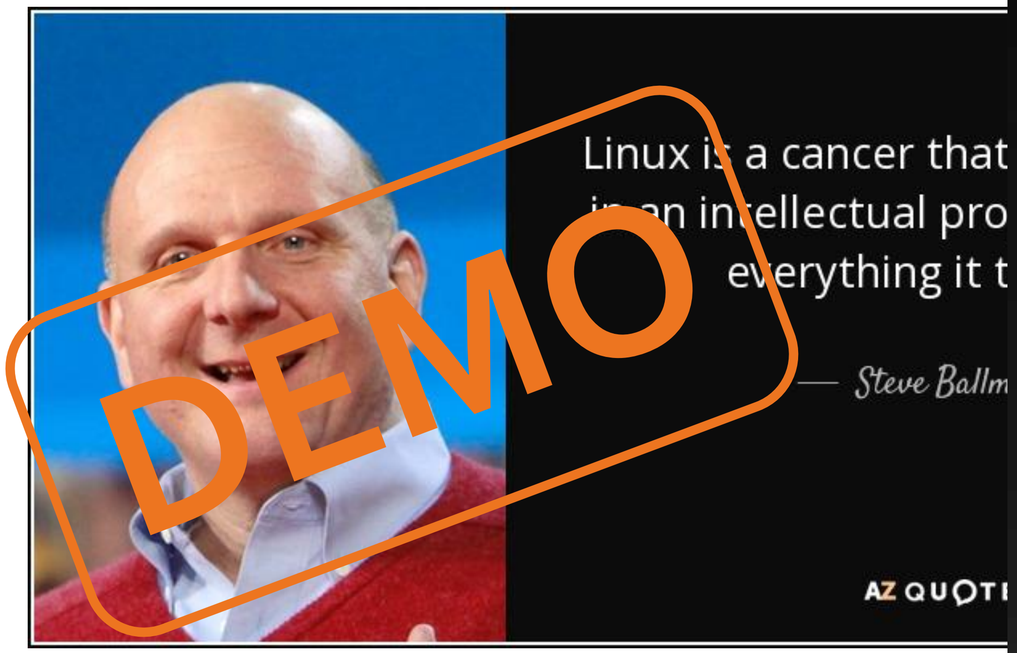


DEMO

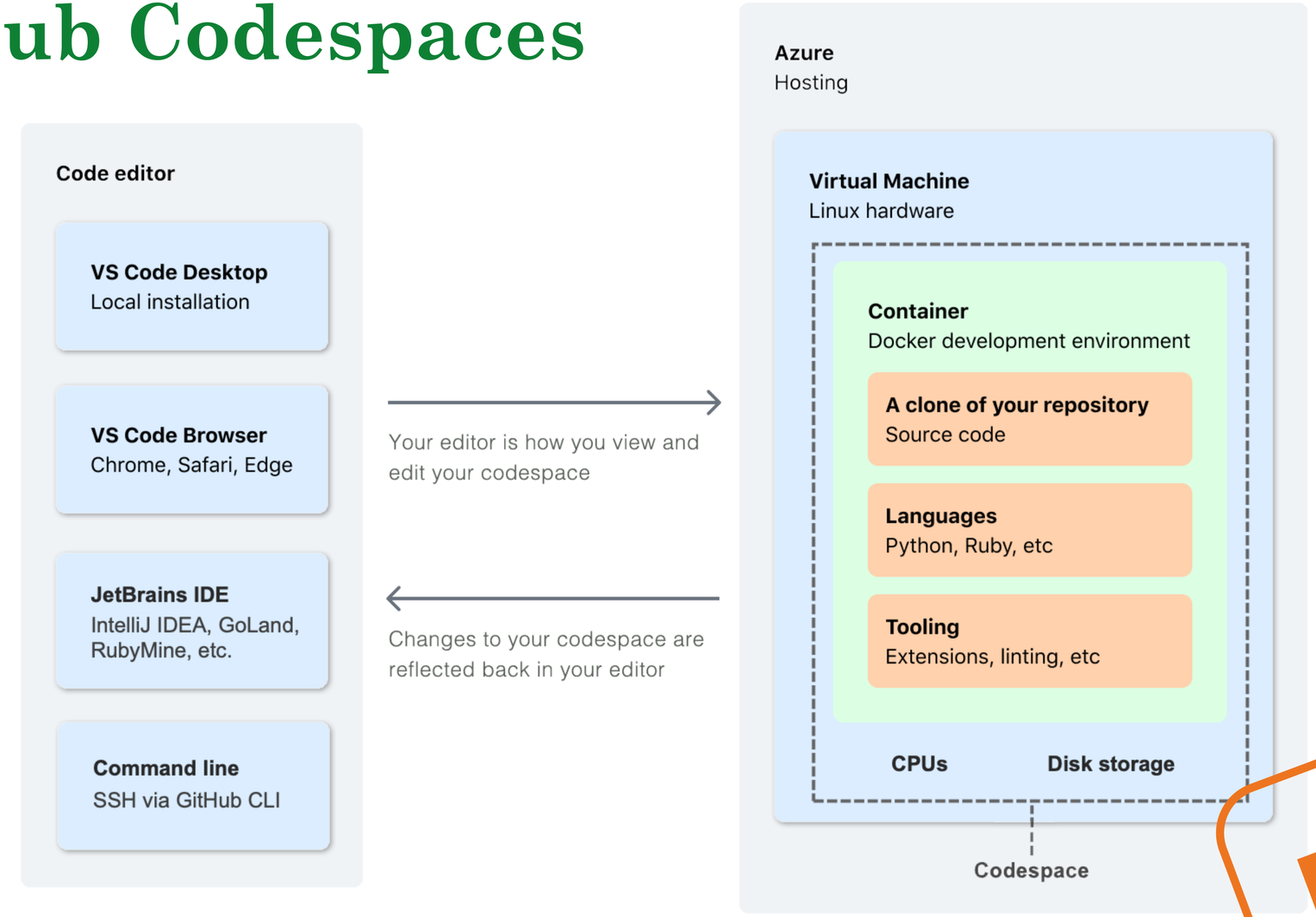
Source: https://code.visualstudio.com/docs/devcontainers/containers

VS Code devcontainers

- No support for Windows containers
- But for Windows hosts! → Windows Subsystem for Linux (easiest with Docker Desktop)



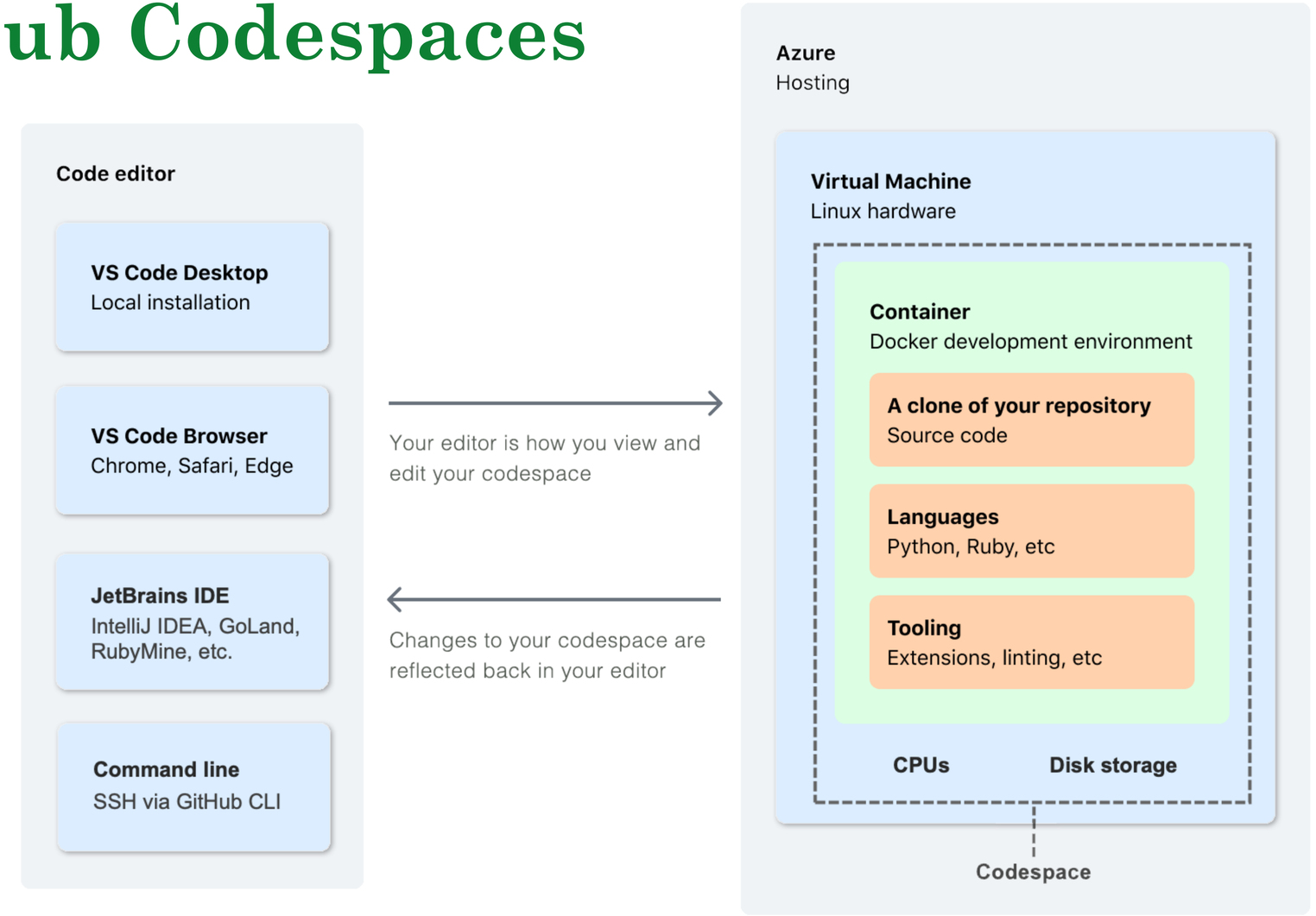
GitHub Codespaces

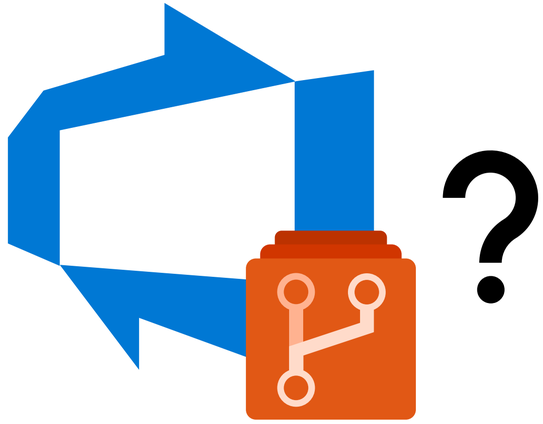


DEMO

Source: https://docs.github.com/en/codespaces/overview

GitHub Codespaces





Source: https://docs.github.com/en/codespaces/overview

DevOps Repo

- Use a devcontainer “feature” to clone the repo into the Codespace:

```
{
    "name": "C# (.NET)",
    "image": "mcr.microsoft.com/devcontainers/dotnet:0-6.0",
    "features": {
        "ghcr.io/microsoft/codespace-features/external-repository:latest": {
            "cloneUrl": "https://dev.azure.com/demo-codespaces/al-demo/_git/al-demo-app",
            "cloneSecret": "ADO_PAT",
            "folder": "/workspaces/al-demo-app"
        }
    },
    "customizations": {
```

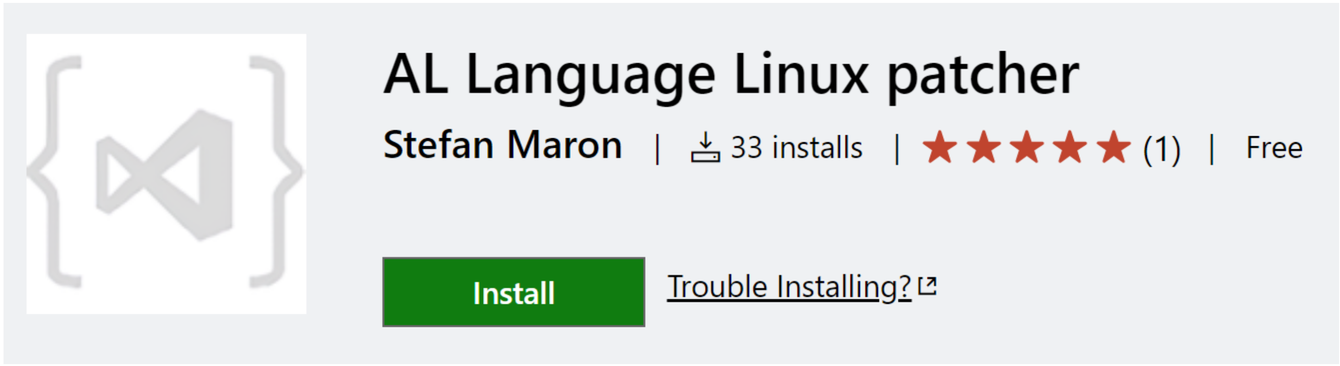
- Check https://github.com/tfenster/azdevops-codespaces and https://tobiasfenster.io/al-development-in-a-github-codespace-with-sources-in-an-azure-devops-repo for the details

BC/NAV TECH DAYS 2023
mibuso.com

Q&A

Any Questions?

Bonus topic


AL Language Linux patcher
Stefan Maron | 33 installs | (1) | Free
Install
Trouble Installing?

BC/NAV TECH DAYS 2023
mibuso.com

Thank You!