CONTAINERISIERUNG SCHON IN DER (LOKALEN) ENTWICKLUNGSUMGEBUNG, WARUM UND WIE?

SPRECHER INTRO

TOBIAS FENSTER

Business

- Managing Director bei 4PS Deutschland, Teil der 4PS Gruppe
  - Hersteller eines Cloud-basierten ERP für die Baubranche

Community

- Microsoft Regional Director und MVP für Azure und BC
- Docker Captain

Social und Blog

- tobiasfenster bei Twitter und LinkedIn
- tobiasfenster@hachyderm.io bei Mastodon
- tobiasfenster.io
- “Window on Technology” podcast

WARUM?

CONTAINERBASIERTE ENTWICKLUNG

WARUM?

- Sauber getrennte Entwicklungssysteme bei besserer Ressourcennutzung als z.B. mit VMs
  - Keine Versionskonflikte und Seiteneffekte
  - Kein „Vermüllen“ → einfach wegwerfen und neu erstellen
- Alle Abhängigkeiten, Tools, etc. inklusive Versionen in Konfigurationsdateien im Repo beschrieben
  - IaC-Ansatz für lokale Entwicklungsumgebungen
  - Kein Auseinanderdriften unterschiedlicher Entwickler
  - Klarer und einfacher Rollout von Änderungen im Entwicklungs-Stack
- Extrem schnelles Setup von Entwicklungsumgebungen
  - Damit auch extrem schnelles Onboarding neuer Entwickler
  - Einfaches und sauberes Wechseln zwischen Projekten

VS CODE ENTWICKLUNGSCONTAINER

WIE?

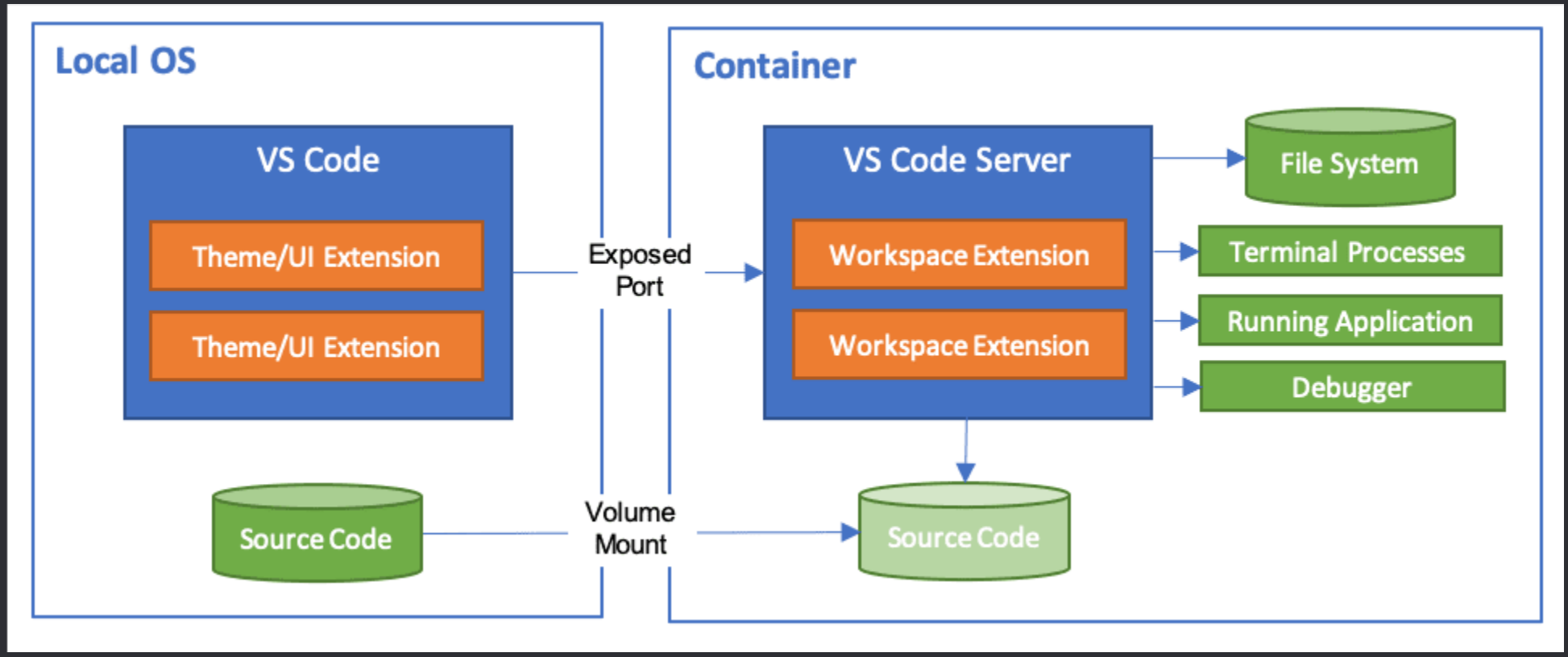
- Visual Studio Code Entwicklungscontainer
  - Containerisierte, konfigurierbare, lokale Entwicklungsumgebung
  - Verbindung über VS Code
  - Volle Funktionalität inkl. Extensions und Zugriff auf lokales und Offline-Support

→ Sehr gute Entwicklungsumgebung für alle Szenarien (außer Windows-basierte Entwicklung…)

- Idealer Ausgangspunkt für GitHub Codespaces (gleiche Technologie)

ARCHITEKTUR

VISUAL STUDIO CODE ENTWICKLUNGSCONTAINER



Quelle: https://code.visualstudio.com/docs/remote/containers

SERVERLESS ENTWICKLUNG

WARUM?

- Noch schnelleres und einfacheres Setup direkt im Browser
- Keine lokale Infrastruktur mehr, entsprechend auch keine lokale Abhängigkeit
  - Verwendete (und zu bezahlende) CPU/RAM/Storage konfigurierbar; pay per use
  - Entwicklung auf dem iPad?!
- (Fast) alle Vorteile von Entwicklungscontainern
  - Kein Offline-Support
  - Kein Zugriff auf lokale Ressourcen
- GitHub selbst nutzt es für den Großteil: Over the past months, we've left our macOS model behind and moved to Codespaces for the majority of GitHub.com development. (https://github.blog/2021-08-11-githubs-engineering-team-moved-codespaces/)

GITHUB CODESPACES

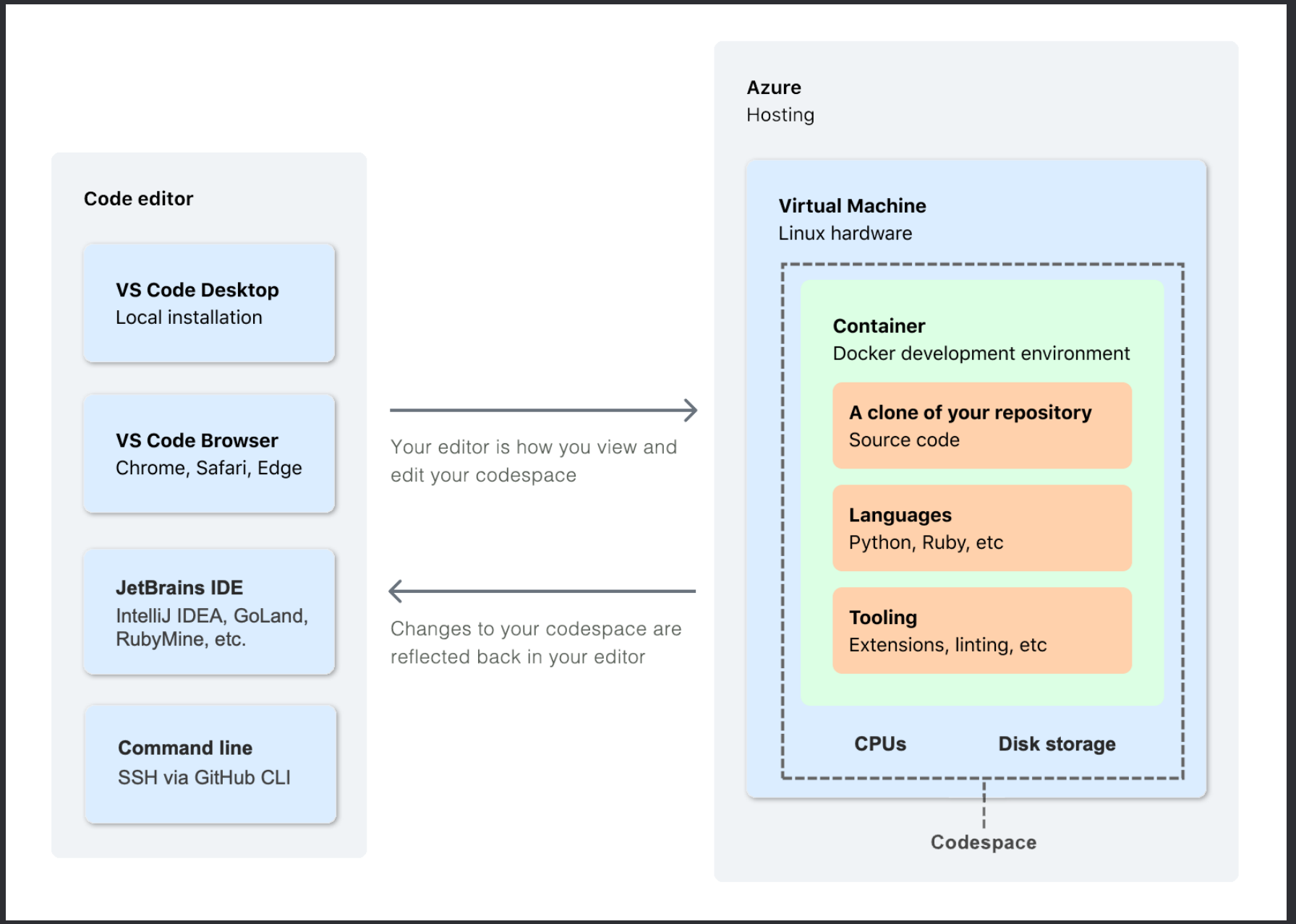
WIE?

- GitHub Codespaces
  - Containerisierte, konfigurierbare, Cloud-Entwicklungsumgebung
  - Verbindung über VS Code oder Browser
  - Volle Funktionalität inkl. Extensions

→ Sehr gute Entwicklungsumgebung für die meisten Szenarien

ARCHITEKTUR

GITHUB CODESPACES



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

MÖGLICHE ALTERNATIVEN

- Gitpod (gitpod.io)
  - Open Source
  - Kein Support für devcontainer-Standard
  - Reines Cloud-Angebot (früher auch self-hosted)

- Coder (coder.com)
  - Open Source
  - Support für devcontainer-Standard
  - Reines Cloud-Angebot (auch self-hosted, lokal auf der Roadmap)

- Docker Desktop Dev Environments (docs.docker.com/desktop/dev-environments)
  - Closed Source, Integriert in Docker Desktop
  - Kein Support für devcontainer-Standard
  - Rein lokale Nutzung (Umweg über VM in der Cloud)

- DevPod (devpod.sh)
  - Open Source
  - Support für devcontainer-Standard
  - Lokal und Cloud möglich

VIELEN DANK!

WELCHE FRAGEN DARF ICH BEANTWORTEN?

BONUS THEMA: MULTI-CONTAINER ENTWICKLUNG

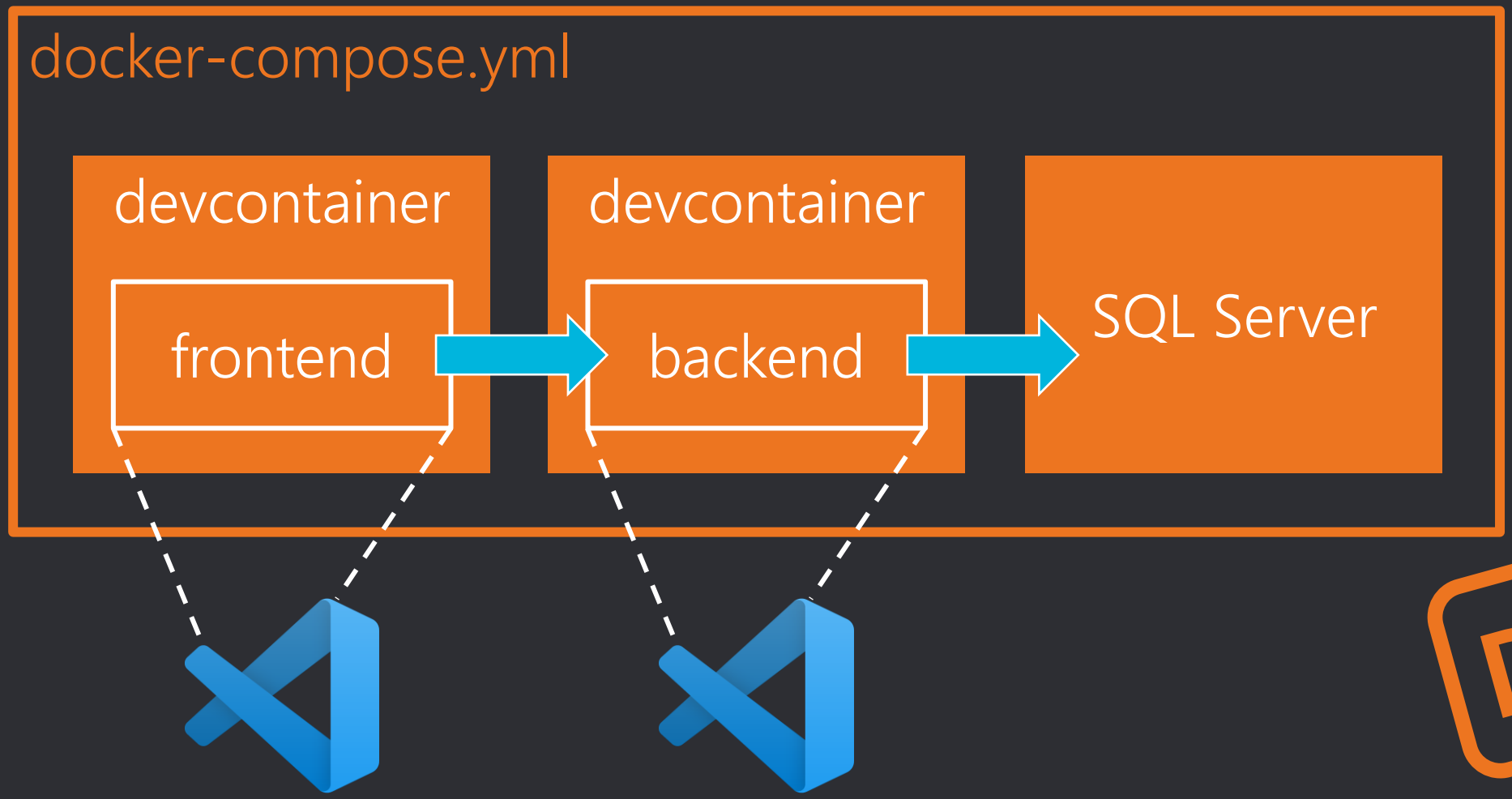
MULTI-CONTAINER ENTWICKLUNG

WAS WENN EIN CONTAINER NICHT REICHT?

- Szenario:
  - Backend wie bisher mit .NET webapi
  - Frontend mit Blazor
  - SQL Server
- Definiert in gemeinsamer docker-compose.yml
- Zwei VS Code Instanzen verbunden mit je einem Container

MULTI-CONTAINER ENTWICKLUNG

WAS WENN EIN CONTAINER NICHT REICHT?



DEMO