



# Containerbasierte Entwicklung für .NET-Entwickler

Tobias Fenster

# Special Day

## Cloud-Native Business Applications

Thema	Sprecher	Datum, Uhrzeit
<b>Cloud-Native-all-the-things: Definition, Praktiken &amp; Patterns</b>	Thorsten Hans, Christian Weyer	DI, 21. Februar 2023, 10.45 bis 11.45
<b>Containerbasierte Entwicklung für .NET-Entwickler</b>	Tobias Fenster	DI, 21. Februar 2023, 12.15 bis 13.15
<b>Was guckst du? Observability von Cloud-Native-Anwendungen – mit OpenTelemetry</b>	Thorsten Hans	DI, 21. Februar 2023, 15.30 bis 16.30
<b>Cloud-Native Microservices: On-Premises oder in der Cloud – mit Dapr</b>	Christian Weyer	DI, 21. Februar 2023, 17.00 bis 18.00
<b>Serverless Containers mit Azure Container Apps</b>	Thorsten Hans	DI, 21. Februar 2023, 19.00 bis 20.00

SPRECHER INTRO

# TOBIAS FENSTER



## Business

- Managing Partner 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, Portainer und Traefik ambassador

## Social und Blog

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



# CONTAINERBASIERTE ENTWICKLUNG FÜR .NET

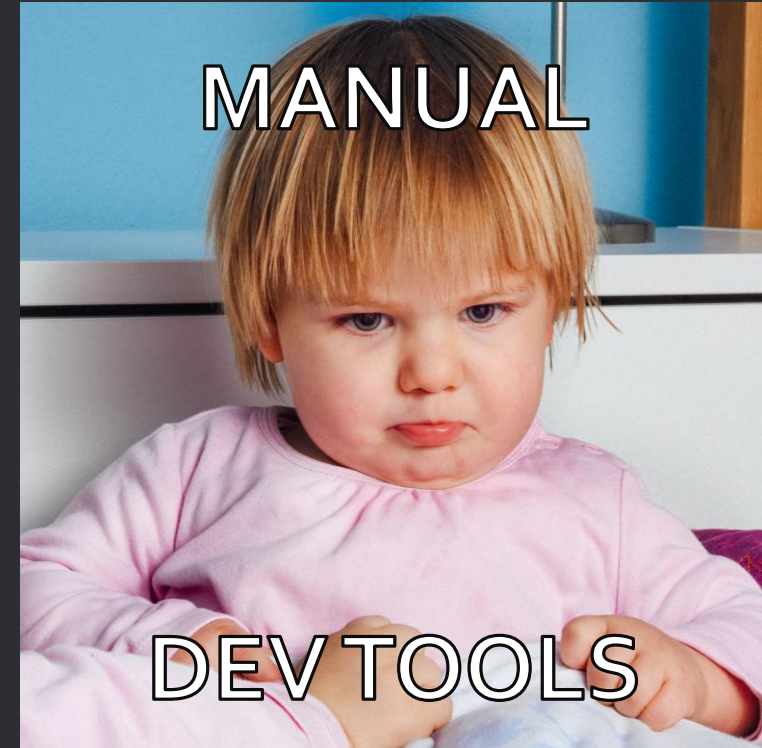
WÄHREND DER  
ENTWICKLUNG

WÄHREND DER  
LAUFZEIT

# CONTAINERBASIERTE ENTWICKLUNG FÜR .NET

CONTAINERBASIERTE ENTWICKLUNG

WARUM?



# WARUM?

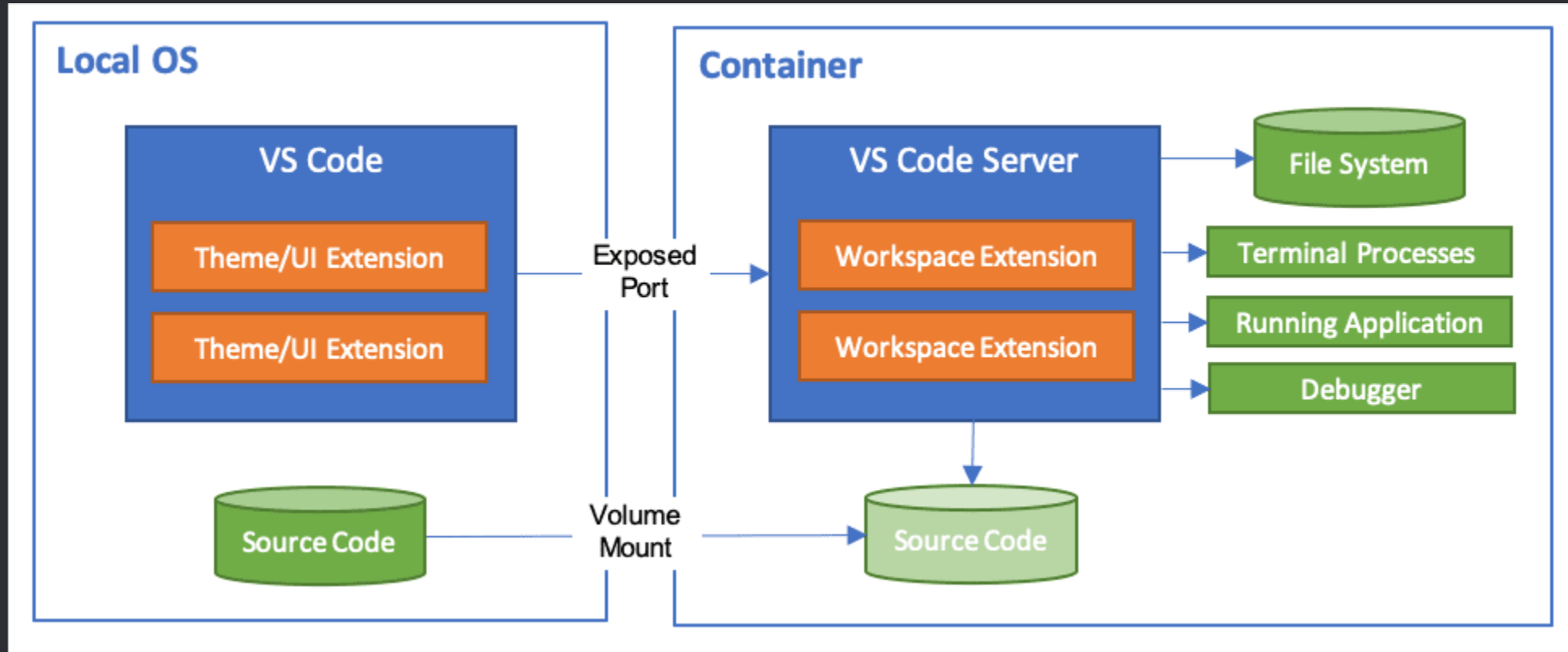
- **Sauber getrennte Entwicklungssysteme** bei besserer Ressourcennutzung als z.B. mit VMs
  - Keine Versionskonflikte und Seiteneffekte
  - Kein „Vermüllen“ → einfach wegwerfen und neu erstellen
- Alle Abhängigkeiten, Tools, etc. inklusive Versionen in **Konfigurationsdateien im Repo** beschrieben
  - 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 Entwicklung
  - Einfaches und sauberes Wechseln zwischen Projekten

## 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)



# VISUAL STUDIO CODE ENTWICKLUNGSCONTAINER



DEMO

# BEST PRACTICES FÜR PRODUKTIV-CONTAINER

BEST PRACTICES

# DEMO SZENARIO

- Projekt „project“ in devcontainer: `dotnet new webapi`
- Aufgaben:
  - „Containerisieren“
  - Container image optimieren
  - Vulnerabilities finden und fixen



# WELCHES BASE IMAGE?

- Image zum **Build**: [mcr.microsoft.com/dotnet/sdk](https://mcr.microsoft.com/dotnet/sdk)
  - .NET CLI
  - .NET runtime
  - ASP.NET Core runtime
- **Niemals latest!**
  - Heute 7.0, vor kurzem noch 6.0...
  - Bei Abhängigkeiten auch z.B. 6.0.8
- An die Größe (Download, Attack Surface) denken → **alpine**



# WIE DAS IMAGE AUFBAUEN?

- Anfangs nur `.csproject` kopieren und `restore` ausführen
  - Bei Änderungen am Code Cache nutzbar (Layering-Mechanismus in Images)
- Schneller, optimierter `Publish`
  - `no-restore`
  - `PublishTrimmed`, `PublishReadyToRun`, `PublishSingleFile`
- `Multi-stage` images
  - Image für `Runtime`:
    - `mcr.microsoft.com/dotnet/runtime`: .NET runtime
    - `mcr.microsoft.com/dotnet/aspnet`: ASP.NET Core runtime
  - Image für `Runtime mit self-contained executable`:
    - `mcr.microsoft.com/dotnet/runtime-deps`: nur notwendige dependencies



# WAS KOMMT IN DEN BUILD CONTEXT?

- So wenig lokale Artefakte wie möglich im **Build Context**
  - .dockerignore analog .gitignore
  - Generieren lassen von VS Code...
  - Aber wir schauen rein



# WIE GEHT ES DEM CONTAINER?

- Container können „health“ Status anzeigen
  - Definition ebenfalls im Dockerfile
- Wichtig z.B. bei Orchestratoren wie Kubernetes
  - Erkennen, wann ein Container ersetzt werden muss





# WIE SOLL DER CONTAINER SICH VERHALTEN?

- Normale Konfigurationsmechanismen in .NET 1:1 auch auf Container abbildbar:
  - AspNetcore\_environment
  - Environment variables



# WELCHER USER?

- **Non-root** unter Linux
  - User anlegen, ohne Passwort
  - Zugriff auf Verzeichnis geben
- System = gMSA unter Windows



# GEHT DAS NOCH EINFACHER UND OHNE CONTAINER-TOOLING?

- .NET 7 bringt direkten Support für die Erstellung von Container Images
  - Ganz ohne Dockerfile (und damit auch mit weniger Kontrolle)
  - Lokale Images unter lokaler Nutzung von Docker
  - Bei Nutzung von remote Repos komplette ohne Docker möglich
- Bisher nur Erfahrung durch erstes Ausprobieren und [lippertmarkus.com](https://lippertmarkus.com) / offizielle Announcements
- Mehr wird kommen, Gaps sind bereits dokumentiert (kein Windows, kein Auth gegen remote Repos)



# BONUS THEMA: 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

# WAS WENN EIN CONTAINER NICHT REICHT?

docker-compose.yml

devcontainer

frontend

devcontainer

project

SQL Server



**DEMO**

**VIELEN DANK!**

**WELCHE FRAGEN DARF ICH  
BEANTWORTEN?**

# Special Day

## Cloud-Native Business Applications

Thema	Sprecher	Datum, Uhrzeit
<b>Cloud-Native-all-the-things: Definition, Praktiken &amp; Patterns</b>	Thorsten Hans, Christian Weyer	DI, 21. Februar 2023, 10.45 bis 11.45
<b>Containerbasierte Entwicklung für .NET-Entwickler</b>	Tobias Fenster	DI, 21. Februar 2023, 12.15 bis 13.15
<b>Was guckst du? Observability von Cloud-Native-Anwendungen – mit OpenTelemetry</b>	Thorsten Hans	DI, 21. Februar 2023, 15.30 bis 16.30
<b>Cloud-Native Microservices: On-Premises oder in der Cloud – mit Dapr</b>	Christian Weyer	DI, 21. Februar 2023, 17.00 bis 18.00
<b>Serverless Containers mit Azure Container Apps</b>	Thorsten Hans	DI, 21. Februar 2023, 19.00 bis 20.00