

Infoma 

D365 BUSINESS CENTRAL CONTAINER UND APIS AI MEETS BIZAPPS

Tobias Fenster
München, 13.10.2018



Tobias Fenster

Technischer Leiter bei Axians Infoma

Microsoft MVP für Business Applications

↑ @tobiasfenster und ↑ <https://navblog.axians-infoma.de>

Agenda

- Überblick: D365 Business Central **custom App Typen** und Betrieb im Container
 - Hands on 1: **Azure Container Instance** mit D365 BC
 - Hands on 2: **API Client** gegen BC verbinden (Connect App)
 - Hands on 3: Container **konfigurieren und Verhalten anpassen**
 - Hands on 4: **Eigenes** Business Central Image erzeugen
-

- ▶ Microsoft unterscheidet drei Arten von Apps:
 - ▶ **Connect App**: Verbunden über **Schnittstelle / API**, App selbst aber außerhalb BC
 - Anbindung über **OData / REST** oder **SOAP** mit Azure AD (produktiv) oder Basic Auth / WebService Key (Entwicklung)
 - Entwicklung in **beliebiger Programmiersprache**
 - Scope von Hands on 2
 - ▶ **Add-on App**: Erweitert BC um **eigene Datenstrukturen und Logik**, interagiert mit der bereits vorhandenen Business Logik über Events
 - Umsetzung über sog. **Extensions**
 - Entwicklung in **AL mit Visual Studio Code**
 - Scope von Hands on 4
 - ▶ **Embed App**: **Tief integrierte Änderungen und Anpassungen** an Strukturen und Logik
 - Umsetzung ebenfalls über **Extensions**, teilweise aktuell noch **direkte Codeänderungen** notwendig
 - Entwicklung bis 2020 noch teilweise in **C/AL mit C/SIDE**, ansonsten auch **AL mit VS Code**
-

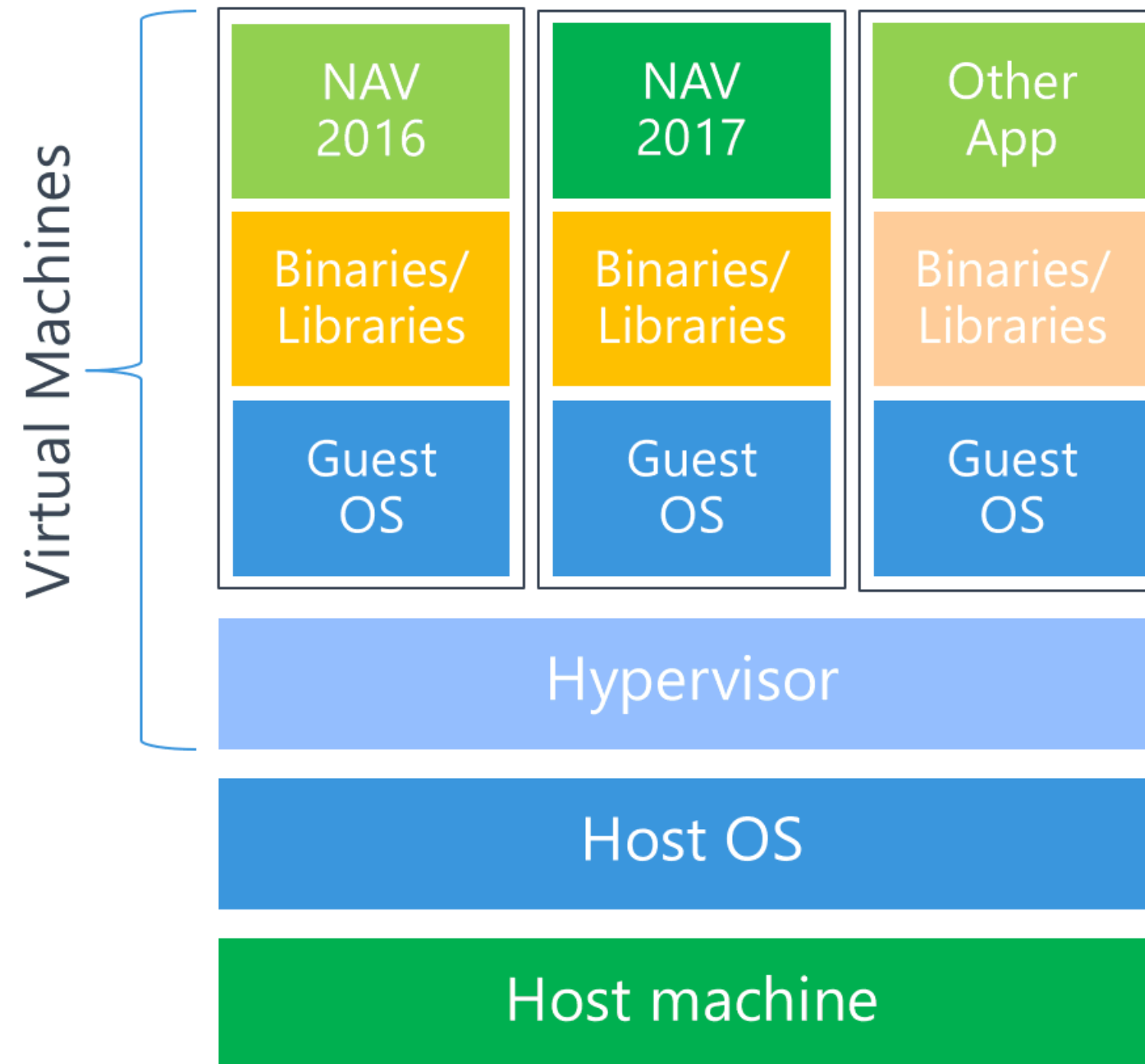
- ▶ **Docker**: Führende cross-platform Umgebung für Software Container (einzige Option unter Windows)
 - ▶ Was sind Docker Container, Image und Host?
 - Ein **Image** ist ein **Template** mit minimal notwendigem Betriebssystem, Bibliotheken und Anwendungslibraries und wird über ein Dockerfile definiert
 - Ein **Container** ist die **Instanz eines Image** mit einer nicht veränderbaren Basis aus dem Image und darauf allen Änderungen
 - Ein Container ist **keine VM**, insbesondere hat er **keine GUI** und kann **nicht per RDP** verbunden werden
 - Ein Docker **Host** ist die physische oder virtuelle Maschine, auf der **Container laufen**
 - ▶ Wesentliche Vorteile:
 - **Einfacher** Weg, um Deployments / Konfigurationen **sehr stabil und sicher reproduzierbar** darzustellen (kein "works here", vermeidet Gap Dev vs. Ops)
 - **Bessere Ressourcennutzung** als mit klassischen VMs, insbes. da es kein Guest OS gibt, sondern der Kernel des Host **direkt verwendet** wird
 - Großes Ökosystem vorhandener Images in sog. Registries, primär auf Docker Hub
-

- ▶ Microsoft bietet Container Images für NAV ab 2015 und für Business Central als Sandbox für die Cloud-Variante sowie direkt für On-Prem
 - ▶ Aktuell nur für Entwicklung und Test, nicht für Produktion freigegeben
 - ▶ Betrieb auf eigenem Docker Host:
 - Unter Windows 10 funktioniert Docker Community Edition: Neuer, aber instabiler
 - Unter Windows Server 2016/19 funktioniert Docker Enterprise Edition (kostenlos in Windows Server enthalten): Etwas älter, aber deutlich stabiler
 - Wenn der eigene Laptop der Docker Host ist: Evtl. Betrieb einer Windows-Server-VM mit Docker
 - ▶ Betrieb auf Azure:
 - Diverse Varianten, u.a. Azure Container Instances als sehr schneller, einfacher Weg
 - ▶ Hochverfügbarkeit ist analog zu "normalen" Installationen oder über Docker Swarm möglich
-

- ▶ Images für **alle supporteten NAV / BC Releases** (technisch ab 2013R2 möglich) mit entsprechenden CUs und Länderversionen, z.B. `microsoft/dynamics-nav:2017-cu14-de`
 - Image beinhaltet **SQL, NAV und IIS**, wobei alles außer NAV **deaktiviert** werden kann
 - Jeder Container beinhaltet genau **1 NAV Instanz** (multi-tenancy ist möglich)
 - Windows-Anmeldung ist möglich, bedarf aber etwas mehr Aufwand
 - Windows Client und finsql sind per ClickOnce verfügbar
 - Sehr viel über **Parametrisierung** möglich, alternativ oder zusätzlich sind **Scripts im Image überschreibbar**
 - ▶ **Öffentlich** verfügbare Image-Typen: `dynamics-nav`, `bcsandbox`, `bconprem`
 - **Geschlossene Beta** für `bcsandbox` (daily build) und `bcsandbox-master` (nächstes Release) über Collaborate
 - Erläuterung (noch vor `bconprem`): [↑ What Docker Image is right for you?](#)
 - ▶ Informationen insbesondere hier: [↑ https://blogs.msdn.microsoft.com/freddyk/](https://blogs.msdn.microsoft.com/freddyk/) und Unterstützung bei Fehlern hier [↑ https://github.com/microsoft/nav-docker](https://github.com/microsoft/nav-docker)
 - Intro in Docker allgemein und NAV im Container: [↑ https://www.youtube.com/watch?v=9c5Yl51yXb8](https://www.youtube.com/watch?v=9c5Yl51yXb8)
-

-
- ▶ Überblick: D365 Business Central custom App Typen und Betrieb im Container
 - ▶ Hands on 1: Azure Container Instance mit D365 BC
 - ▶ Hands on 2: API Client gegen BC verbinden (Connect App)
 - ▶ Hands on 3: Container konfigurieren und Verhalten anpassen
 - ▶ Hands on 4: Eigenes Business Central Image erzeugen
-

Virtual Machines



- ▶ Im **Azure Portal**: Neue Resource Group "test" in West Europe anlegen, dann neue **Container Instance** → Einfacher Wizard, nicht alle Features
 - ▶ Schritt 1 **Basics** ausfüllen und als **Image microsoft/bcsandbox:de** verwenden
 - ▶ Schritt 2 **Configuration**:
 - OS Type: Windows / Number of cores: 2 / Memory (GB): 7
 - DNS name: eindeutiger Wert / Port: 443 / Open additional ports: Yes / Port: 7049 / Port: 8080
 - Environment variable: "accept_eula":"Y"
 - Rest Default
 - ▶ Schritt 3 **bestätigen** und warten...
 - ▶ Im Azure Portal über **Container Instance** → **Containers** → **Events** warten bis "Pulling" beendet ist (5-10 Minuten) und Container Status "Running" bekommt. Zwischenzeitliche Failures beim Pulling sind unbedenklich, wird automatisch erneut versucht
 - ▶ Unter **Overview** den **FQDN** suchen und mit `http://<fqdn>:8080` auf Dateien bzw. `https://<fqdn>/nav` auf BC zugreifen
-

- Per **Azure CLI** z.B. in der [↑]Cloud Shell automatisierbar

```
az container create -g test --name testcont --image microsoft/bcsandbox:de \
--os-type Windows --cpu 4 --memory 7 --ports 80 443 7048 7049 8080 \
--dns-name-label tst124tfe \
-e accept_eula=Y username=admin password=Passw0rd*123 \
ContactEMailForLetsEncrypt=tobias.fenster@axians-infoma.de \
PublicDnsName=tst124tfe.westeurope.azurecontainer.io \
customNavSettings=ApiServicesEnabled=true \
folders=c:\\run\\my=https://github.com/Azure/azure-quickstart-
templates/raw/master/101-aci-dynamicsnav/scripts/SetupCertificate.zip
```

- Mit dns-name-label, PublicDnsName und ContactEMailForLetsEncrypt (keine Registrierung notwendig) wird ein **valides SSL-Zertifikat** abgerufen
- Unbedingt **dns-name-label und eMail-Adresse anpassen...**
- Präsentation unter [↑]<https://ve.link/AlMeetsBizAppsHandson> für Copy und Paste

HANDS ON 1.1: ACI IM PORTAL



- ▶ Schnellere Alternative: Azure Resource Manager (ARM) **Template** nutzen, in dem Teile schon **vordefiniert** sind
- ▶ **Azure Quickstart Template** für NAV / BC unter https://ve.link/bc_aci → In Azure bereitstellen
- ▶ Resource group auswählen
- ▶ Dns Prefix (muss **eindeutig** sein) und Lets Encrypt Mail angeben → **Valides SSL-Zertifikat**
- ▶ Nav Release auf microsoft/bcsandbox:de ändern
- ▶ Username und Password vergeben, Cpu Cores auf 4, Memory in Gb auf 8, Accept Eula auf Y
- ▶ Ebenfalls bestätigen und per Azure Portal Fortschritt verfolgen
- ▶ Hat neben Port 443, 8080 und 7049 auch **7048** (OData Service) offen

- Per **Azure CLI** z.B. in der  Cloud Shell automatisierbar

```
az group deployment create --name mydeployment --resource-group test \
  --parameters dnsPrefix=tst123tfe letsEncryptMail=tobias.fenster@axians-
infoma.de \
  navRelease=microsoft/bcsandbox:de username=admin password=Passw0rd* \
  cpuCores=4 memoryInGb=8 acceptEula=Y \
  --template-uri https://raw.githubusercontent.com/Azure/azure-quickstart-
templates/master/101-aci-dynamicsnav/azuredeploy.json
```

- Unbedingt **dnsPrefix** und **eMail-Adresse** anpassen...

HANDS ON 1.2: ACI PER QUICKSTART TEMPLATE



-
- Überblick: D365 Business Central custom App Typen und Betrieb im Container
 - Hands on 1: Azure Container Instance mit D365 BC
 - Hands on 2: API Client gegen BC verbinden (Connect App)
 - Hands on 3: Container konfigurieren und Verhalten anpassen
 - Hands on 4: Eigenes Business Central Image erzeugen
-

- ▶ Dokumentation zur API unter [↑https://docs.microsoft.com/en-us/dynamics-nav/fin-graph/](https://docs.microsoft.com/en-us/dynamics-nav/fin-graph/)
- ▶ Über einfache REST Requests Daten erzeugen, lesen, ändern und löschen
- ▶ Komplexe Aktivitäten über Bound Actions aufrufen (z.B. ein Buch.-Blatt buchen)

- ▶ Einfacher Weg zur Generierung von Beispiel-REST-Aufrufen:
 - Visual Studio Code öffnen (ggf. vorher von [↑https://code.visualstudio.com/download](https://code.visualstudio.com/download) herunterladen)
 - Extension ALRunner suchen und installieren
 - Ctrl+Shift+P → ALRunner: Generate an API Client for Business Central, dann URL `https://<fqdn>/nav`, Benutzername und Passwort eingeben

- ▶ Wer im vorherigen Hands on nicht die LetsEncrypt-Parameter beim Erstellen des Containers angegeben hat oder es nicht geklappt hat: Ctrl+Shift+P → ALRunner: Go API on Azure!
 - Erstellt eine Azure Container Instance anhand des Template und generiert die Beispiel-REST-Aufrufe

HANDS ON 2: CLIENT GENERIEREN



-
- ▶ Überblick: D365 Business Central custom App Typen und Betrieb im Container
 - ▶ Hands on 1: Azure Container Instance mit D365 BC
 - ▶ Hands on 2: API Client gegen BC verbinden (Connect App)
 - ▶ Hands on 3: Container konfigurieren und Verhalten anpassen
 - ▶ Hands on 4: Eigenes Business Central Image erzeugen
-

- ▶ Im Azure Portal über **Container Instance** → **Containers** → **Connect** eine Verbindung starten
- ▶ Als **Start Up Command** das Eingabefeld auswählen und powershell eingeben
- ▶ `c:\run\prompt.ps1` ausführen (importiert die BC Cmdlets)

```
Windows PowerShell
Copyright (C) 2016 Microsoft Corporation. All rights reserved.

PS C:\> C:\Run\Prompt.ps1
Welcome to the NAV Container PowerShell prompt
```

- ▶ z.B. `Get-NAVServerConfiguration NAV` ausführen, um **aktuelle Konfig** anzuzeigen

- ▶ NAV Server Instanz im Container konfigurieren: OData Services deaktivieren

```
PS C:\Run> Set-NAVServerConfiguration -KeyName ODataServicesEnabled -KeyValue false NAV
WARNING: The new settings value will not take effect until you stop and restart the service.
PS C:\Run> Restart-NAVServerInstance NAV

ServerInstance : MicrosoftDynamicsNavServer$NAV
DisplayName    : Microsoft Dynamics NAV Server [NAV]
State          : Running
ServiceAccount : NT AUTHORITY\SYSTEM
Version        : 13.0.24623.24800
Default        : True
```

- ▶ OData Services aktivieren funktioniert identisch, nur -KeyValue true und anschließend ebenfalls **Restart**

HANDS ON 3: ODATA SERVICES (DE-)AKTIVIEREN

- ▶ Konfigurationseinstellungen beim **Start mitgeben** über sog. Docker environment variables wie z.B. `customNavSettings=0DataServicesEnabled=false`
- ▶ Alle Basis-Scripts im Container ([↑ https://github.com/microsoft/nav-docker](https://github.com/microsoft/nav-docker)) über **gleichnamige Dateien** in `c:\run\my` überschreibbar
 - Beim Start können .zip Dateien **heruntergeladen und in den Ordner kopiert** werden, z.B. `folders=c:\run\my=https://raw.githubusercontent.com/Azure/azure-quickstart-templates/master/101-aci-dynamicsnav/scripts/SetupCertificate.zip`
- ▶ Genauso z.B. eigene DLL Libraries oder andere notwendige Dateien beim Start in Container bringen: `folders:c:\temp=https://www.axians-infoma.de/logo.gif`
- ▶ Haben wir vorhin bereits verwendet, um obenstehendes Script für LetsEncrypt-Zertifikate einzubinden (folders-Parameter) und API Services zu aktivieren (customNavSettings)

-
- Überblick: D365 Business Central custom App Typen und Betrieb im Container
 - Hands on 1: [Azure Container Instance](#) mit D365 BC
 - Hands on 2: [API Client](#) gegen BC verbinden (Connect App)
 - Hands on 3: Container konfigurieren und Verhalten anpassen
 - Hands on 4: Eigenes Business Central Image erzeugen
-

HANDS ON 4

EIGENES BUSINESS CENTRAL IMAGE ERZEUGEN

- ▶ Azure Container Registry: [↑https://azure.microsoft.com/de-de/services/container-registry/](https://azure.microsoft.com/de-de/services/container-registry/)
- ▶ Erlaubt über Azure CLI Erzeugen eigener Images ohne selbst Docker installiert zu haben
- ▶ Registry erzeugen

```
az acr create --name testregtfe --resource-group test --sku Standard
```

- ▶ Dockerfile erzeugen. Dafür Editor mit Kommando code starten

```
FROM microsoft/bcsandbox:de
SHELL ["powershell", "-Command", "$ErrorActionPreference = 'Stop';
$ProgressPreference = 'SilentlyContinue';"]
RUN mkdir c:/run/my
COPY AdditionalSetup.ps1 c:/run/my
```

- ▶ AdditionalSetup.ps1 erzeugen. Zunächst mit Kommando touch AdditionalSetup.ps1 erstellen, dann mit code AdditionalSetup.ps1 öffnen

```
Set-NAVServerConfiguration -KeyName ApiServicesEnabled -KeyValue true
NAV
Restart-NAVServerInstance NAV
```

► Image erzeugen

```
az acr build --registry testregtfe --os Windows --image myownbc:v1 .
```

► Container starten

```
az container create -g test --name testcont --image  
testregtfe.azurecr.io/myownbc:v1 \  
--os-type Windows --cpu 4 --memory 7 --ports 80 443 7048 7049 8080 \  
--dns-name-label tst125tfe \  
-e accept_eula=Y username=admin password=Passw0rd*123 \  
ContactEMailForLetsEncrypt=tobias.fenster@axians-infoma.de \  
PublicDnsName=tst125tfe.westeurope.azurecontainer.io \  
folders=c:\\run\\my=https://github.com/Azure/azure-quickstart-  
templates/raw/master/101-aci-dynamicsnav/scripts/SetupCertificate.zip
```

HANDS ON 4: EIGENES BUSINESS CENTRAL IMAGE ERZEUGEN



WELCHE FRAGEN DARF ICH BEANTWORTEN?

Kontakt Daten:
tobias.fenster@axians-infoma.de
+49 731 1551-964

