COSMO CONSULT
Business-Software für Menschen

Running multiple NAV/BC containers on an Azure VM

Tobias Fenster

What problem are we solving?
Introduction to the scenario

- Docker containers allow running **multiple versions / CUs** of Business Central on the **same VM**
- Docker containers have a **much lower resource overhead** than full VMs
- Creating / starting and stopping / deleting containers is **a lot quicker** than full VMs

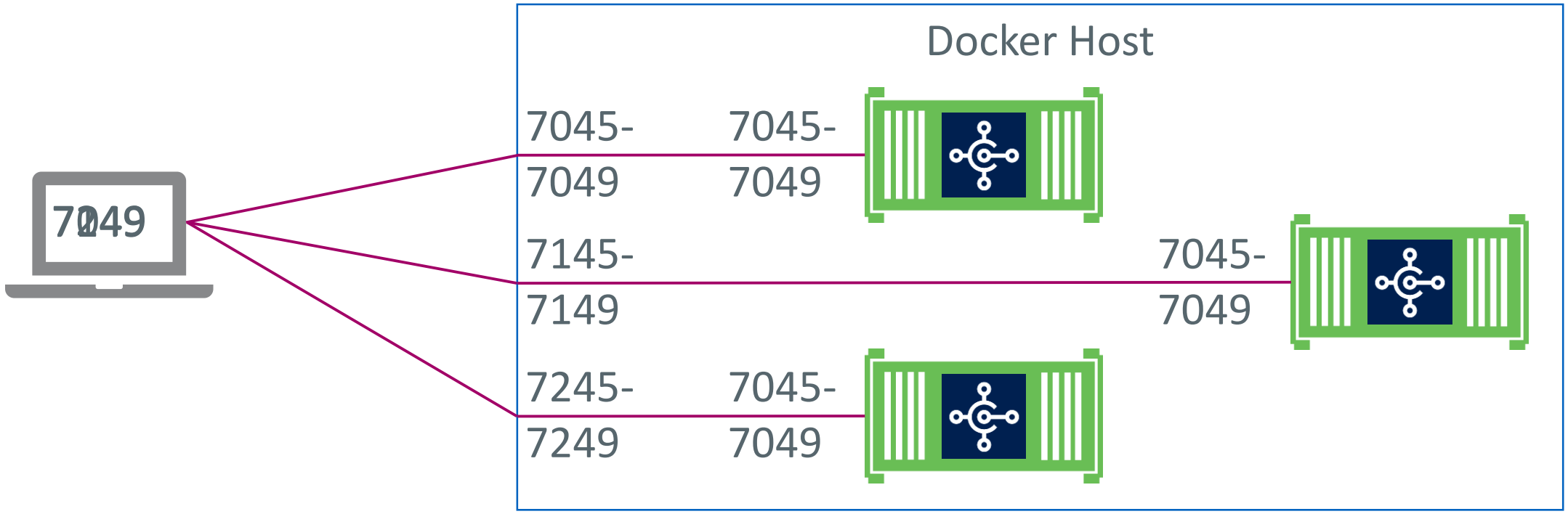
➔ You want to run **multiple containers on the same VM**

➔ But how can you connect your development / test / etc. machines to those containers?

Demo 1: Standard-Konfiguration

How can we solve that problem?
Mapping ports

- Run your containers and **map their ports** to different host ports



Demo 2: **Port-Mapping**

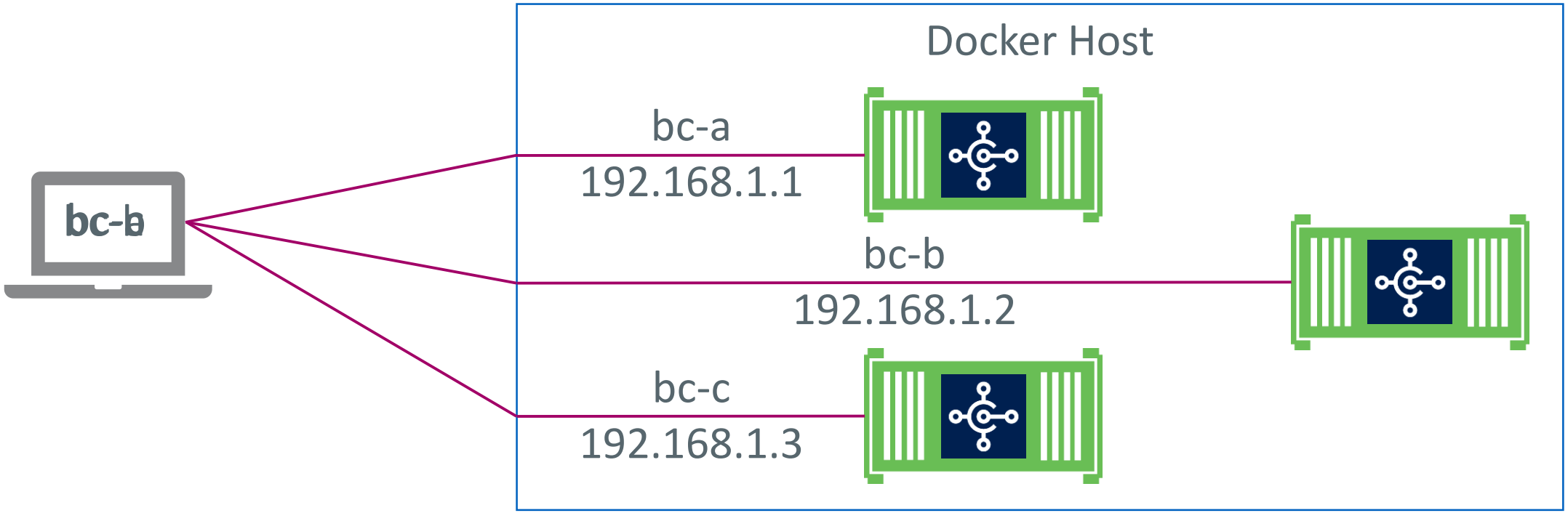
How can we solve that problem?
Mapping ports – the good and the bad

- Good:
  - **Easy to connect** to from the client (if you know the right port)
- Bad:
  - Always need to determine **which ports are free** for the next container
  - Don’t forget 80, 443, 8080
  - Need to open ports on the **firewall** of the VM
  - On Azure that becomes two firewalls (VM and Azure networking)

➔ Possible but somewhat **complicated and error prone**

How can we solve that problem?
Transparent networking

- Run every container with **its own IP (and name)**



Demo 3: **Transparent networking**

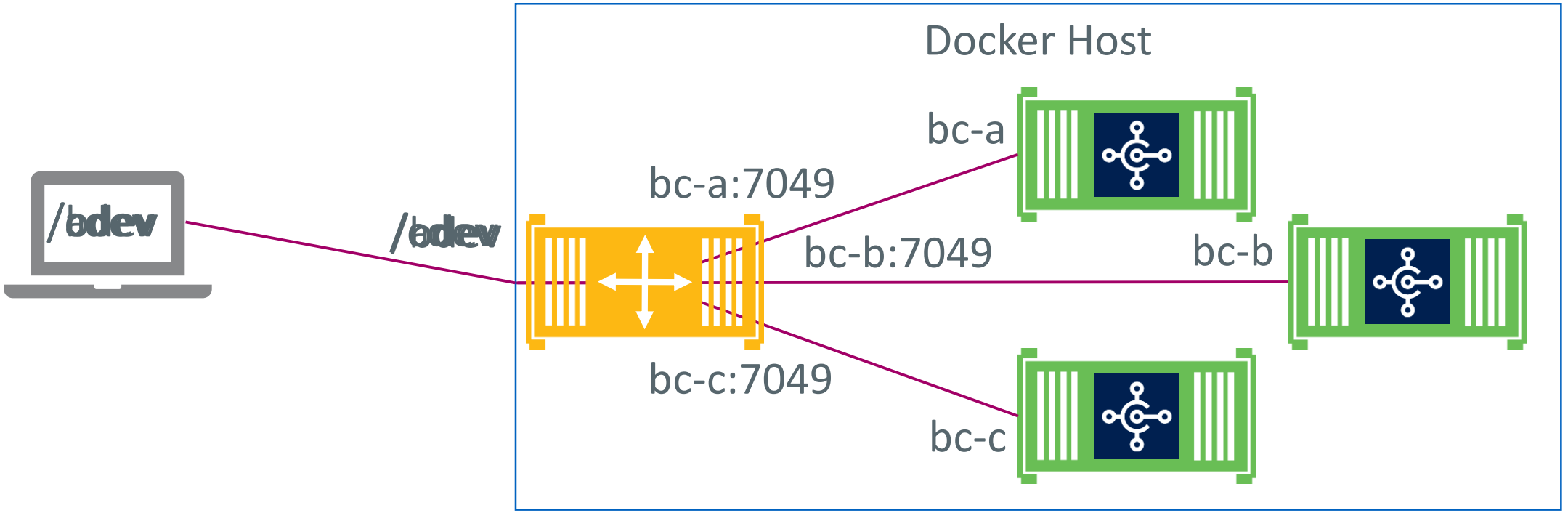
How can we solve that problem?
Transparent networking – the good and the bad

- Good:
  - **Easy to connect** to from the client (you only need the name)
  - Creating a new container **is easy**
- Bad:
  - Needs to be **allowed** on your network
  - Needs a **specific setting** on your hypervisor (MAC address spoofing)
  - Not directly possible on Azure VMs

➔ Good solution for on prem if allowed but not for Azure

How can we solve that problem?
Reverse proxy

- Run your containers **behind a reverse proxy**



Demo 4: **Reverse proxy**

How can we solve that problem?
Reverse proxy – the good and the bad

- Good:
  - **Easy to connect** to from the client (you only need the name)
  - Creating a new container **is easy** and it **works on Azure**
  - You only need **one entry point per service** in the firewalls
- Bad:
  - **One more component** to set up and maintain
  - Non-TCP-traffic needs more work (RTC and SQL/finsql) → traefik 2.0 supports that, but I haven't tested yet
  - URLs returned from SOAP and REST endpoints **not correct**

→ Good solution for both worlds, especially with automated setup

How can we solve that problem?
Reverse proxy – the details

- Implemented using **traefik** (https://traefik.io/)
  - Cloud-native, container-native reverse proxy
  - **Easy to set up and run**, e.g. integrated LetsEncrypt support
  - Picks new containers up by **checking their labels**
- Regex-based rules for the **mapping**, e.g.
  - https://myvm.westeurope.cloudapp.azure.com/bc-arest/* maps to http://bc-a:7048/NAV/OData/*
  - https://myvm.westeurope.cloudapp.azure.com/bc-a/* maps to http://bc-a:80/bc-a/*

How can we solve that problem?
Reverse proxy – the details

- **Additional config** for the Business Central container:
  - Set PublicODataBaseUrl, PublicSOAPBaseUrl, PublicWebBaseUrl and PublicDnsName so that Business Central knows what it is called from the outside
  - Set WebServerInstance to a different name as it otherwise insists on redirecting to /NAV
  - Health check needs to be different: Traefik only picks up healthy containers but for the regular health check to work, traefik routing needs to be in place…
- Traefik needs a **setup file** called traefik.toml

How can we solve that problem?
Reverse proxy – the details

- Integrated into
  - aka.ms/getbc and related **Azure ARM templates** with a „Use Traefik“ toggle
  - **navcontainerhelper** with -useTraefik
- Base setup needed
  - New navcontainerhelper cmdlet **Setup-TraefikContainerForNavContainers**

Demo 5: Let‘s check the details

ANY QUESTIONS?

Thanks for your interest!