axians

Product

Docker on Windows and BC on Docker deep dive

VINCI ENERGIES

Tobias Fenster

CTO at Axians Infoma
Microsoft MVP for Business Applications

@tobiasfenster
https://techblog.axians-infoma.com

Agenda

- Introduction to Docker on Windows
- Deep dive into Business Central on Docker
- Lunch 13-14, other breaks on the fly

axians

Hands on how to

- We are part of the Axians company network of ICT companies
- Axians has agreed to sponsor your Azure VMs, so the hands on is powered by Axians
- You have connection info in your inbox
  - 1 big (16 core / 64 GB), 1 small (2/8) VM
- For every hands on, you will get a link to the commands and expected results
- VMs will be deleted after the workshop

Part 1 – **Docker on Windows**

Before we start

A word about editions and isolation

- Docker community edition (CE): Latest release, open source, runs on Windows 10 with Hyper-V isolation (process comes with 1809)
- Docker enterprise edition (EE): More stable, a bit behind, runs on Windows Server, supports both isolation types
  - Windows Server comes with EE basics for free, paid EE contains a lot of tooling around Docker for professional production usage
- Process isolation has minimal overhead, can only run on „matching“ hosts (backward compatible from Win Server 2019 on)
- Hyper-V isolation creates a „mini VM“ for running non matching containers

Installing Docker on Windows Server

- 3 part process
  - Enable container role (manually or automatically in the next part)
  - Install DockerMsftProvider, a package provider through OneGet
  - Install Docker (enable container role if necessary)
- Restart machine because of the Windows feature “Containers”
- Run sample container

https://ve.link/t2mre

Working with Docker on Windows Server

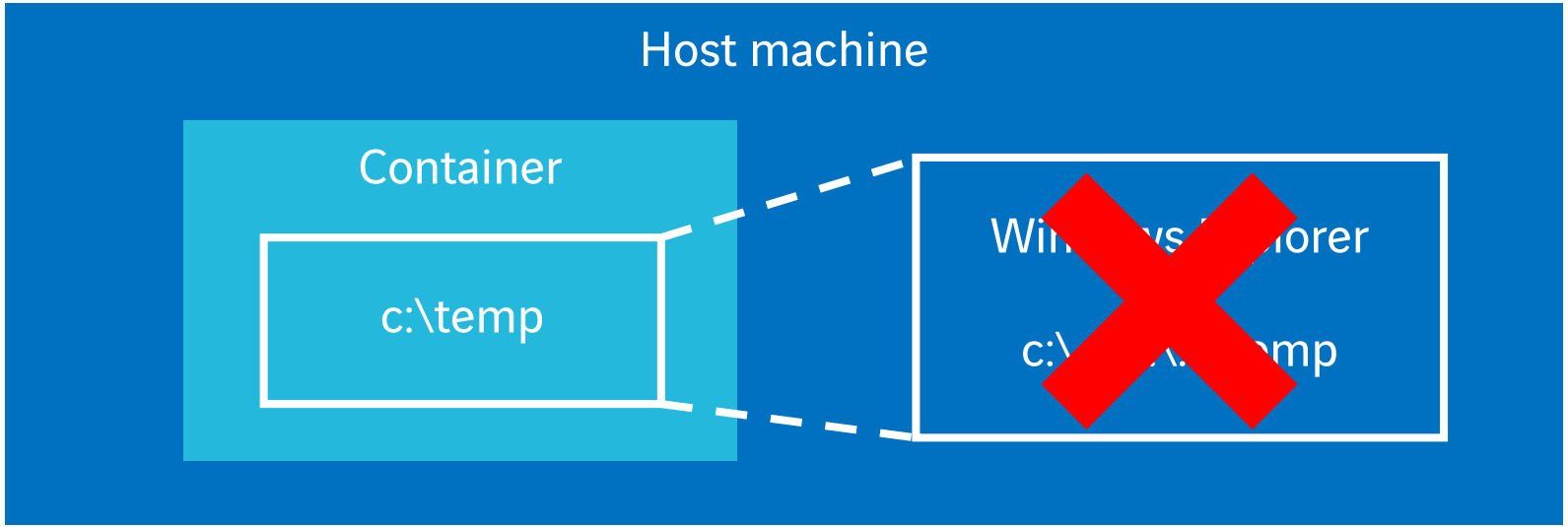
The basics of container handling

- Show running and all containers
- Create a container in interactive mode
- Show resource consumption and logs
- Get a PowerShell session inside a container
- Stop and remove containers, remove images

https://ve.link/wivk4

Working with Docker on Windows Server

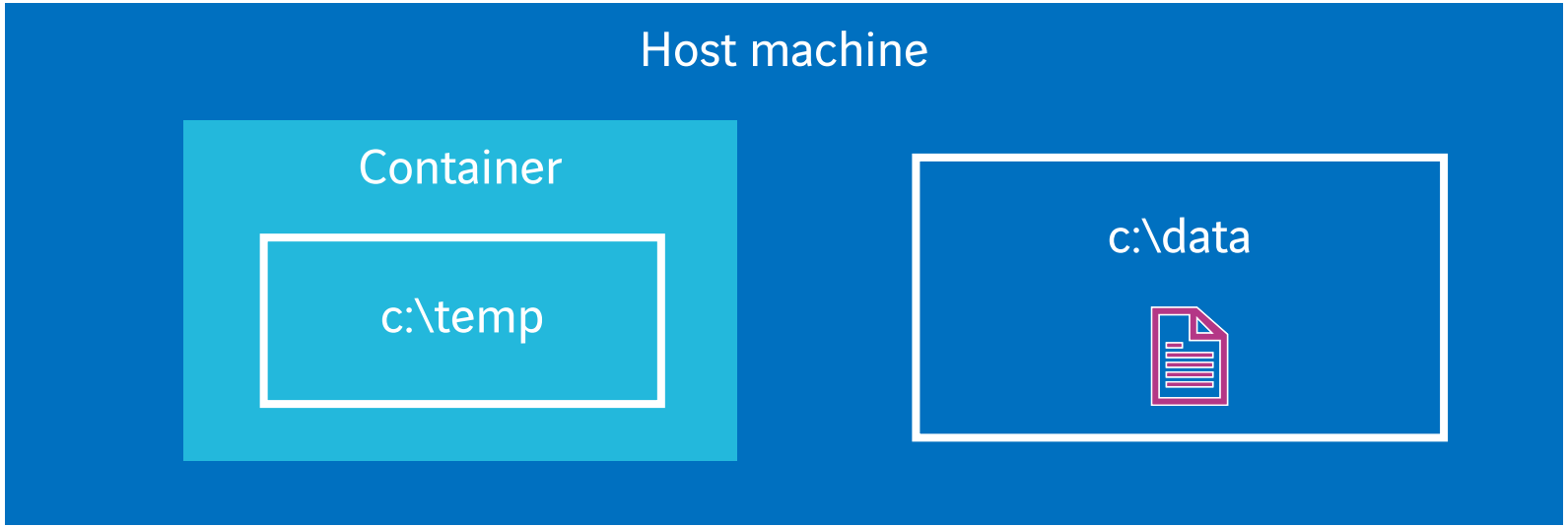
File handling and volumes



Standard fs setup: nothing configured

Working with Docker on Windows Server

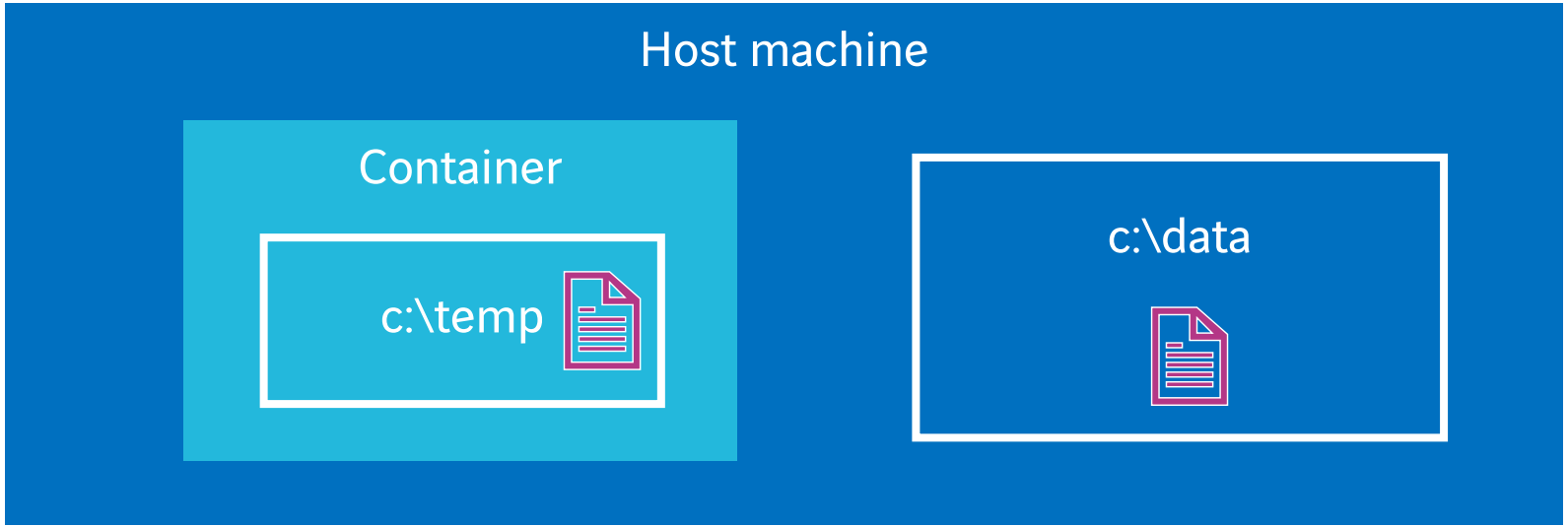
File handling and volumes



Standard fs setup: nothing configured. Use docker cp to copy files

Working with Docker on Windows Server

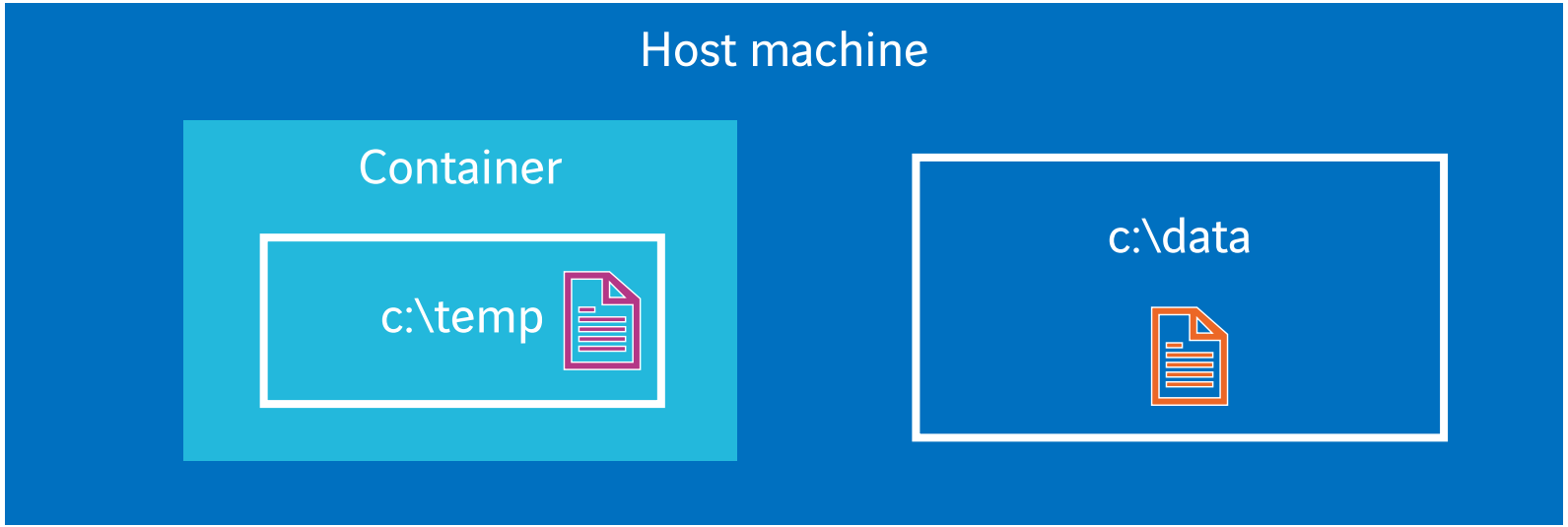
File handling and volumes



Standard fs setup: nothing configured. Use docker cp to copy files

Working with Docker on Windows Server

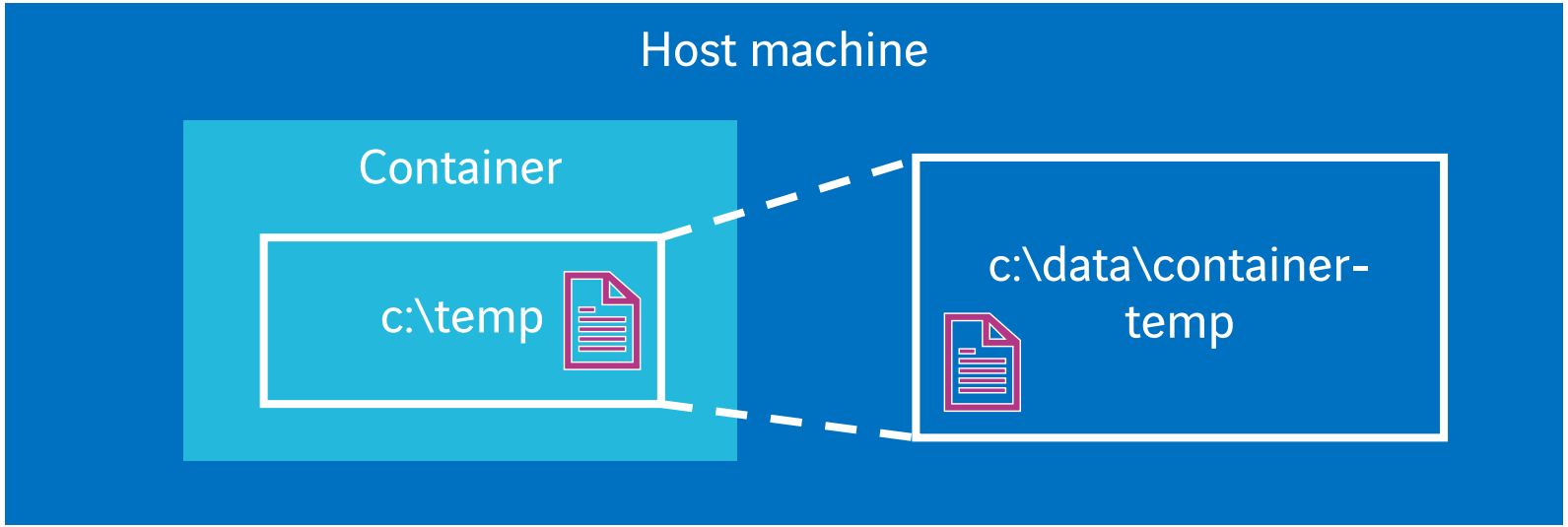
File handling and volumes



Standard fs setup: nothing configured. Use docker cp to copy files

Working with Docker on Windows Server

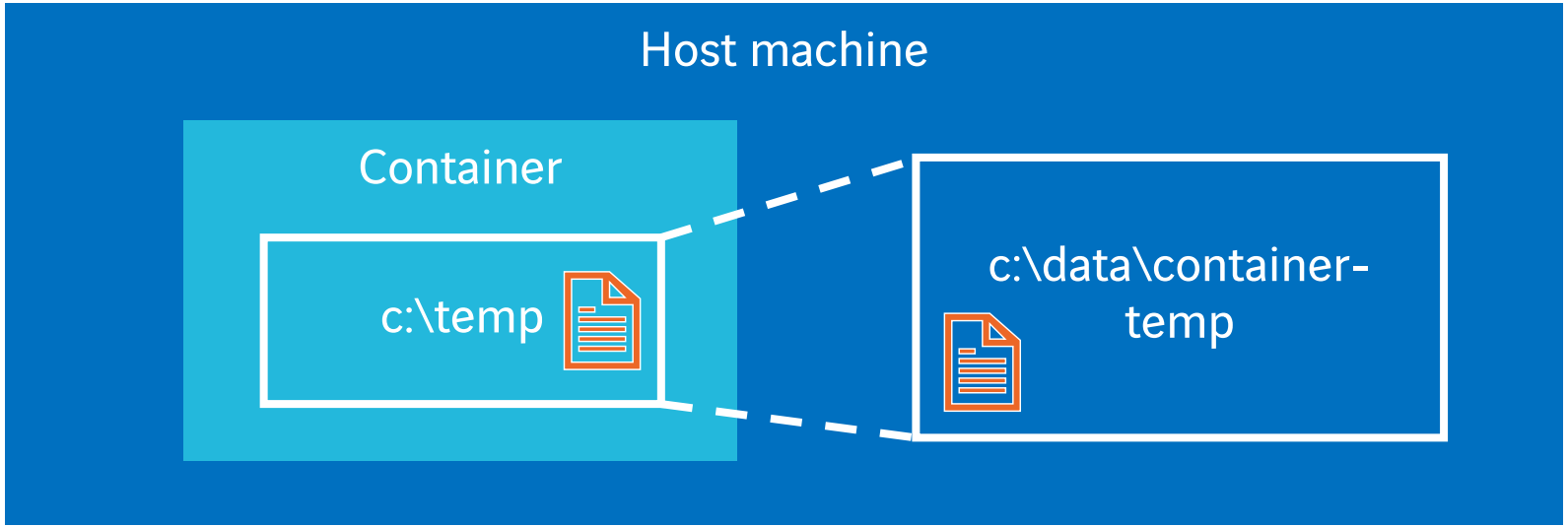
File handling and volumes



fs setup with a volume mapping, e.g. -v c:\data\container-temp:c:\temp

Working with Docker on Windows Server

File handling and volumes



fs setup with a volume mapping, e.g. -v c:\data\container-temp:c:\temp

Working with Docker on Windows Server

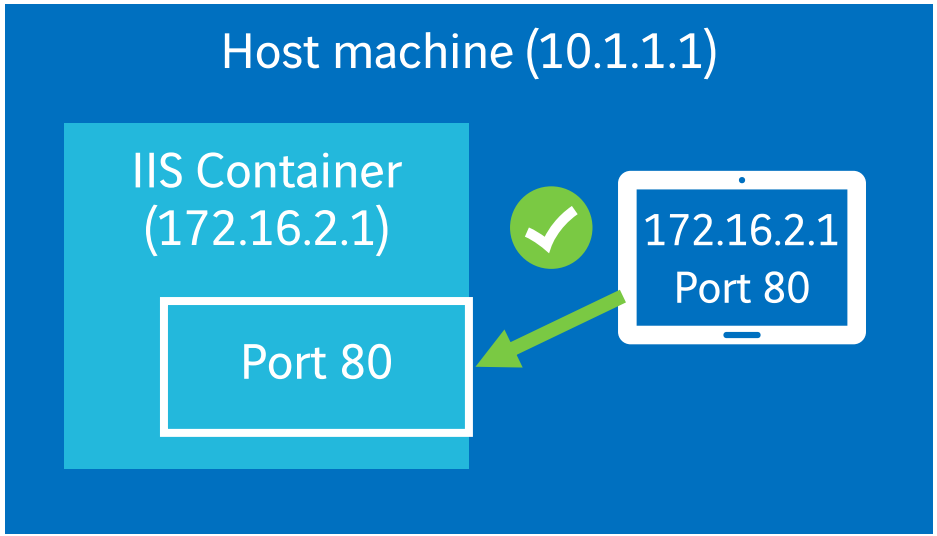
File handling and volumes

- Two options for file sharing between host and container:
  - Command docker cp allows copying files, that means afterwards you have two identical but unrelated files → works anytime
  - Parameter –v for volumes allow sharing folders between host and container (currently only possible for empty target folders in the container, but will improve with Server 2019) → can only be set up on startup

https://ve.link/1njyn

Working with Docker on Windows Server

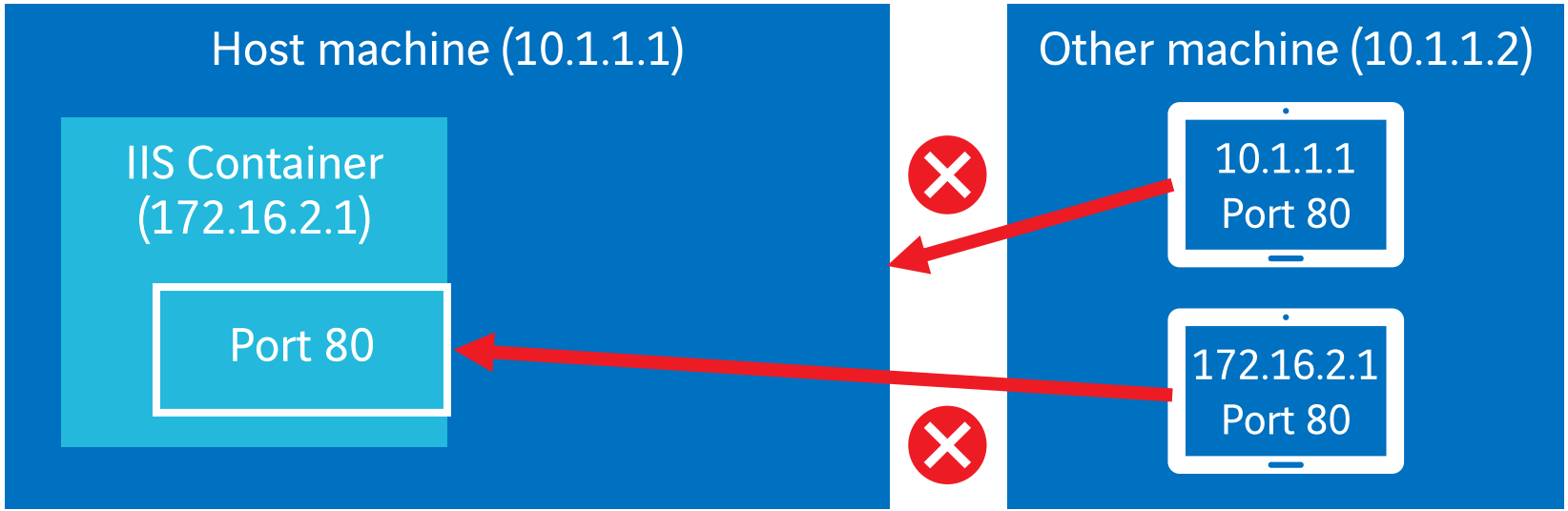
Networking



Standard network setup: NAT

Working with Docker on Windows Server

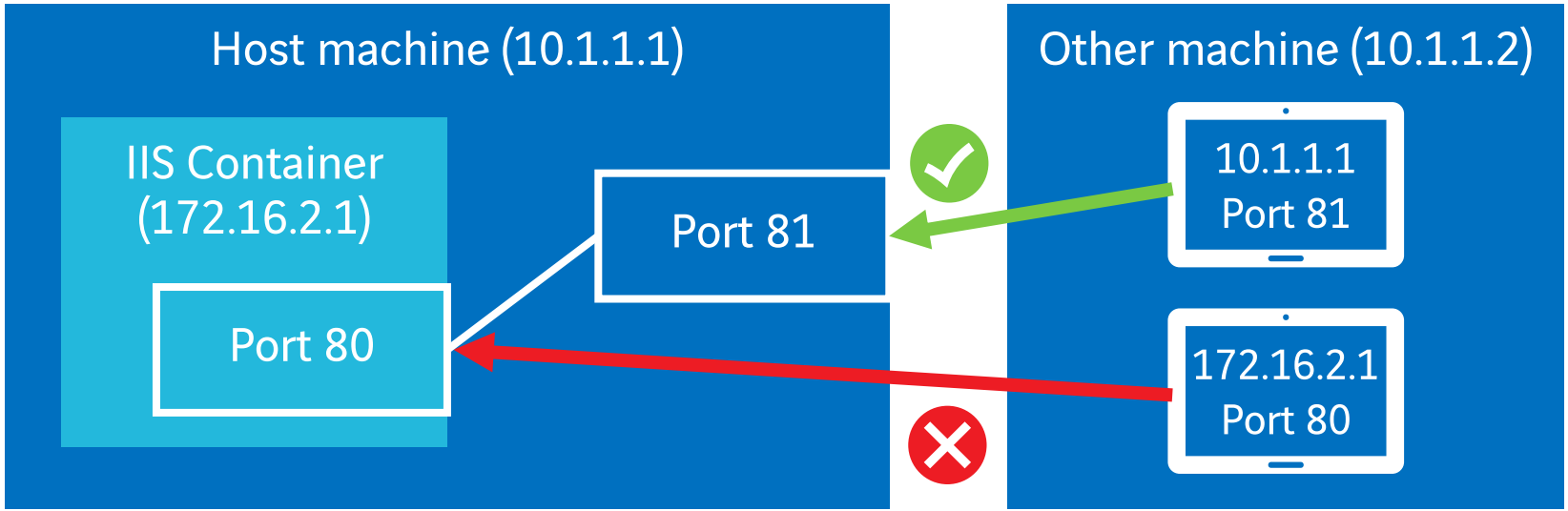
Networking



Standard network setup: NAT

Working with Docker on Windows Server

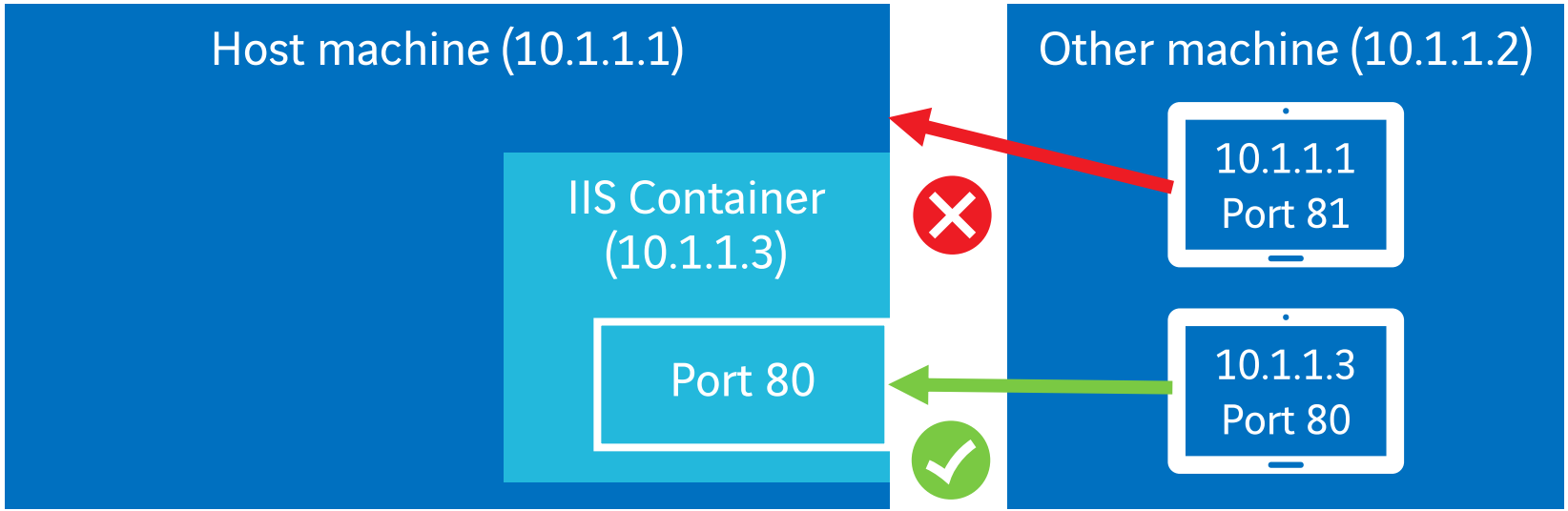
Networking



Standard network setup with port mapping, e.g param -p 81:80

Working with Docker on Windows Server

Networking



Transparent network setup: host and container “share” the network adapter

Working with Docker on Windows Server

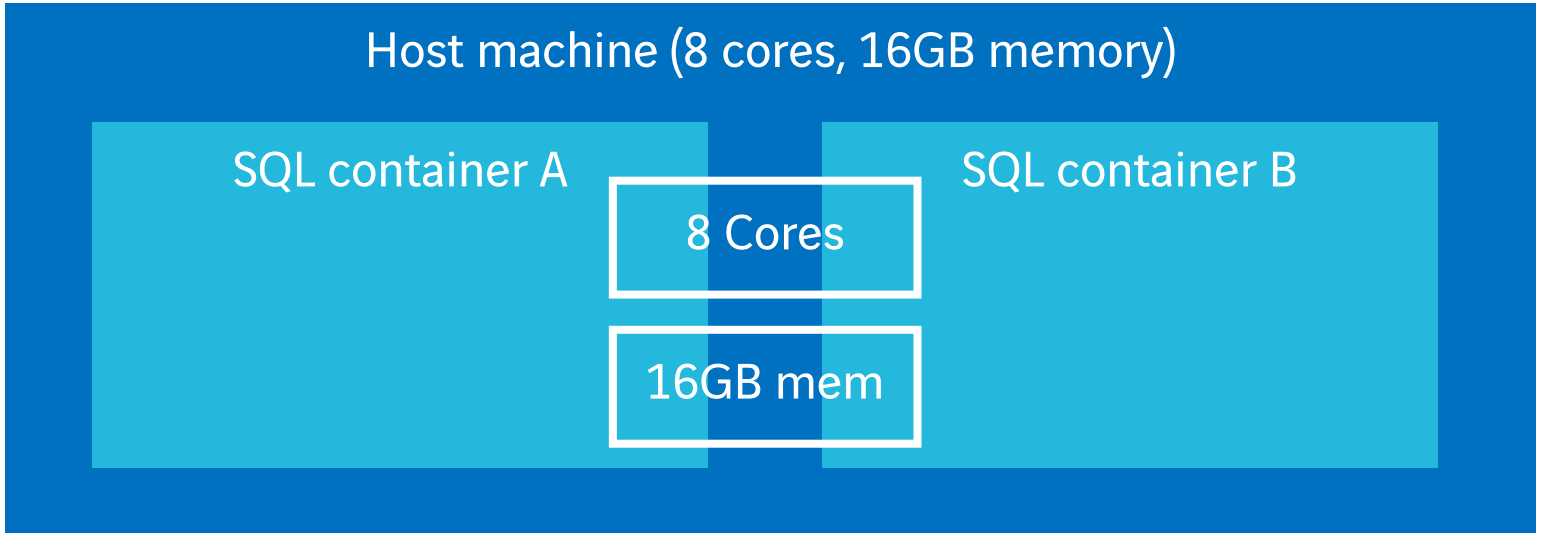
Networking

- Three options for network connections to the container:
  - Do nothing: Default NAT allows connections only from the host
  - Port mapping of 1-n ports on the container to 1-n possibly different ports on the hosts
  - Sharing the network through transparent config gets a dedicated IP (static or dynamic) for every container and makes it reachable on that network
- Can only be set up on startup
- We'll also see a remote connection to the Docker API

https://ve.link/9vp6z

Working with Docker on Windows Server

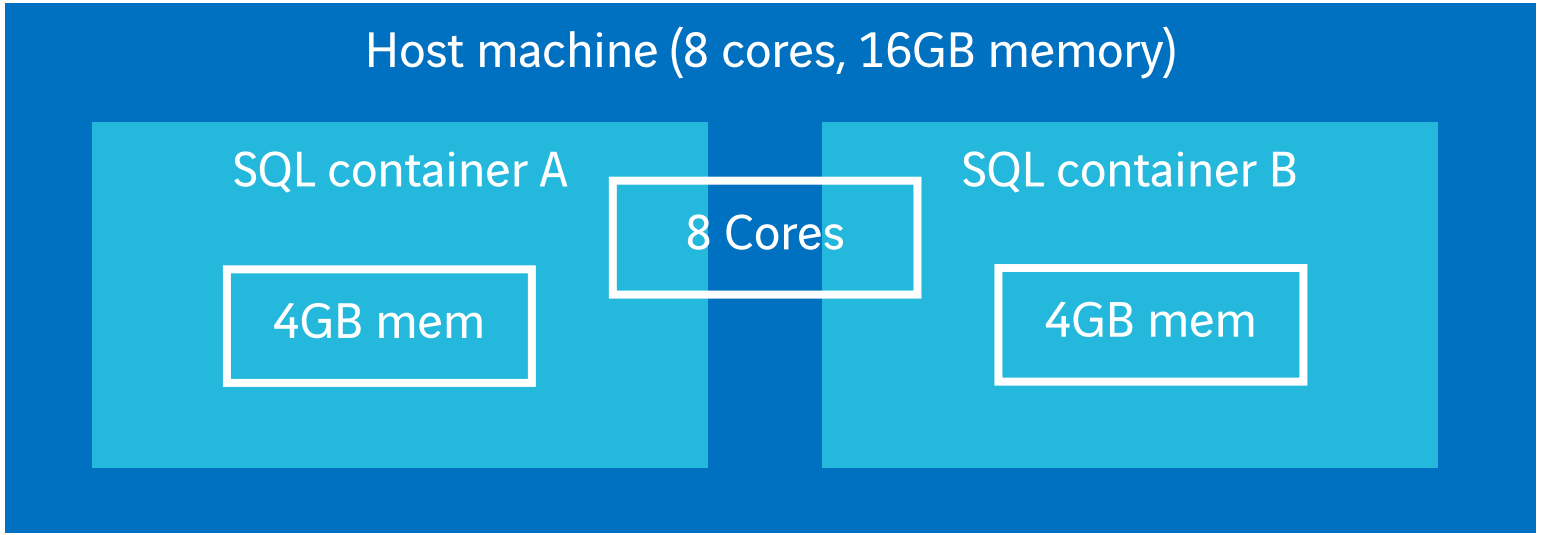
Resource limits



Standard resource setup: nothing configured

Working with Docker on Windows Server

Resource limits



Specific resource setup: limits are configured, e.g. -m 4g

Working with Docker on Windows Server

Resource limits

- Various options to limit CPU, memory and IO
  - See docker run --help
- Can only be set up on startup

https://ve.link/43dk9

Working with Docker on Windows Server

Dockerfiles

- A Dockerfile is like a script that describes the steps to take in order to create an image:
  - FROM = On what base does the start, e.g. microsoft/nanoserver
  - COPY = Copy file(s) into the image, e.g. an installer or sources
  - RUN = Run a command inside the image, e.g. building your app
  - CMD = Defines the default command when a container runs
  - EXPOSE = Defines on which port(s) a container has a listening process
  - SHELL = Defines the default shell to use for RUN commands
  - LABEL = Set descriptive metadata
  - See https://docs.docker.com/engine/reference/builder/ for full docs

Working with Docker on Windows Server

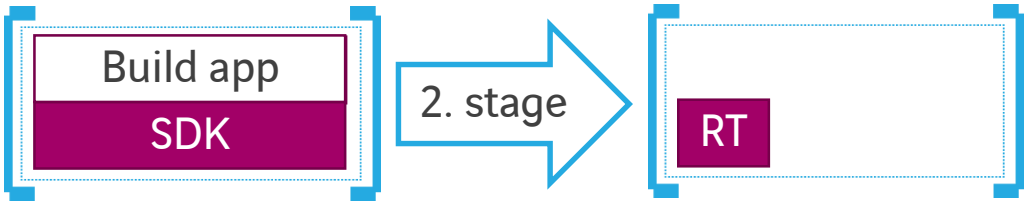
Dockerfiles

- Example 1: Create an Apache web server image by expanding the zip and install the .NET prereq using a silent installer
- Example 2: Create an image by building a custom solution where we have the sources
- Example 3: Use a multi-stage image to have an image as small as possible

Working with Docker on Windows Server

Dockerfiles

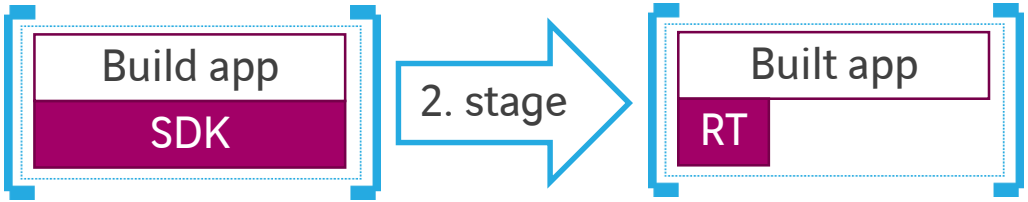
- Example 1: Create an Apache web server image by expanding the zip and install the .NET prereq using a silent installer
- Example 2: Create an image by building a custom solution where we have the sources
- Example 3: Use a multi-stage image to have an image as small as possible



Working with Docker on Windows Server

Dockerfiles

- Example 1: Create an Apache web server image by expanding the zip and install the .NET prereq using a silent installer
- Example 2: Create an image by building a custom solution where we have the sources
- Example 3: Use a multi-stage image to have an image as small as possible



Working with Docker on Windows Server

Dockerfiles

- Example 1: Create an Apache web server image by expanding the zip and install the .NET prereq using a silent installer
- Example 2: Create an image by building a custom solution where we have the sources
- Example 3: Use a multi-stage image to have an image as small as possible

https://ve.link/z4b8k

Working with Docker on Windows Server

Docker compose

- Docker compose is an additional tool, also provided by Docker Inc
- Allows defining multiple containers (called „services“) working together in an easy to read syntax
- Basically puts together multiple docker run commands in one file → „Infrastructure as Code“
- Example: Web application with SQL database backend

https://ve.link/v9b4t

Working with Docker on Windows Server

Additional notes

- Working with the Docker API works remotely as well, but needs configuration of the Docker host (including firewall)
  - Easiest, but unsecure way: Configure the daemon to listen on all IPs by setting the following in c:\programdata\docker\config\daemon.json:
    `{ "hosts": ["tcp://0.0.0.0:2375", "npipe://"] }`
  - Use it by setting `DOCKER_HOST=<host>:2375` on the client machine
  - How to do it properly: https://stefanscherer.github.io/protecting-a-windows-2016-docker-engine-with-tls/
- Check https://portainer.io for a container-based “Docker GUI” (easier to setup with Server 2019 as well, using named pipes)

Working with Docker on Windows Server

Upcoming improvements with Windows Server 2019

- Volume mounts are no longer symlinks and can target non-empty directories
- gMSAs no longer need to have the exact same name as the container and the hostname of the container
- Published ports are available on localhost
- Standard images are a lot smaller and a new image type “Windows” with more capabilities
- Networking improvements for Docker Swarm and easier API access
- Better LCOW support

Resources

Links to know and people to follow

- https://docs.microsoft.com/en-us/virtualization/windowscontainers/
- https://docs.docker.com/
- @Docker
- @EltonStoneman (dev advocate at Docker / Microsoft MVP)
- @stefscherer (Docker Caption / Microsoft MVP)
- @ManoMarks (director dev relations at Docker)

Working with Docker on Windows Server

Bonus topic

- Implement remote access to the Docker API
- File system binds work server side, docker cp client side

https://ve.link/9bnad

Part 2 – **BC on Docker**

Business Central on Docker

Basic structure as of now – very frequent improvements!

- Public repository https://github.com/microsoft/nav-docker
- Base of all: „generic“ image
  - FROM windowsservercore with .NET runtime 4.7.2 (from 1809 only windowsservercore as it then includes .NET 4.7.2)
  - Install SQL Server and IIS dependencies
  - Copy files from Run folder into the image
  - Download Report Builder and some utils
- Same for all types (dynamics-nav, bcsandbox, bconprem): „specific“ images W1 (called „base“ in bcsandbox) and local versions behave a bit different

Business Central on Docker

Basic structure as of now – very frequent improvements!

- W1 / base built FROM generic:
  - Download NAV DVD and .vsix (AL extension for VS Code)
  - Move the right version specific files (folders 70 to 130) in place
  - Call navinstall.ps1 which starts SQL and IIS and „installs“ NAV / BC with dependencies, restores country independent CRONUS database and generates a Service Tier
- Country specific built FROM W1 / base:
  - Uses importCountry.ps1 to restore country Cronus<lang> databases like CronusDK or CronusDE, run local installers and adjust Service Tier conf
  - Also generates AL symbols from NAV 2018 onward

Business Central on Docker

Basic structure as of now – very frequent improvements!

- bcsandbox-master (preview builds available through collaborate) work the same afaik
- start.ps1 is called on docker run and calls mainly the following, depending on params and whether it is the first start of the container and whether the DNS name has changed:

  Download „folders“ -> navstart.ps1 -> SetupVariables.ps1 -> start SQL and IIS -> SetupDatabase.ps1 -> SetupConfiguration.ps1 -> SetupWebClient.ps1 / SetupWebClientConfiguration.ps1 -> SetupFileShare.ps1 -> Setup*Users.ps1 -> AdditionalSetup.ps1 -> AdditionalOutput.ps1 -> MainLoop.ps1

Business Central on Docker

Examples of parameter usage

- Example 1: Custom NAV settings / Web settings
- Example 2: Use Windows authentication and enable ClickOnce
- Example 3: Connect to an external SQL Server with bconprem
  - SQL and NAV in separate containers
  - Expand the sample to add a staging and a test environment
  - All defined using compose
- Good way to find all possible parameters: Check SetupVariables.ps1

https://ve.link/85fwy

Business Central on Docker

Scripts overwriting

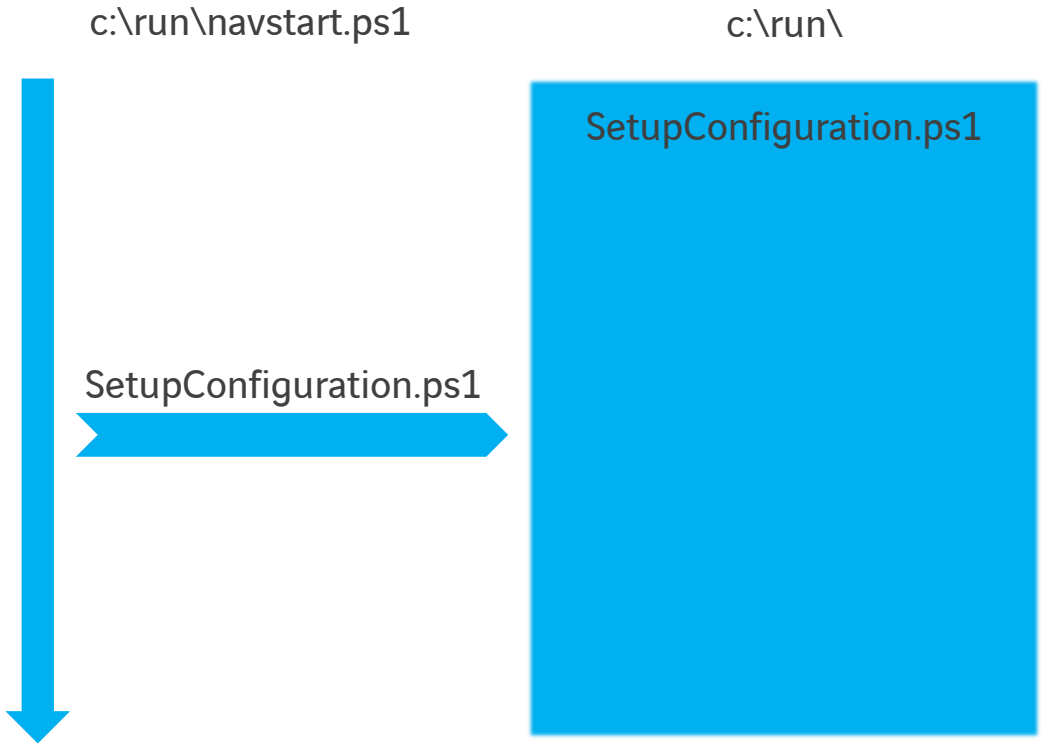
- Before calling standard scripts in c:\run, c:\run\my is checked for a script with the exact same name
- If the script exists in c:\run\my, it is called instead of the standard
- Make sure to call the standard script as well if necessary, before / during / after your lines:

```
. (Join-Path $runPath $MyInvocation.MyCommand.Name)
```

- Other example: Persist the database from a container to the host and re-attach it on the next startup - https://github.com/tfenster/nav-docker-samples/tree/simple-volume-persistence, overwrites SetupDatabase.ps1

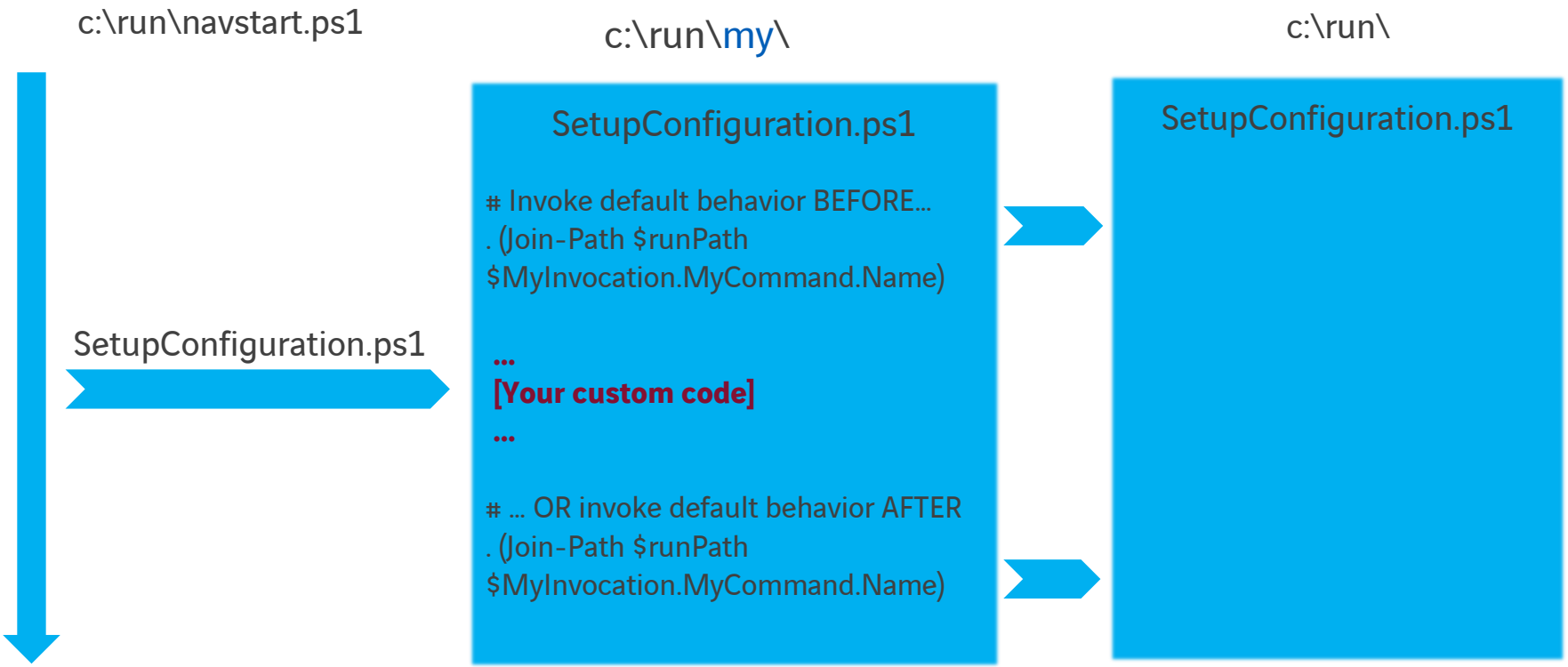
Business Central on Docker

Scripts overwriting



Business Central on Docker

Scripts overwriting



Business Central on Docker

Getting files into the container

- Docker cp and volumes (including c:\run\my) work fine
- Example 1: Special folder c:\run\Add-ins
- Example 2: Download script on startup
  - Environment param „folders“ download files and puts them inside the container, e.g. -e folders=“c:\temp=https://files.axians-infoma.de/devlicense.flf“ → No need to have it locally on the container host
  - Can be used for public scripts e.g. on GitHub or any other type of file
  - Example is activating the API endpoint by calling a codeunit

https://ve.link/t7ycb

Business Central on Docker

Custom images

- Almost everything can be changed on the fly, not too many scenarios where a custom image is strictly necessary
- Still can be nice for some cases, e.g.
  - Do time consuming tasks like e.g. installing additional PS modules
  - Add your own .bak and .dlls to have a „version image“
  - Make sure custom scripts are never changing
- Example: Create an image with activated API as seen through a very simple Dockerfile

https://ve.link/vunxo

Business Central on Docker

navcontainerhelper

- Collection of helper Cmdlets and Scripts to ease container usage mainly for NAV / BC development and devops
- Also the base for Freddy's CI/CD scripts and aka.ms/getbc and others
- No "magic", but extensive set of common use cases like
  - New-NAVContainer, Replace-NavServerContainer
  - Convert-ModifiedObjectsToAl
  - Compile-AppInNavContainer, Compile-ObjectsInNavContainer
  - Install-NavContainerApp, Publish-NavContainerApp
  - New-LetsEncryptCertificate, Renew-LetsEncryptCertificate
  - Convert-AlcOutputToAzureDevOps, ...

Business Central on Docker

navcontainerhelper

- Install with `install-module navcontainerhelper -force`
- Example 1: Run your first container
- Example 2: Compile an extension in a container
- Example 3: Publish the extension to a container

https://ve.link/g38uu

Resources

Links to know and people to follow

- https://blogs.msdn.microsoft.com/freddyk/ and @freddydk
- https://github.com/Microsoft/nav-docker/
- https://github.com/Microsoft/navcontainerhelper


axians
Thanks a lot for being in this workshop!
Any questions? You can reach me at:
@tobiasfenster / tobias.fenster@axians-infoma.de