COSMO CONSULT

Business-Software for People

Docker on Windows 101 and BC on Docker deep dive

Antwerp, Nov 19/20 2019

Tobias Fenster

INTRODUCTION AND AGENDA

Tobias Fenster

CTO at COSMO CONSULT Group

Dual Microsoft MVP for Business Applications and Azure

@tobiasfenster

tobiasfenster.io

tobias.fenster@cosmoconsult.com

1. Introduction to Docker on Windows with 7+1 labs
2. Lunch break 12:30-13:30, 1st floor, other breaks on the fly
3. Deep dive into Business Central on Docker with 5 labs

HOW-TO FOR THIS WORKSHOP

- You should have connection information in your inbox for two VMs
  - 1 big (16 cores / 64 GB) and 1 small (2 cores / 8 GB)
- For every lab, you will get a link to commands and expected results through a page where you can also indicate when you start and finish the lab: http://bit.ly/td19-vote
- That documentation will still be available after the workshop and you will get all slides
- The VMs will be deleted after the workshop!
- Please feel free to always ask questions right away or raise your hand when you have problems with one of the labs

HAVE FUN!

QUICK INTRO TO docker

- What is Docker? Leading cross platform software container environment

- What is a Docker container and a Docker image?
  - An image is a template with the minimum amount of os, libraries and application binaries needed
  - A container is an instance of an image with an immutable base and it's changes on top
  - A container is NOT a VM, you especially don't have a GUI and nothing you can connect to with RDP!
- The main components are the Docker engine, which does the actual work and the Docker client, which is used to communicate with the engine

QUICK INTRO TO docker

- What is a Docker host? The (physical or virtual) machine where the containers are running
- What is a Docker registry? A place where you and others can upload (push) and download (pull) images
- Why Docker?
  - Easy way to create deployments / configuration in a very stable and reliable way (no "works here", helps a lot to avoid gaps between dev and ops)
  - Better resource usage than in VMs, especially because there is no guest os as the host kernel is directly used
  - Big ecosystem of readily available images, primarily on Docker Hub

PART 1 – DOCKER ON WINDOWS

LAB 1: 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 (enables container role if necessary)
- Restart machine because of the Windows feature “Containers”
- Run sample container

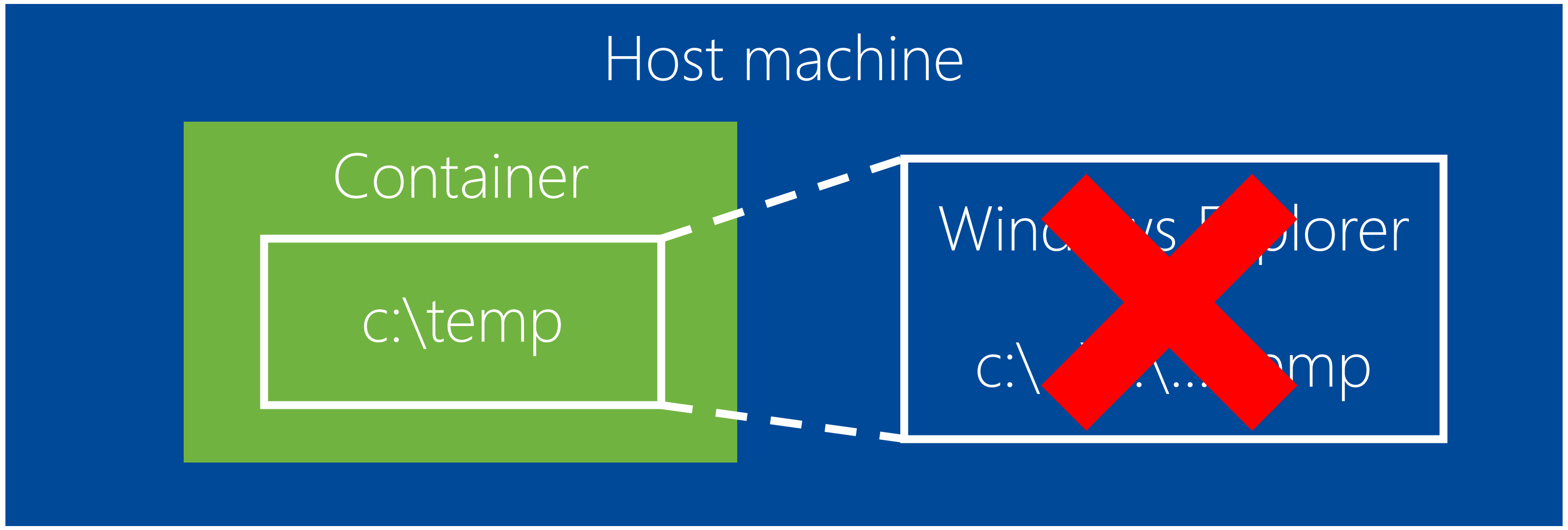
- Goal: The Docker service is running and you have validated that it works using the sample container

LAB 2: THE BASICS OF CONTAINER HANDLING

- Create a container in interactive mode
- Show running and all containers
- Show resource consumption and logs
- Get a cmd session inside an already running container
- Inspect the configuration of a container
- Stop and remove containers
- Show and remove images
- Give your container a name and reference it that way

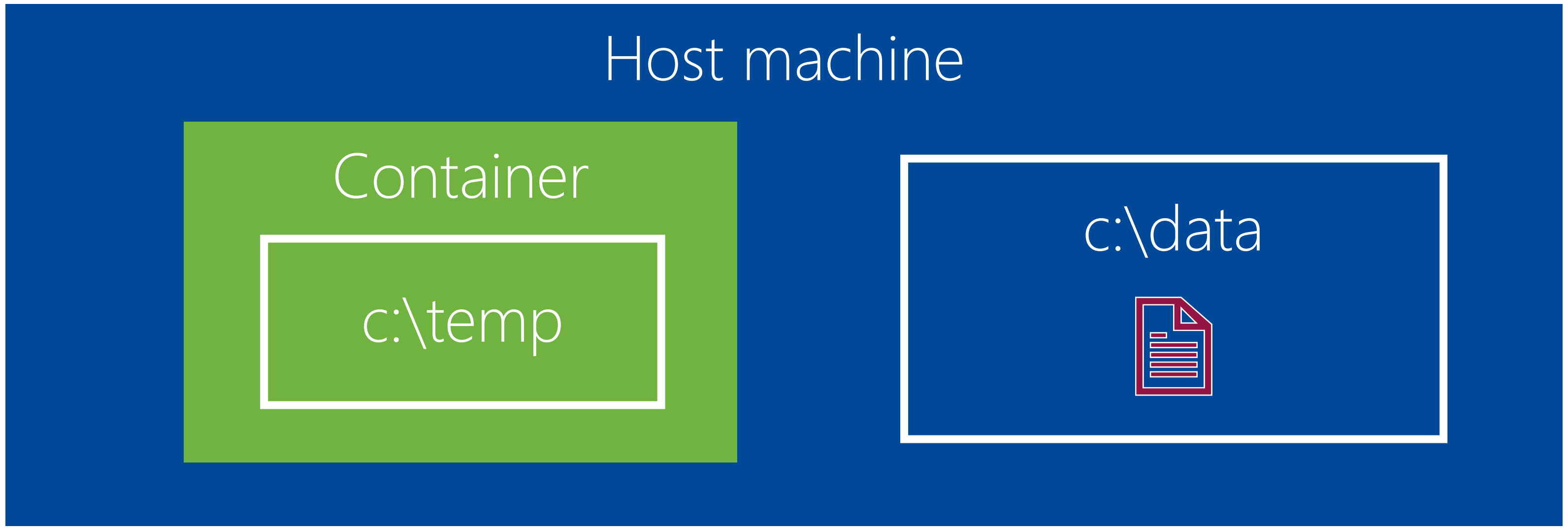
- Goal: You know the basic commands to work with containers

LAB 3: FILE HANDLING AND BIND MOUNTS



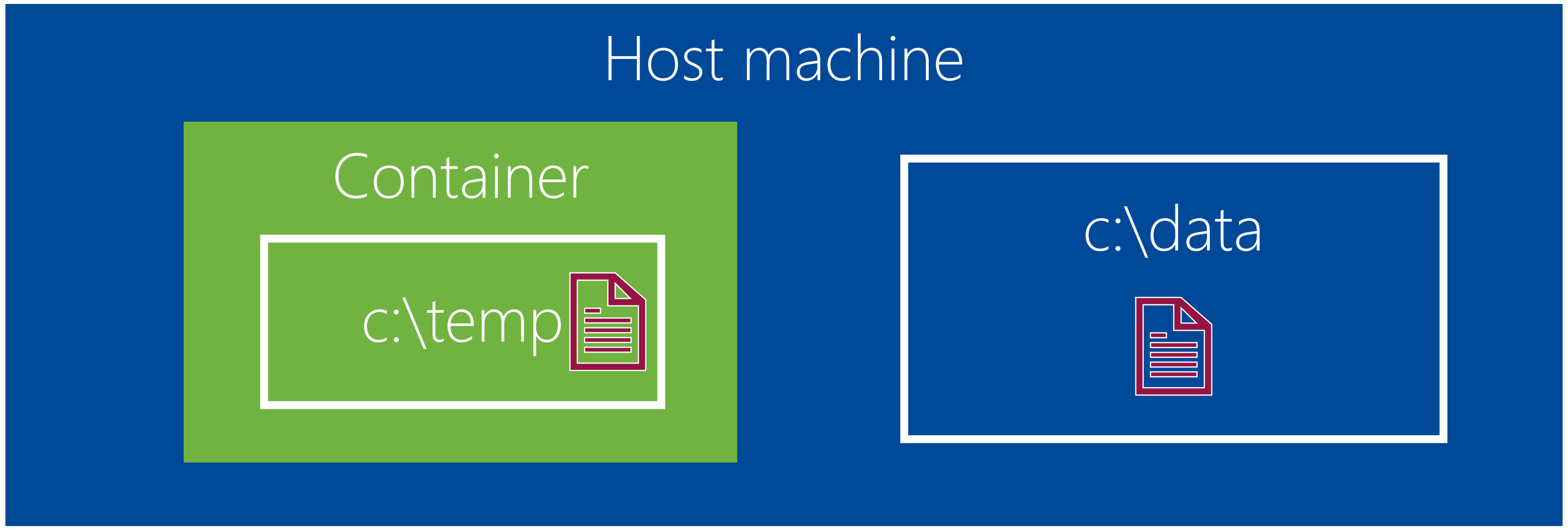
Standard filesystem setup: nothing configured

LAB 3: FILE HANDLING AND BIND MOUNTS



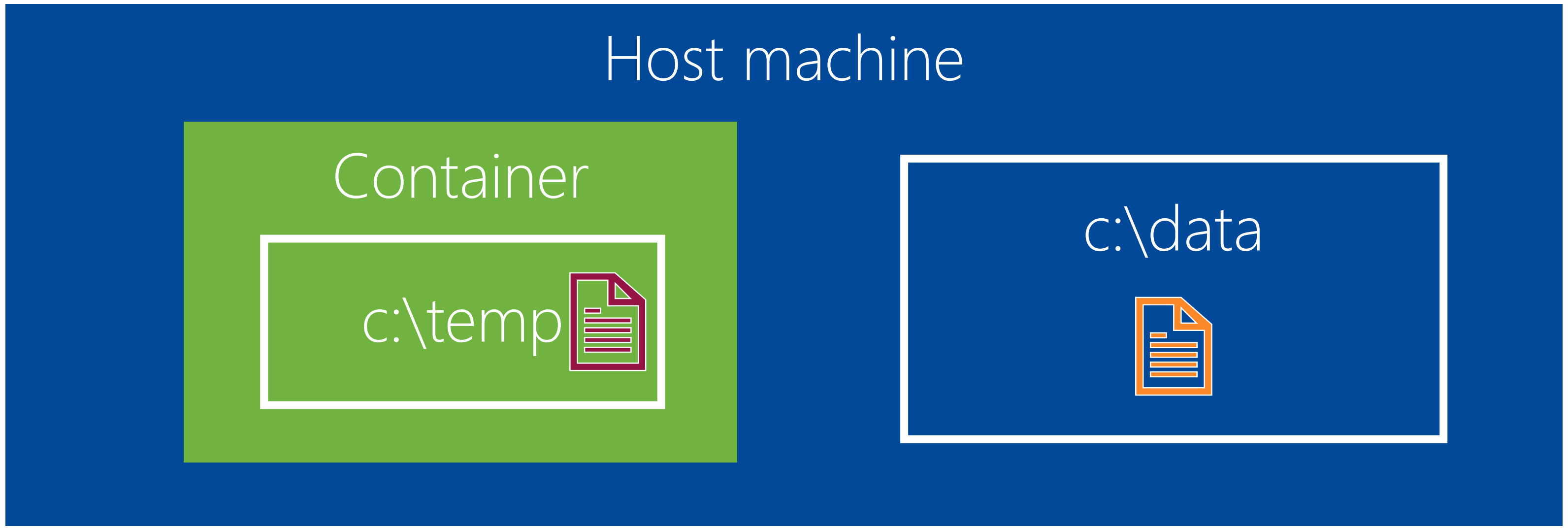
Standard filesystem setup: nothing configured. Use `docker cp` to copy files

LAB 3: FILE HANDLING AND BIND MOUNTS



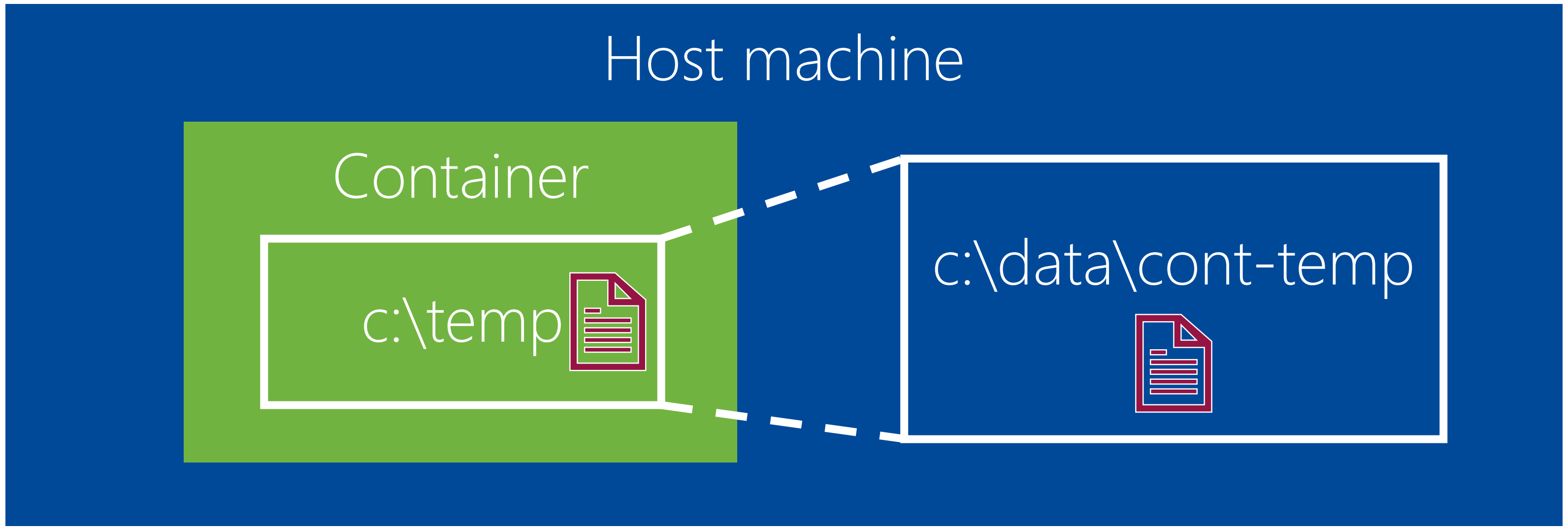
Standard filesystem setup: nothing configured. Use `docker cp` to copy files

LAB 3: FILE HANDLING AND BIND MOUNTS



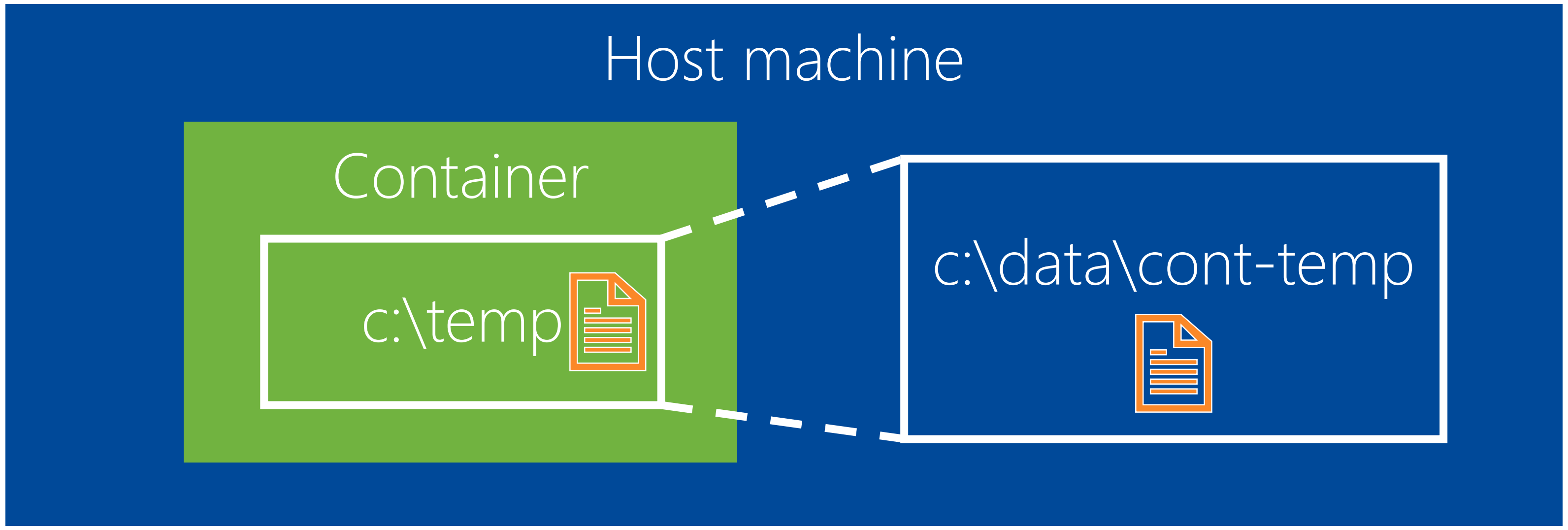
Standard filesystem setup: nothing configured. Use `docker cp` to copy files

LAB 3: FILE HANDLING AND BIND MOUNTS



Filesystem setup with a bind mount, e.g. `-v c:\data\cont-temp:c:\temp`

LAB 3: FILE HANDLING AND BIND MOUNTS



Filesystem setup with a bind mount, e.g. `-v c:\data\cont-temp:c:\temp`

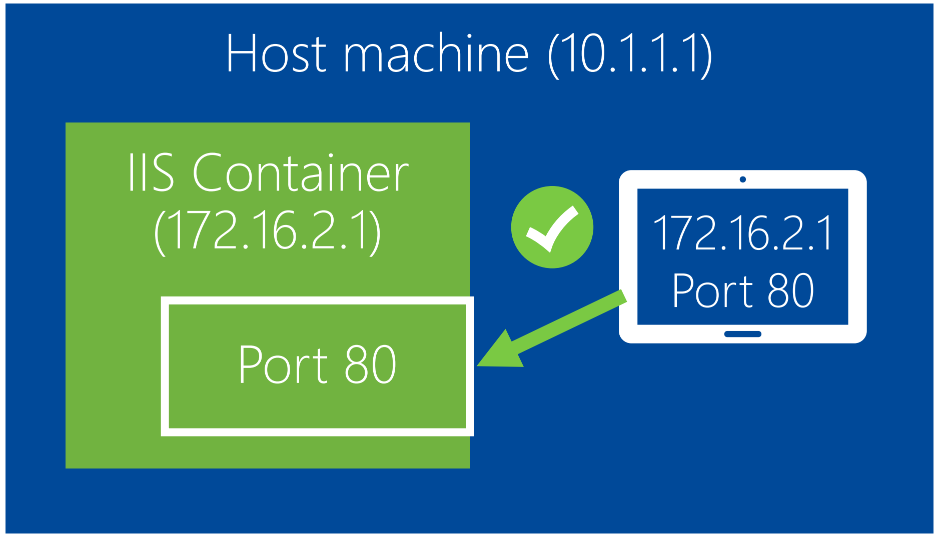
LAB 3: FILE HANDLING AND BIND MOUNTS

- 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
  - Use docker cp to copy files

  - Parameter -v for bind mounts allow sharing folders between host and container -> can only be set up on startup
  - Use a bind mount to share files between host and container

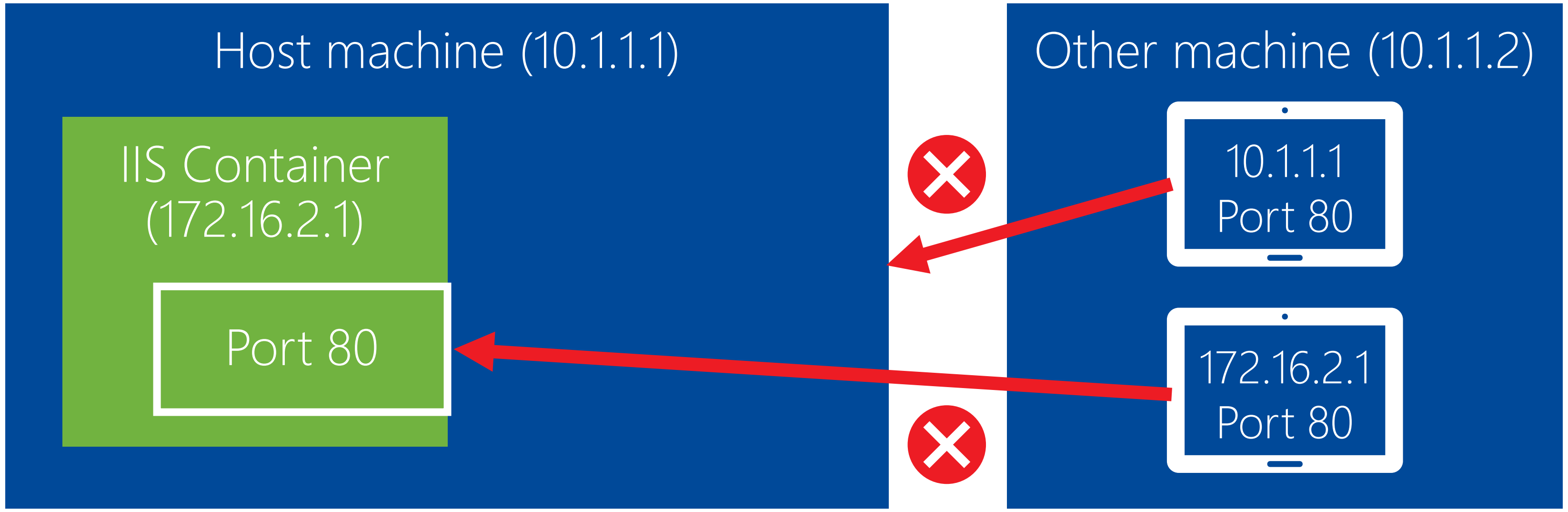
- See the bind mount in the inspect output
- Goal: See how the two different options for file sharing are used

LAB 4: NETWORKING



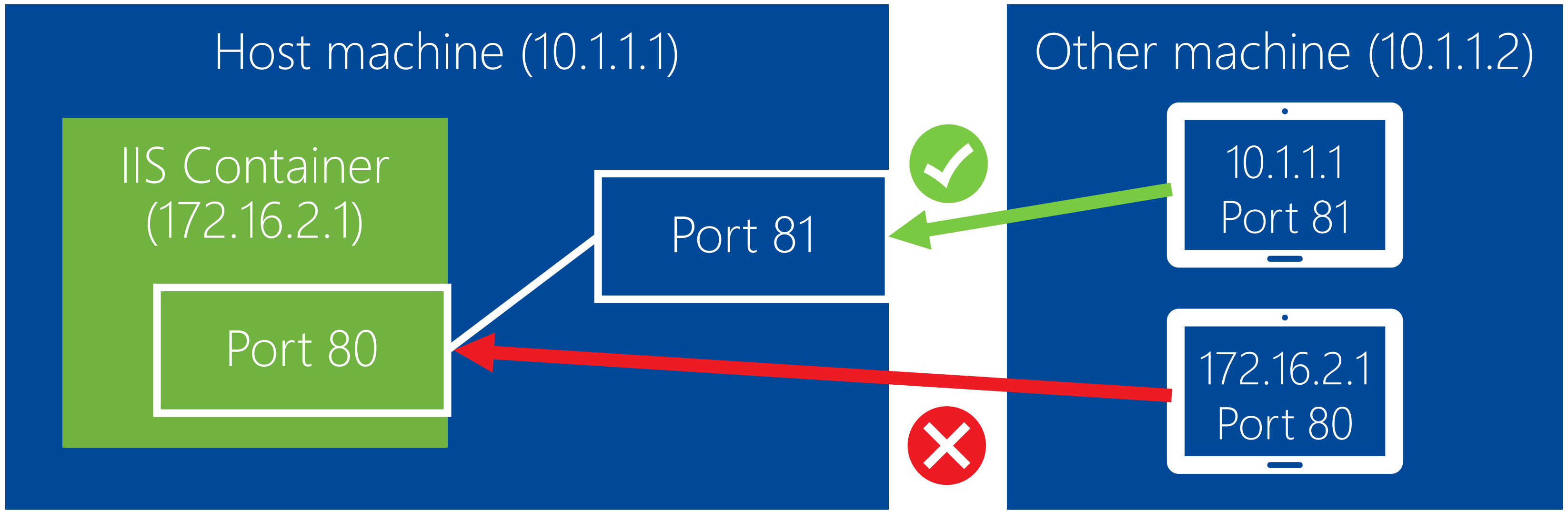
Standard networking setup: NAT

LAB 4: NETWORKING



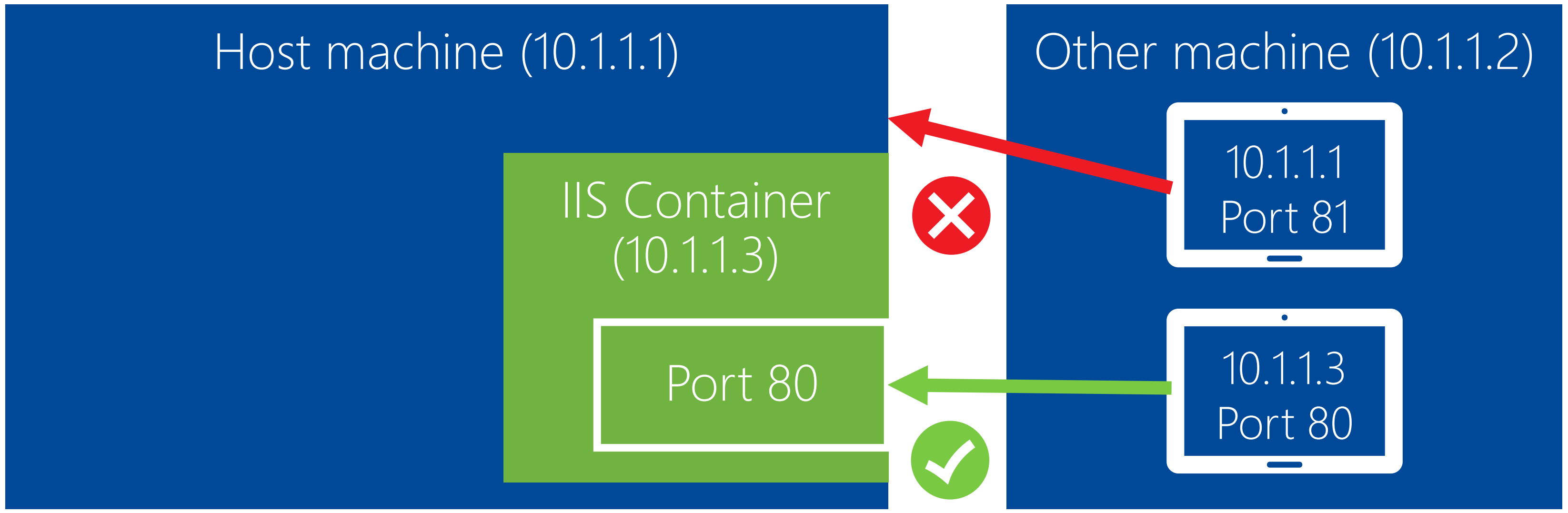
Standard networking setup: NAT

LAB 4: NETWORKING



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

LAB 4: NETWORKING



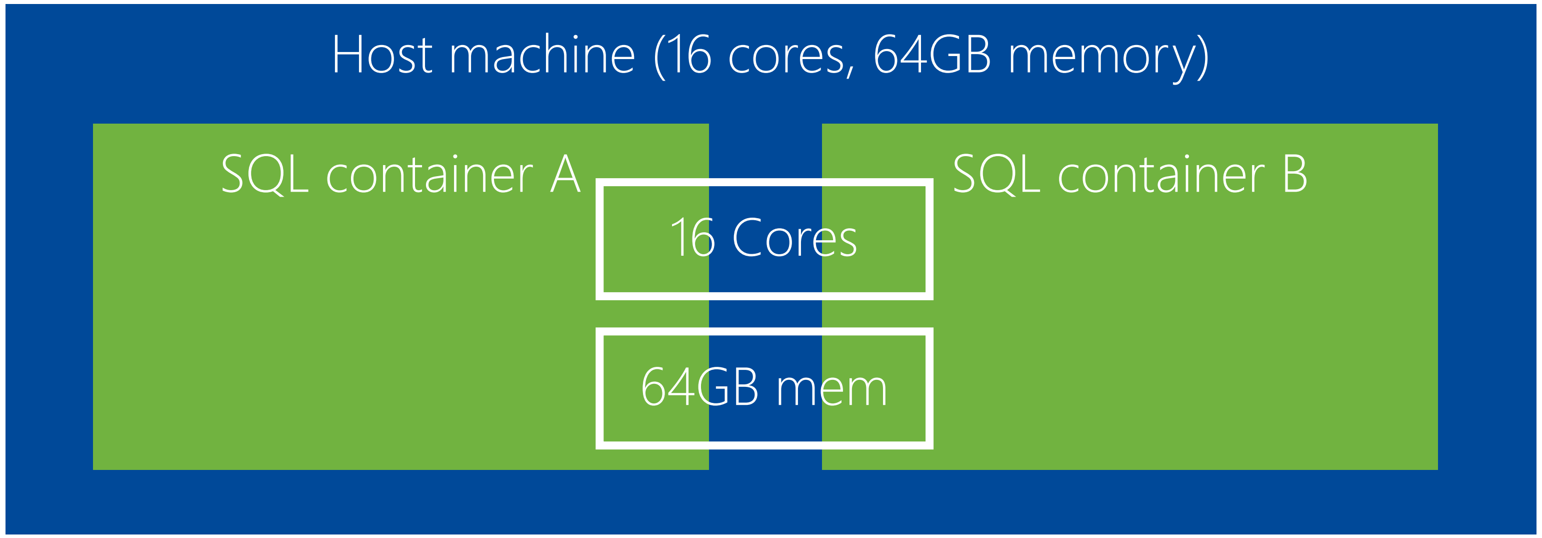
Transparent network setup: host and container “share” the network adapter

LAB 4: 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 evesy container and makes it reachable on that network

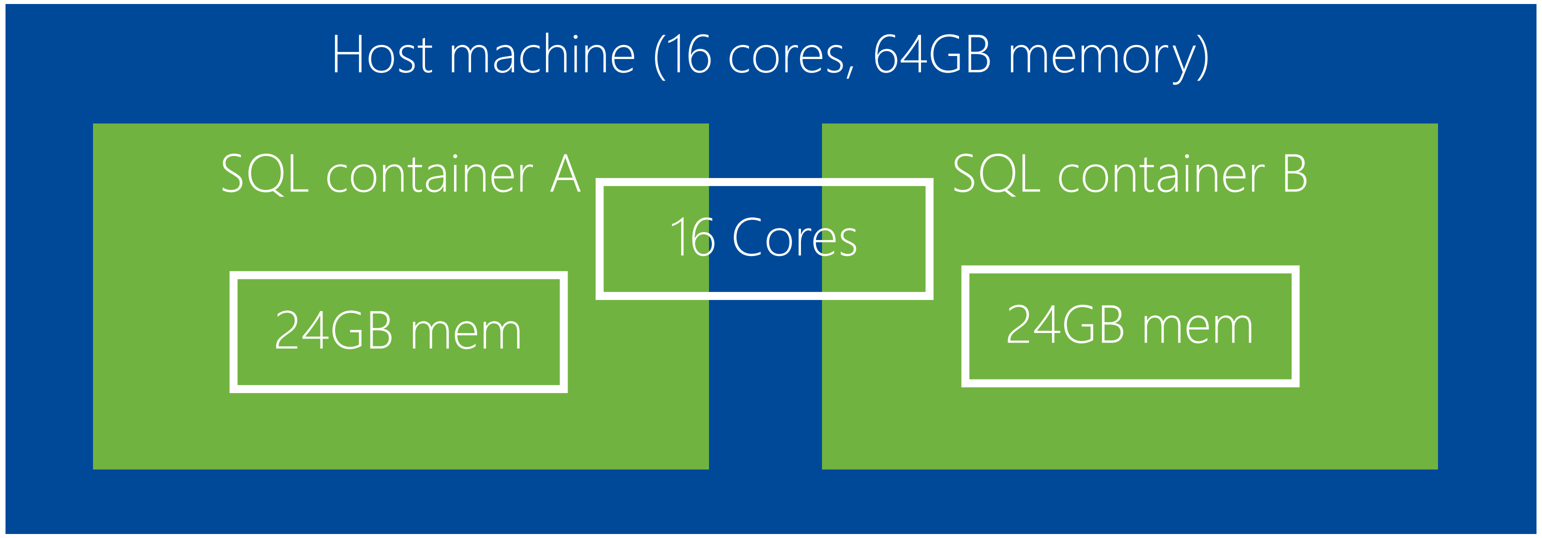
- Goal: Understand and test the different networking options

LAB 5: RESOURCE LIMITS



Standard resource setup: nothing configured

LAB 5: RESOURCE LIMITS



Specific resource setup: limits are configured, e.g. `-m 24g`

LAB 5: RESOURCE LIMITS

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

- Goal: Understand how to use those options and see how containers are behaving differently when those options are used

LAB 6: 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. mcr.microsoft.com/windows/servercore:1809
  - 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
- We use `docker build` to create an image, giving it a name with the `-t` param

LAB 6: DOCKERFILES

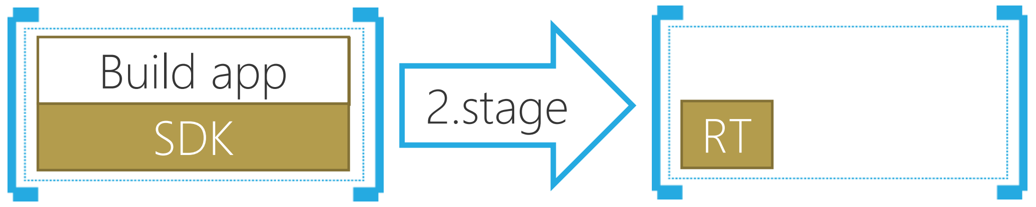
- Example 1: Create an Apache web server image by expanding the zip and install the .NET prereq using a silent installer
- Example 2: Create an image by building a custom solution where we have the sources
- Example 3: Use a multi-stage image to have an image as small as possible

- Goal: Understand how Dockerfiles work for different scenarios

LAB 6: 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

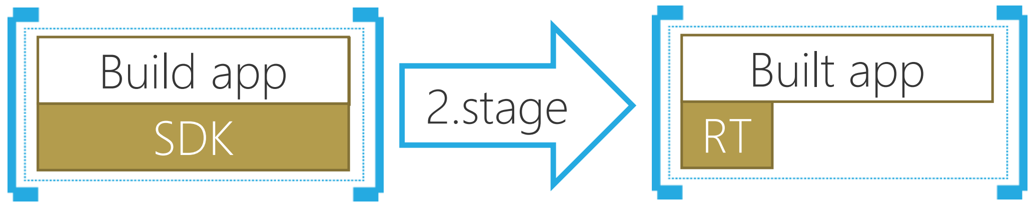
- Goal: Understand how Dockerfiles work for different scenarios



LAB 6: 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

- Goal: Understand how Dockerfiles work for different scenarios



LAB 7: 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: WordPress with a MySQL backend and admin tool

- Goal: See how you can easily define and manage multiple containers working together

ADDITIONAL NOTES AND BONUS LAB 8

- Working with the Docker API works remotely as well, but needs configuration of the Docker host (including firewall)
  - Easiest, but unsecure (make sure this is never exposed to the internet!) 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"

- Goal: Understand remote access and play with portainer

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 developer / Microsoft MVP)
- @SteveLasker (Microsoft PM working on container experiences)

LUNCH

PART 2 – BUSINESS CENTRAL ON DOCKER

BASIC STRUCTURE

- Public source repository https://github.com/microsoft/nav-docker
- On Premises images:
  - mcr.microsoft.com/dynamicsnav and mcr.microsoft.com/businesscentral/onprem
  - Tagged with:<ver>-<cu>-<country>-<winver>, e.g. 2017-cu22-de-ltsc2019
- SaaS images:
  - mcr.microsoft.com/businesscentral/sandbox -> current
  - bcinsider.azurecr.io/bcsandbox -> next minor
  - bcinsider.azurecr.io/bcsandbox-master -> next major
  - Tagged with:<build>-<country>-<winver>, e.g. 13.1.25940.26323-dk-ltsc2019

BASIC STRUCTURE

- Base of all: „generic“ image
  - FROM windowsservercore with .NET runtime 4.7.2 / 4.8
  - Install SQL Server and IIS dependencies
  - Copy files from Run folder into the image
  - Download Report Builder and some utils
- Same for all types (dynamicsnav, sandbox, onprem): „specific“ images W1 (called „base“ in bcsandbox) and local versions behave a bit different

BASIC STRUCTURE

- W1 / base built FROM generic:
  - Download NAV DVD and .vsix (AL extension for VS Code)
  - Move the right version specific files (folders 70 to 150) 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
- start.ps1 is called on docker run and calls all other scripts, depending on params and whether it is the first start of the container and whether the DNS name has changed

LAB 9: CONFIGURE THE NAV/BC CONTAINERS

- Custom NAV/BC settings and Web settings
- Use Windows authentication and enable ClickOnce
- Connect to an external SQL Server
  - As we are of course doing this in a container, we put it all in a compose file and extend it to add a test environment

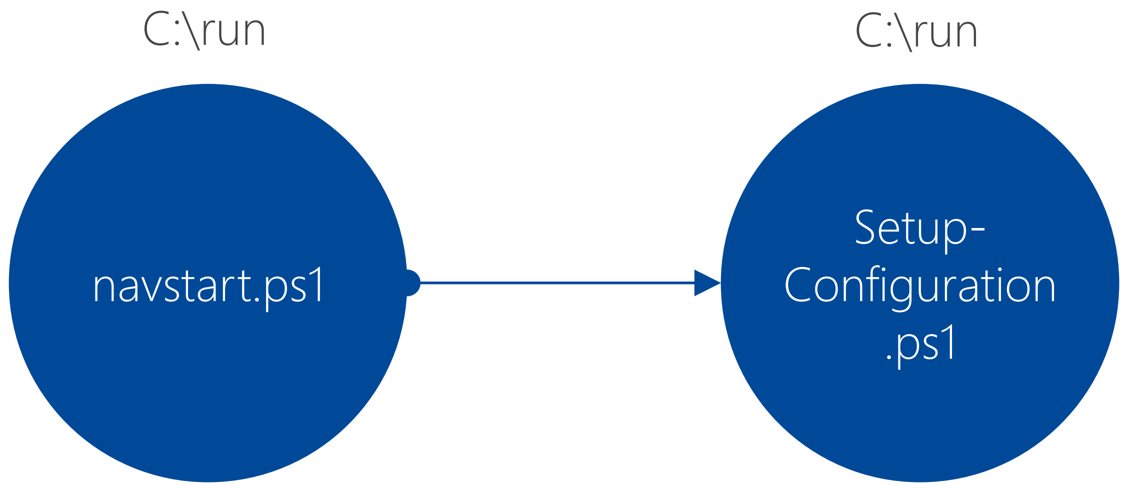
- Good way to find all possible parameters: Check SetupVariables.ps1

- Goal: See how you can easily configure the behavior of NAV/BC containers with simple start parameters

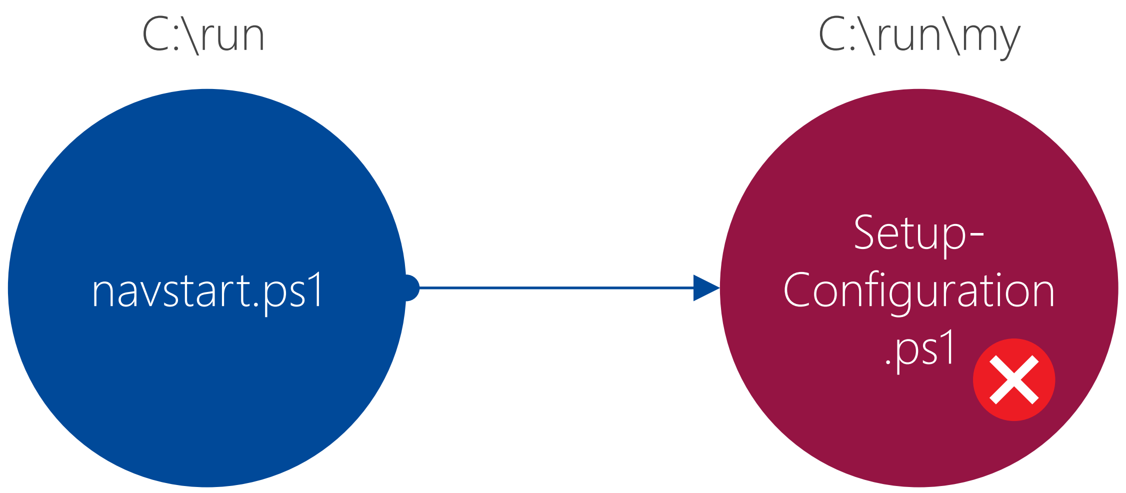
SCRIPT OVERWRITING

- If you really can't meet your requirements with env variables and configuring, you can also overwrite the existing scripts by using the following mechanism:
  - 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)
- Some scripts explicitly there just for overwriting like AdditionalOutput.ps or AdditionalSetup.ps1
- Startup works roughly like this: navstart.ps1 -> SetupVariables.ps1 -> SetupDatabase.ps1 -> SetupConfiguration.ps1 -> SetupWebClient.ps1 / SetupWebClientConfiguration.ps1 -> SetupFileShare.ps1 -> Setup*Users.ps1 -> AdditionalSetup.ps1 -> AdditionalOutput.ps1 -> MainLoop.ps1

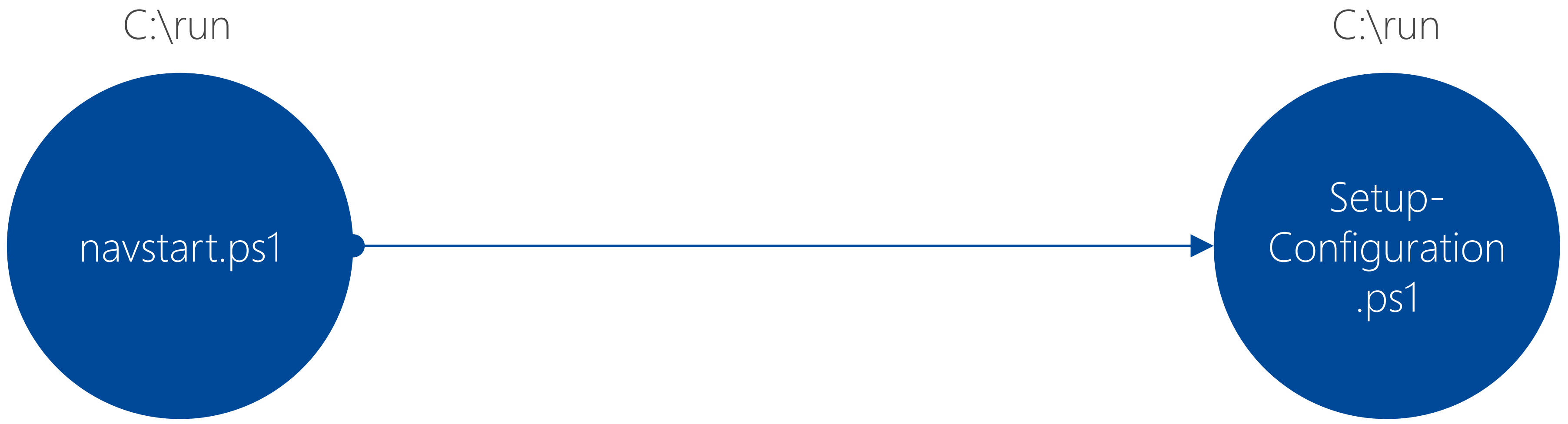
SCRIPT OVERWRITING



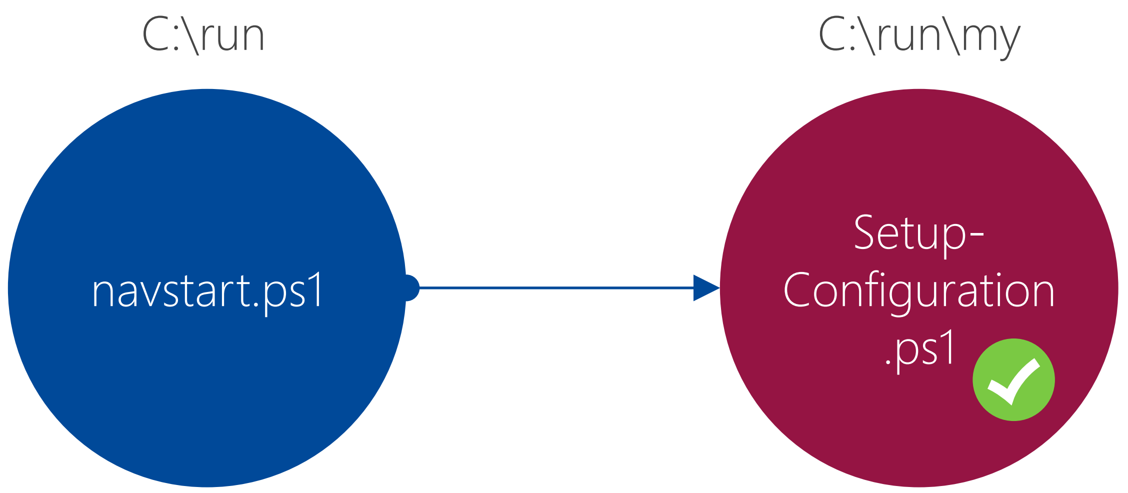
SCRIPT OVERWRITING



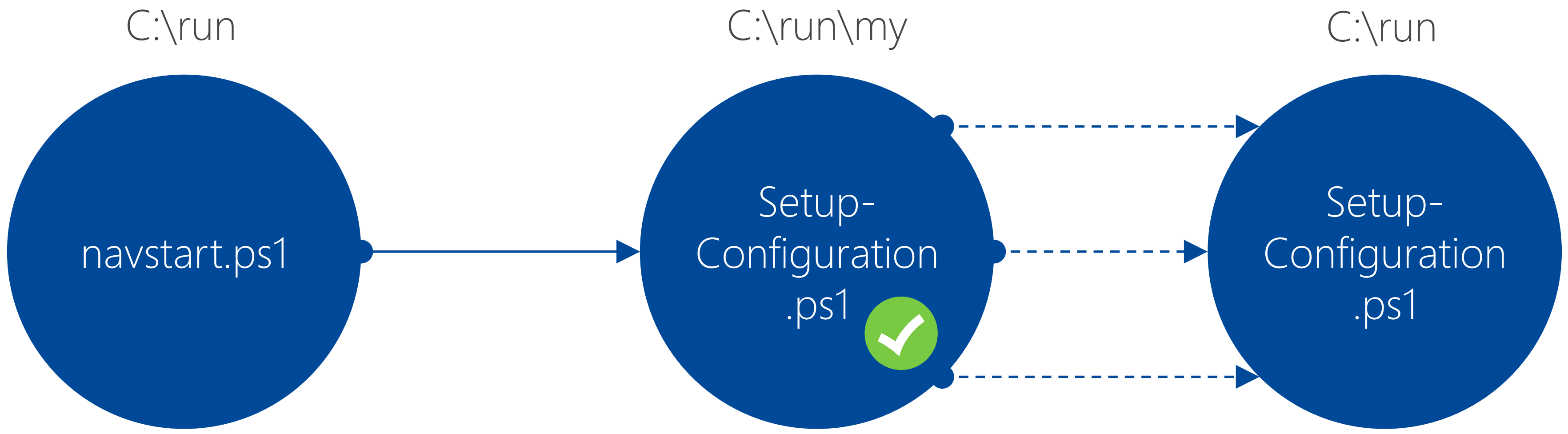
SCRIPT OVERWRITING



SCRIPT OVERWRITING



SCRIPT OVERWRITING



LAB 10: SCRIPT OVERWRITING

- Activating the API was a two-step process in BC 13: Enable the setting `ApiServicesEnabled` in the NST config and run Codeunit 5465. This can be achieved on startup using the already familiar `customNavSettings` env variable and then overwriting `AdditionalSetup.ps1` to run the Codeunit
- Validate this by calling the API
- Goal: See how the mechanism works

LAB 11: SPECIAL FILE HANDLING IN THE BC IMAGES

- Docker cp and bind mounts (including c:\run\my) work as expected
- Folder c:\run\Add-Ins is special: If you bind mount a folder with DLLs there, they automatically get distributed to the right places
- Script download using the folder parameter:
  - Environment param „folders“ triggers file download and puts them inside the container, e.g. -e folders="c:\temp=https://files.cosmoconsult.com/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 the same as before: activating the API endpoint by calling a codeunit

- Goal: See how you download files and put them in your container on startup

LAB 12: 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

- Goal: See how you can easily create an image on top of the standard images

LAB 13: 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, …

LAB 13: 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

- Goal: Get an introduction into the usage of navcontainerhelper

LINKS TO KNOW AND PEOPLE TO FOLLOW

- https://freddysblog.com/ and @freddydk
- https://github.com/Microsoft/nav-docker/
- https://github.com/Microsoft/navcontainerhelper

- "Using Docker and the ContainerHelper to convert your C/AL solution to an AL solution" by Freddy Kristiansen and Nicola Kukrika, Thursday 16-17:30h, room 5
- More advanced networking options and multi-container scenarios: "Make the most out of Business Central on Docker" by myself, Friday 11-12:30, room 9

COSMO CONSULT

Business-Software for People

THANK YOU

FOR TAKING PART

AND NOW IT'S UP TO YOU…