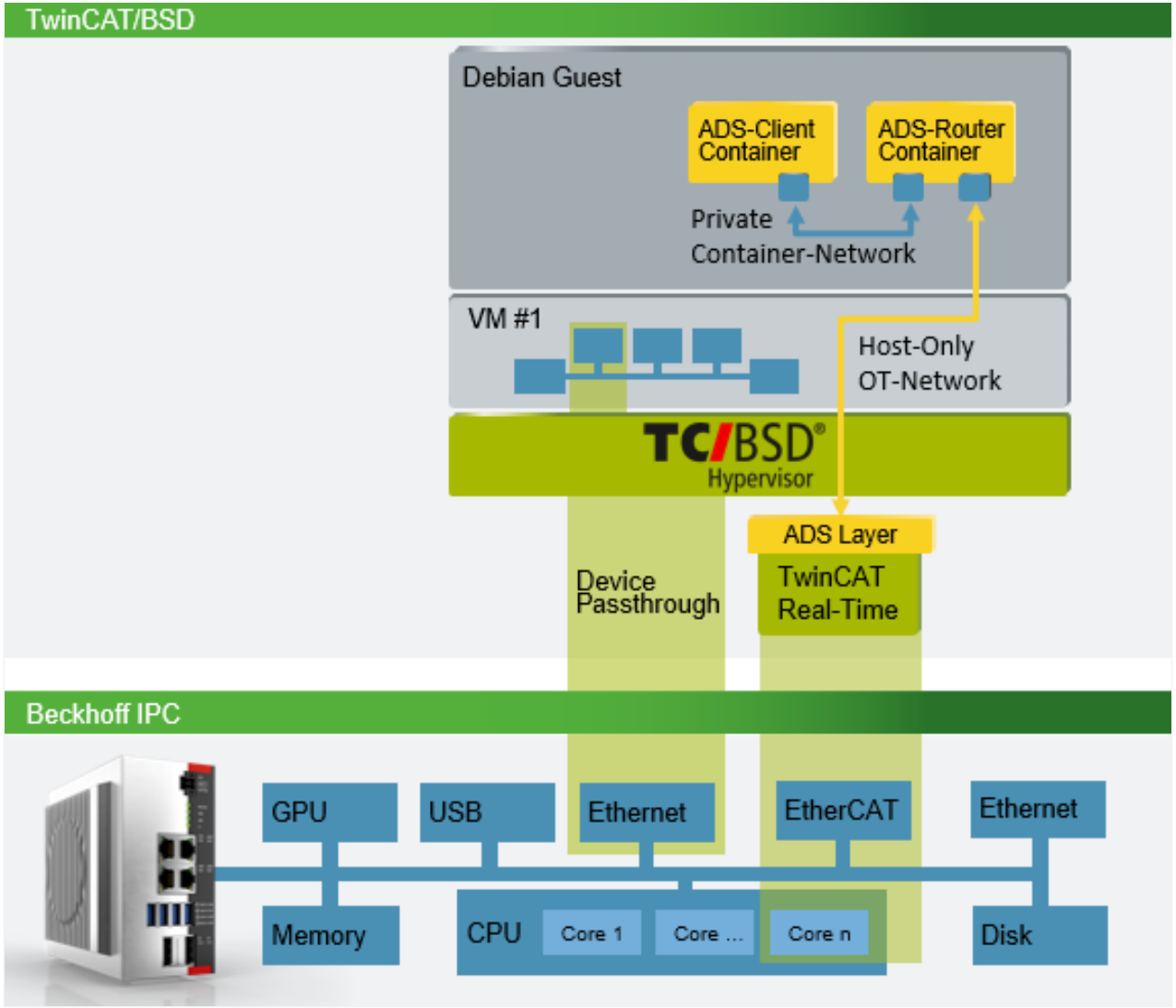
TC/BSD®

Hypervisor

Overview

BECKHOFF

- Create a Virtual Machine with 2 Docker containers for ADS Communication
- Tc/BSD hosts a Debian VM using Bhyve and runs a TwinCAT project.
- The Debian guest runs two Docker containers
  - A TcAdsRouter container to handle ADS comms between the other container and the TwinCAT run-time on the host
  - A Client container that sends requests to the TwinCAT run-time.



Steps

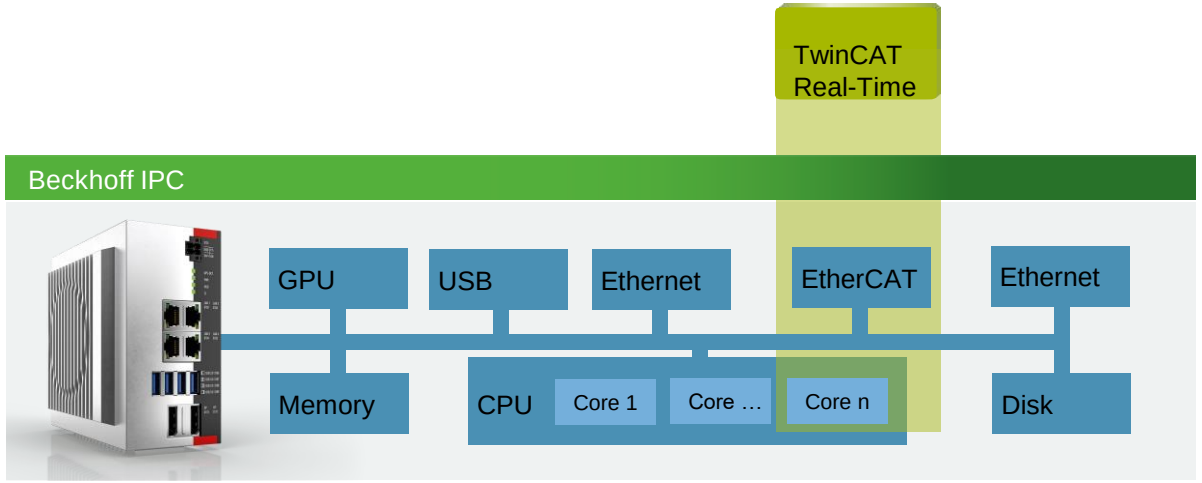
BECKHOFF

1. **Preparing Tc/BSD for the VM**
2. Installing the Linux guest
3. Installing Docker and other necessary programs
4. Building a Docker container image from a C# application
5. Final Set up and running the container.

Preparing the Tc/BSD Side

BECKHOFF

1. Activate the Sample TwinCAT project. Do NOT put TwinCAT in run mode.
2. Create virtual disk image for guest
3. Get installation media for the Linux distribution.
4. Explore the vmrun.sh script



Install Bhyve using the pkg service

BECKHOFF

- bhyve is not installed on Tc?BSD by default and can be downloaded from the Tc/BSD repository as part of a package.
  - doas pkg update
  - doas pkg install uefi-edk2-bhyve-bhf

Downloading the Debain Installation Disk

BECKHOFF

1. Create directory for vm files
   1. doas mkdir $home/vms
   2. doas mkdir $home/vms/debian
2. Download the iso file from Debian's network
   1. doas fetch -o "$HOME/vms/debian/debian-installer.iso" https://cdimage.debian.org/debian-cd/current/amd64/iso-cd/debian-11.3.0-amd64-netinst.iso

/usr/home/Administrator/

Name	Size	Changed	Rights	Owner
..		4/11/2022 11:26:09 ...	rwxr-xr-x	root
.ssh		3/29/2022 10:27:08 ...	rwxr-xr-x	Admi...
.cshrc	1 KB	3/29/2022 10:27:08 ...	rw-r--r--	Admi...
.default_warning	1 KB	3/29/2022 10:27:08 ...	rwxr--r--	Admi...
.login	1 KB	3/29/2022 10:27:08 ...	rw-r--r--	Admi...
.login_conf	1 KB	3/29/2022 10:27:08 ...	rw-r--r--	Admi...
.mail_aliases	1 KB	3/29/2022 10:27:08 ...	rw-------	Admi...
.mailrc	1 KB	3/29/2022 10:27:08 ...	rw-r--r--	Admi...
.profile	2 KB	3/29/2022 10:27:08 ...	rw-r--r--	Admi...
.shrc	1 KB	3/29/2022 10:27:08 ...	rw-r--r--	Admi...
amslog.cap	3 KB	4/11/2022 5:42:20 PM	rwx------	Admi...
debian-installer.iso	387,07...	3/26/2022 8:11:12 A...	rw-r--r--	Admi...
debian-vm.sh	2 KB	4/8/2022 1:31:38 PM	rw-r--r--	root
debian-vm-run.sh	1 KB	4/6/2022 2:51:53 PM	rw-r--r--	Admi...

3. Create the virtual disk image file
   1. doas zfs create -V 10G -o volmode=dev "zroot/debian-vm_disk0"

/dev/zvol/zroot/

Name	Size	Changed	Rights	Owner
..		4/8/2022 12:02:24 PM	r-xr-xr-x	root
debian-vm_disk0	0 KB	4/12/2022 3:22:32 PM	rw-r-----	root

Set up the Network Configuration

BECKHOFF

- We will need to create a bridge between the guest vm and the host computer to allow network communication to the guest.
- First, we need to know the name of the port we are using
  - Enter: ifconfig
  - Find the adapter name that has the “inet” field.
  - It should be em0/1 or igb0/1. Write it down or make a note.

```
em0: flags=8843<UP,BROADCAST,RUNNING,SIMPLEX,MULTICAST> metric 0 mtu 15
        options=810499<RXCSUM,VLAN_MTU,VLAN_HWTAGGING,VLAN_HWCSUM,LRO,V
        ether 00:01:05:13:fd:26
        inet6 fe80::201:5ff:fe13:fd26%em0 prefixlen 64 scopeid 0x1
        media: Ethernet autoselect
        status: no carrier
        nd6 options=23<PERFORMNUD,ACCEPT_RTADV,AUTO_LINKLOCAL>
em1: flags=8843<UP,BROADCAST,RUNNING,SIMPLEX,MULTICAST> metric 0 mtu 15
        options=810499<RXCSUM,VLAN_MTU,VLAN_HWTAGGING,VLAN_HWCSUM,LRO,V
        ether 00:01:05:13:fd:27
        inet6 fe80::201:5ff:fe13:fd27%em1 prefixlen 64 scopeid 0x2
        inet6 2601:441:4380:f970:201:5ff:fe13:fd27 prefixlen 64 autocon
        inet 192.168.0.15 netmask 0xffffff00 broadcast 192.168.0.255
        media: Ethernet autoselect (1000baseT <full-duplex>)
        status: active
        nd6 options=23<PERFORMNUD,ACCEPT_RTADV,AUTO_LINKLOCAL>
lo0: flags=8049<UP,LOOPBACK,RUNNING,MULTICAST> metric 0 mtu 16384
        options=680003<RXCSUM,TXCSUM,LINKSTATE,RXCSUM_IPV6,TXCSUM_IPV6>
        inet6 ::1 prefixlen 128
        inet6 fe80::1%lo0 prefixlen 64 scopeid 0x3
        inet 127.0.0.1 netmask 0xff000000
        groups: lo
        nd6 options=23<PERFORMNUD,ACCEPT_RTADV,AUTO_LINKLOCAL>
```

Set up the Network Configuration – Creating a Bridge Interface

BECKHOFF

To create the bridge and have it persist between power cycles of the system, we will add it to the rc.conf file.

- doas ee /etc/rc.conf
- Add the following lines to the file:
  - cloned_interfaces="bridge0"
- ifconfig_bridge0="inet <ip address> netmask 255.255.255.0"
- Press Esc then 'a' to exit, and 'a' again to save changes.

```
^[ (escape) menu  ^y search prompt  ^k delete line   ^p prev li   ^g prev page
^o ascii code     ^x search         ^l undelete line ^n next li   ^v next page
^u end of file    ^a begin of line  ^w delete word   ^b back 1 char
^t top of text    ^e end of line    ^r restore word  ^f forward 1 char
^c command        ^d delete char    ^j undelete char ^z next word
=====line 23 col 20 lines from top 23 ==========================================
zfs_enable="YES"
dumpdev="AUTO"
CXID_enable="YES"
MDPService_enable="YES"
ntpd_enable="YES"
pf_enable="YES"
sshd_enable="YES"
TcSystemService_enable="YES"
runonce_enable="YES"
sendmail_enable="NO"
sendmail_submit_enable="NO"
sendmail_outbound_enable="NO"
sendmail_msp_queue_enable="NO"
background_dhclient="YES"
defaultroute_delay="0"
ifconfig_DEFAULT="DHCP inet6 accept_rtadv"
cloned_interfaces="bridge0"
ifoncig_bridge0="inet 192.168.3.1 netmask 255.255.255.0"
rtsold_enable="YES"
allscreens_kbdflags="-b quiet.off"
syslogd_flags="-ss"
devfs_system_ruleset="allow_usb_mount"
hostname="CX-13FD26"
```

Set up the Network Configuration - Firewall

BECKHOFF

- Set up firewall rules for the VM and ADS communication
  - /etc/pf.conf is the default configuration file for the BSD firewall.
  - We have another file in /etc/pf.conf.d/ named bhf for Beckhoff specific rules

Set up the Network Configuration – Firewall Cont.

BECKHOFF

- To open the configuration file use the command

  doas ee /etc/pf.conf.d/bhf

- We can add the following lines to that file to allow Communication to the VM
  - pass in quick on bridge0
  - pass in quick proto tcp to port 5900
- Enable unencrypted ADS as well
  - pass in quick proto tcp to port 48898
- Exit and save
- To enable the rules use the command

  doas pfctl –f /etc/pf.conf.d/bhf

```
^c command        ^d delete char    ^j undelete char ^z next word
=====line 16 col 0 lines from top 16 ==================================
# WARNING this file might be automatically overwritten by system upgrades.
# Add your changes to /etc/pf.conf directly or even better and an anchor
# and your custom config to /etc/pf.conf.d/, just like we did for bhf.

# allow IPC Diagnostics
pass in quick proto tcp to port https

# allow ADS secure
pass in quick proto tcp to port 8016
pass in quick proto udp to port 48899
pass in quick proto tcp tp port 48898

# allow communication for Bhyve VM
pass in quick on bridge0
pass in quick proto tcp to port 5900

# allow dynamic configuration for legacy ADS
anchor ads

# allow dynamic configuration for bhyve
anchor "bhyve/*"

# allow dynamic configuration for TwinCAT ADS Monitor - AMS Logger
anchor tcamslog

# allow dynamic configuration for TF2000-HMI-Server
anchor tchmisrv

# allow dynamic configuration for OPC-UA server
anchor tcopcuaserver
```

Exploring the debian-vm.sh Script

BECKHOFF

For customers to have the option of creating their own custom guest OSs, they are going to need some scripting knowledge to get it running at startup. Especially if they want to try to create a Windows guest.

- For starting the VM we can let the script debian-vm.sh handle the rest.
- The script is meant to specifically launch a bhyve instance for a Debian VM.

Exploring the script - variables

BECKHOFF

- The variables here are mostly paths to the files and directories we created.

- vm_name is the given to the image file.
- Iso_cd0 is the location of the previously downloaded installer cd.
- vnc_port will be the port we connect to using UltraVNC to interface with the Debian guest

```
vm_name="debian-vm"
vm_dir="/usr/home/Administrator/vms/debian"
iso_cd0="${vm_dir}/debian-installer.iso"

vm_bridge0="bridge0"
vnc_port="5900"
nic_port="em1"
```

Exploring the Script – The Main Instructions Section

BECKHOFF

- After creating the variables, this is the next part to execute.
- Its purpose is first to make sure the script is run with Admin rights, and to direct the flow to the correct command.
- When calling the script, the arguments tell it what task to perform.
  - Config runs the rest of the configuration steps.
  - Run will call Bhyve to create the VM
  - Destroy deletes the instance and reverses the steps performed in Config.

```
#Main script instructions
if [ $(id -u) -ne 0 ]; then
    err "${0##*/} must be run as root!"
fi


install_os="false"

for _arg in $@; do
    case ${_arg} in
        --install)
            install_os="true"
            break
        ;;
    esac
done

if [ $# -lt 1 ]; then
    err "No Commands Provided"
fi

case $1 in
    config)
        vm::config
        ;;
    run)
        vm::init
        vm::run
        ;;
    destroy)
        vm::destroy
        vm::cleanup
        ;;
esac
```

Exploring the Script - Configuration

BECKHOFF

```
vm::config(){

    if [ ! -f "${efi_vars}" ]; then
            cp /usr/local/share/uefi-firmware/BHYVE_BHF_UEFI_VARS.fd "${vm_dir}/EFI_VARS.fd"
    fi

    _tap0="$(ifconfig tap create)"
    sysctl net.link.tap.up_on_open=1 #enable the tap interface
    sysctl net.link.bridge.pfil_member=0 #enable communication through the firewall to members of the bridge interface

    # bridge device is created via cloned interfaces in rc.conf
    #this line issues the command to add the tap0 and em1 interfaces as members of bridge0.
    #This will allow communication on the em1 interface to be sent to the tap0 interface where the vm will be able to receive it
    ifconfig bridge0 addm tap0 addm ${nic_port}


}
```

Exploring the Script – Starting Bhyve

BECKHOFF

```
vm::run(){

    while true; do

        if [ "${install_os}" == "true" ]; then #if the caller has indeicat
                _installerdisk="-s 10:0,ahci-cd,${iso_cd0}"
            else
                _installerdisk=""
            fi

        bhyve -c 1 -m 1G \
        -A -H \
        -l bootrom,/usr/local/share/uefi-firmware/BHYVE_BHF_UEFI.fd \
        -s 0:0,hostbridge \
        -s 1:0,virtio-blk,/dev/zvol/zroot/debian-vm_disk0 \
        -s 2:0,virtio-net,tap0 \
        ${ installerdisk} \
        -s 29:0,fbuf,tcp=0.0.0.0:"${vnc_port}",w=1024,h=768,wait \
        -s 30:0,xhci,tablet \
        -s 31:0,lpc \
        debian-vm
```

- Bhyve is the command. We're are passing 1 CPU core and 1GB of Memory.
- -A tells Bhyve to create an ACPI table, and –H to yield the CPU thread when a halt instruction is detected.
- -l is passing the boot firmware.
- 0:0 is the hostbridge.
- 1:0 is the virtual disk created earlier.
- 2:0 is the tap interface created when the config command was called. Any network traffic to the VM will go through it.
- _installerdisk is passed as a cd device if installing, otherwise it is blank.
- 29:0 is the framebuffer it is passing the graphical information to the VNC port and waiting for a connection to start booting.
- 30:0 will enable mouse input through the VNC
- 31:0 addresses the LPC ISA bridge
- debian-vm will be the name of the VM instance.

Steps

BECKHOFF

1. Setting up Tc/BSD for the VM
2. **Installing the Linux guest**
3. Installing Docker and other necessary programs
4. Building a Docker container image from a C# application
5. Final Set up and running the container

Launch the VM

BECKHOFF

- debian-vm.sh will handle the rest of the configuration. Use the command

```
doas sh debian-vm.sh config
```

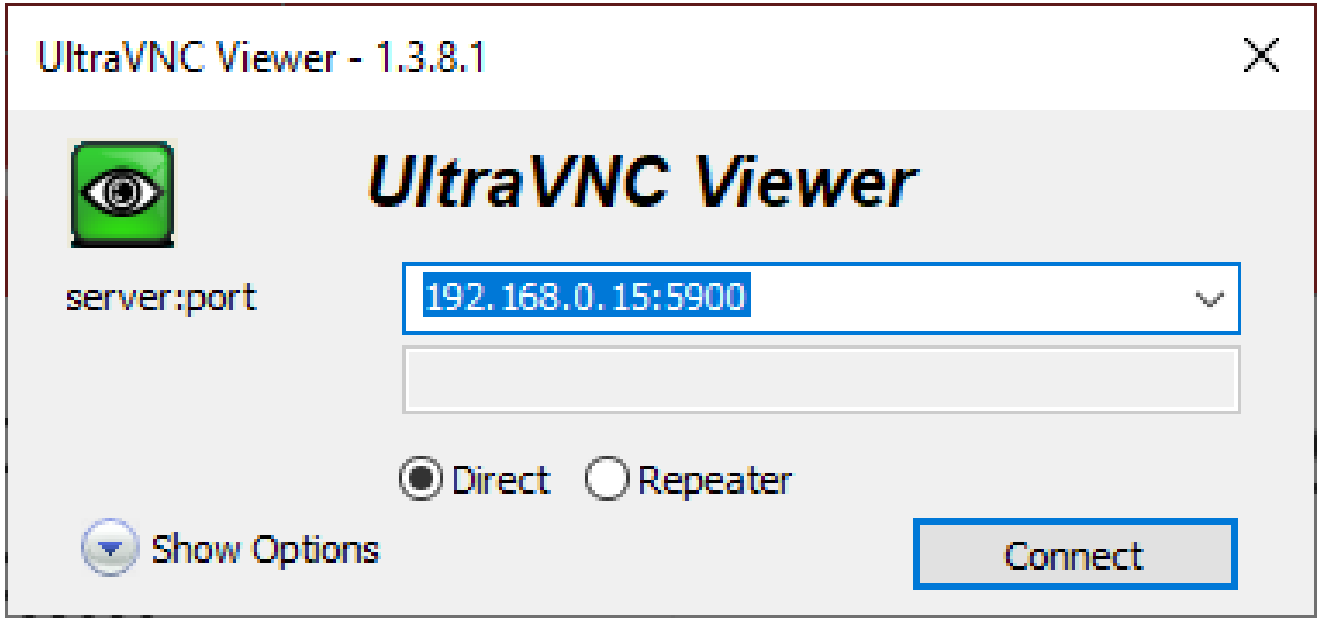
- In order to install the vm we will call that script the --install argument.
- To run the script and have the VM boot to the Debian Installer, enter the command:

```
doas sh debian-vm.sh run –-install
```

Connect to the VM using a VNC viewer

BECKHOFF

1. Open Ultra VNC Viewer on your laptop
2. Connect to <ipaddress>:5900



Follow Debian Installer

BECKHOFF

- Once the installation process is finished, call the script again without the –install argument:

```
doas sh debian-vm.sh run
```

- Then Connect via Ultra VNC again.

Steps

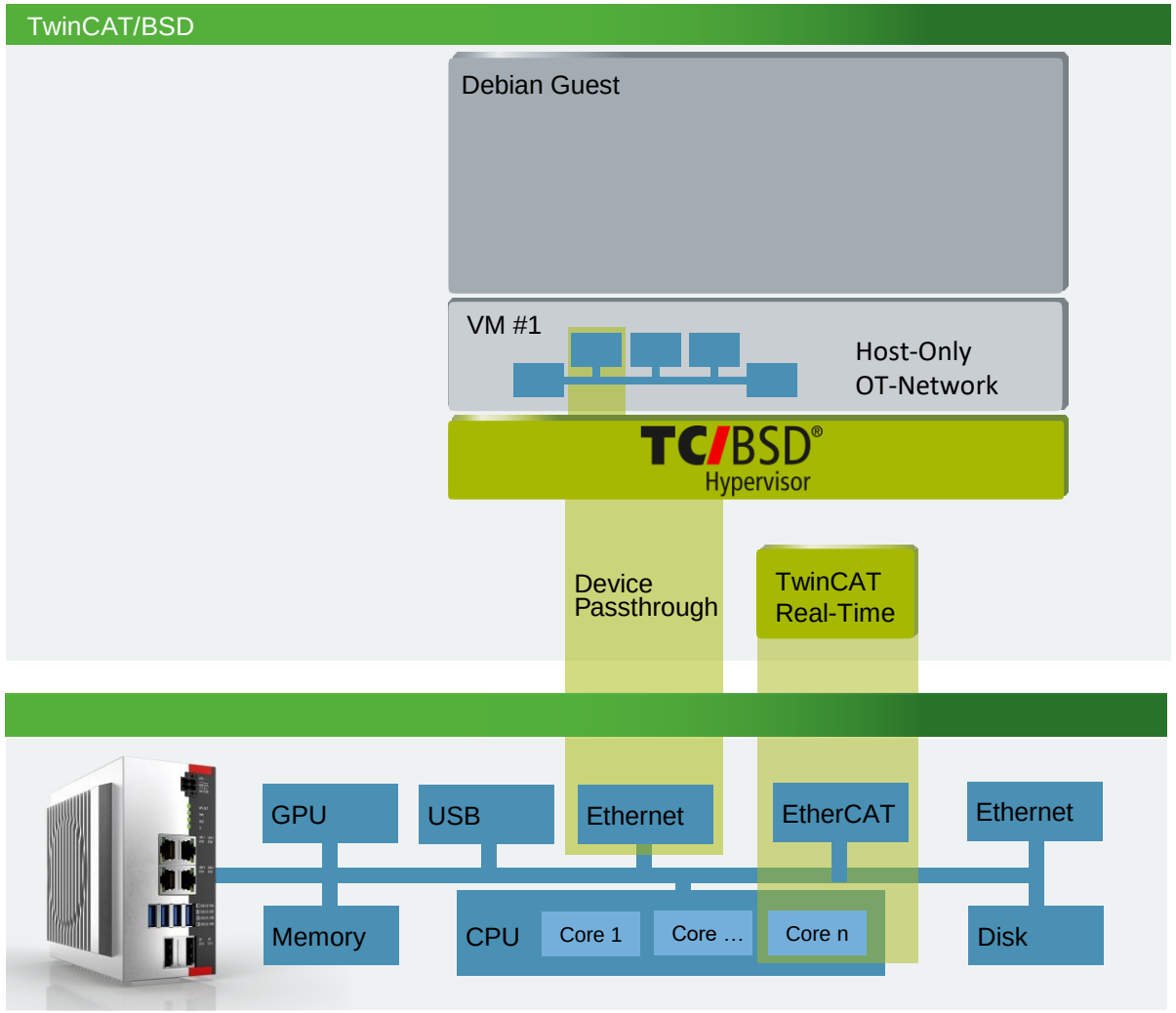
BECKHOFF

1. Setting up Tc/BSD for the VM
2. Installing the Linux guest
3. **Installing Docker and other necessary programs**
4. Building a Docker container image from a C# application
5. Final Set up and running the container

Preparing the Linux Guest

BECKHOFF

1. Connect to the VM using PuTTY
2. Install dotnet 6.0 SDK
3. Install packages required to install and run Docker.
4. Install Docker



A Note About Working in Debian

BECKHOFF

For those who are used to working in Tc/BSD but not Debian:

- sudo = doas
- apt = pkg
- ifconfig is deprecated and not installed by default. Use the command “ip”
  - `ip addr` will perform the same function as ifconfig
- Easy Editor is not installed by default, instead we will be using nano, which is a bit more robust without vims opaque UE.

Install important packages

BECKHOFF

- Get Microsoft signing key and repository for dotnet:
  - cd $Home
  - sudo wget https://packages.microsoft.com/config/debian/11/packages-microsoft-prod.deb -O packages-microsoft-prod.deb
  - sudo dpkg -i packages-microsoft-prod.deb
  - sudo rm packages-microsoft-prod.deb
  - sudo apt update

Install Important Packages

BECKHOFF

1. Install packages needed to install and use docker
    1. sudo apt install -y apt-transport-https ca-certificates curl gnupg2 software-properties-common dotnet-sdk-6.0 unzip
2. Get the Docker repository
    1. sudo curl -fsSL https://download.docker.com/linux/debian/gpg | sudo apt-key add –
    2. sudo add-apt-repository "deb [arch=amd64] https://download.docker.com/linux/debian $(lsb_release -cs) stable"
3. Install Docker
    1. sudo apt update
    2. sudo apt-cache policy docker-ce
    3. sudo apt install -y docker-ce

Steps

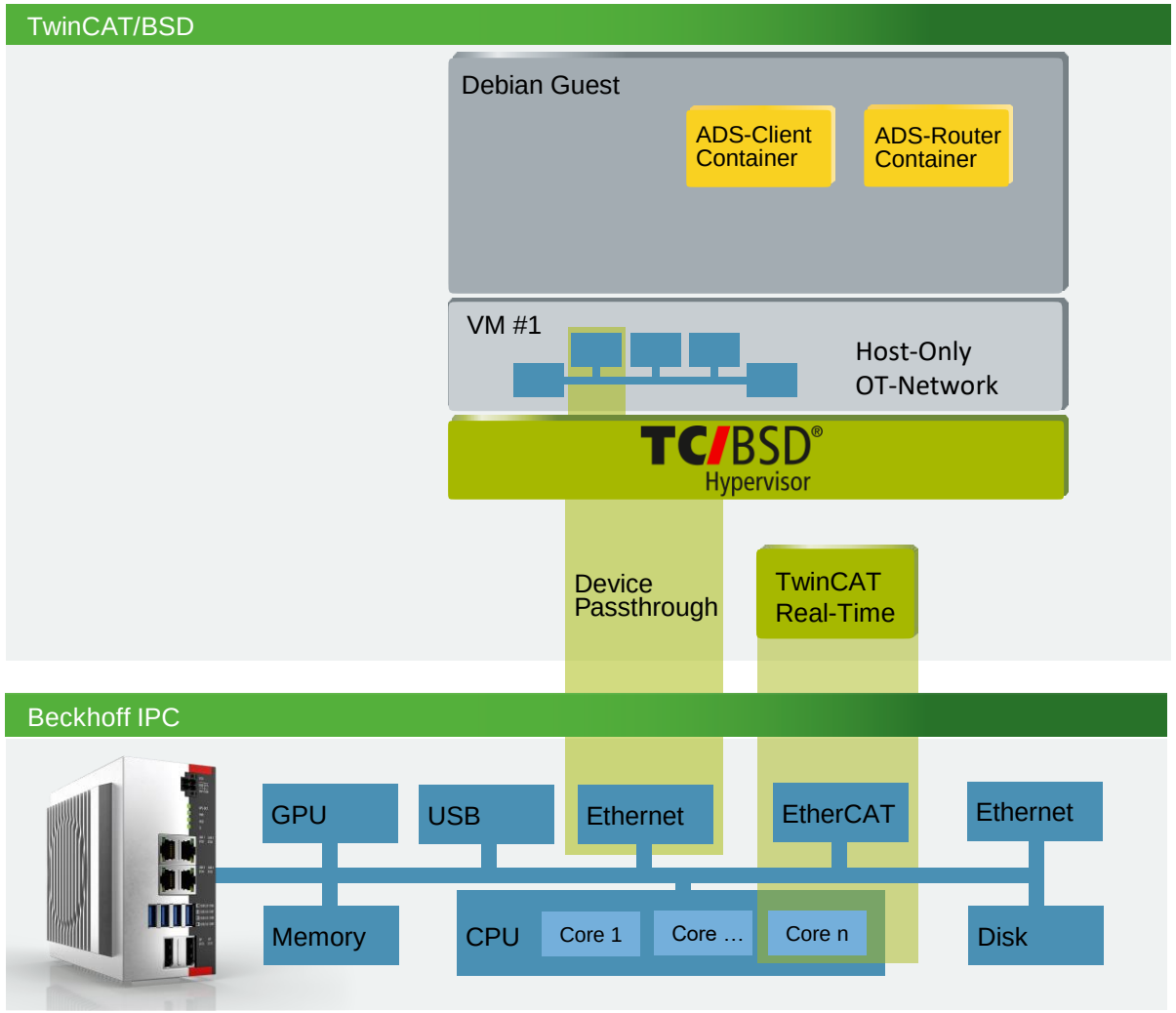
BECKHOFF

1. Setting up Tc/BSD for the VM
2. Installing the Linux guest
3. Installing Docker and other necessary programs
4. **Building a Docker container image from a C# application**
5. Final Set up and running the container

Creating the Docker Containers

BECKHOFF

1. Download sample project from the Beckhoff GitHub
2. Edit the settings files for the containers
3. Build the images



Dowloading the Application for the Container

BECKHOFF

1. Download GitHub samples from the Beckhoff GitHub
    1. sudo wget https://github.com/Beckhoff/TF6000_ADS_DOTNET_V5_Samples/archive/refs/heads/main.zip
    2. sudo unzip main.zip
    3. sudo rm main.zip

Building the AdsRouterConsole Container

BECKHOFF

- Build container for the AdsRouterConsole
  - cd $HOME/TF6000_ADS_DOTNET_V5_Samples-main/Sources/RouterSamples/AdsRouterConsoleApp/
  - sudo docker build -t ads-router-console --target=final --file Dockerfile .
    - The period at the end is meant to be there, do not forget it.

```
#See https://aka.ms/containerfastmode to understand how Visual Stu

FROM mcr.microsoft.com/dotnet/runtime:5.0 AS base
WORKDIR /app

FROM mcr.microsoft.com/dotnet/sdk:5.0 AS build
WORKDIR /src
COPY src/*.csproj .
RUN dotnet restore
COPY src/. .
RUN dotnet build -c Release -o /app/build --framework net5.0

FROM build AS publish
RUN dotnet publish -c Release -o /app/publish --framework net5.0

FROM base AS final
WORKDIR /app
COPY --from=publish /app/publish .

ENTRYPOINT ["dotnet", "AdsRouterConsoleApp.dll"]
```

Building the AdsClientConsole Container

BECKHOFF

- Build the container for the AdsClientConsole
  - cd $HOME/TF6000_ADS_DOTNET_V5_Samples-main/Sources/ClientSamples/AdsCli
  - sudo docker build -t ads-cli-client:latest .
    - The period at the end is important, do not forget it.
- Double Check that our images were built
  - sudo docker images
    - Should show 1 entry for ads-router-console, 1 for client-console

```
#See https://aka.ms/containerfastmode to understand ho

FROM mcr.microsoft.com/dotnet/runtime:6.0 AS base
WORKDIR /app

FROM mcr.microsoft.com/dotnet/sdk:6.0 AS build
WORKDIR /src
COPY src/*.csproj .
RUN dotnet restore
COPY src/. .
RUN dotnet build -c Release -o /app/build

FROM build AS publish
RUN dotnet publish -c Release -o /app/publish

FROM base AS final
WORKDIR /app

ENV AmsConfiguration:LoopbackAddress=127.0.0.1 \
    AmsConfiguration:LoopbackPort=48898

COPY --from=publish /app/publish .

ENTRYPOINT ["dotnet", "AdsCli.dll"]
```

Steps

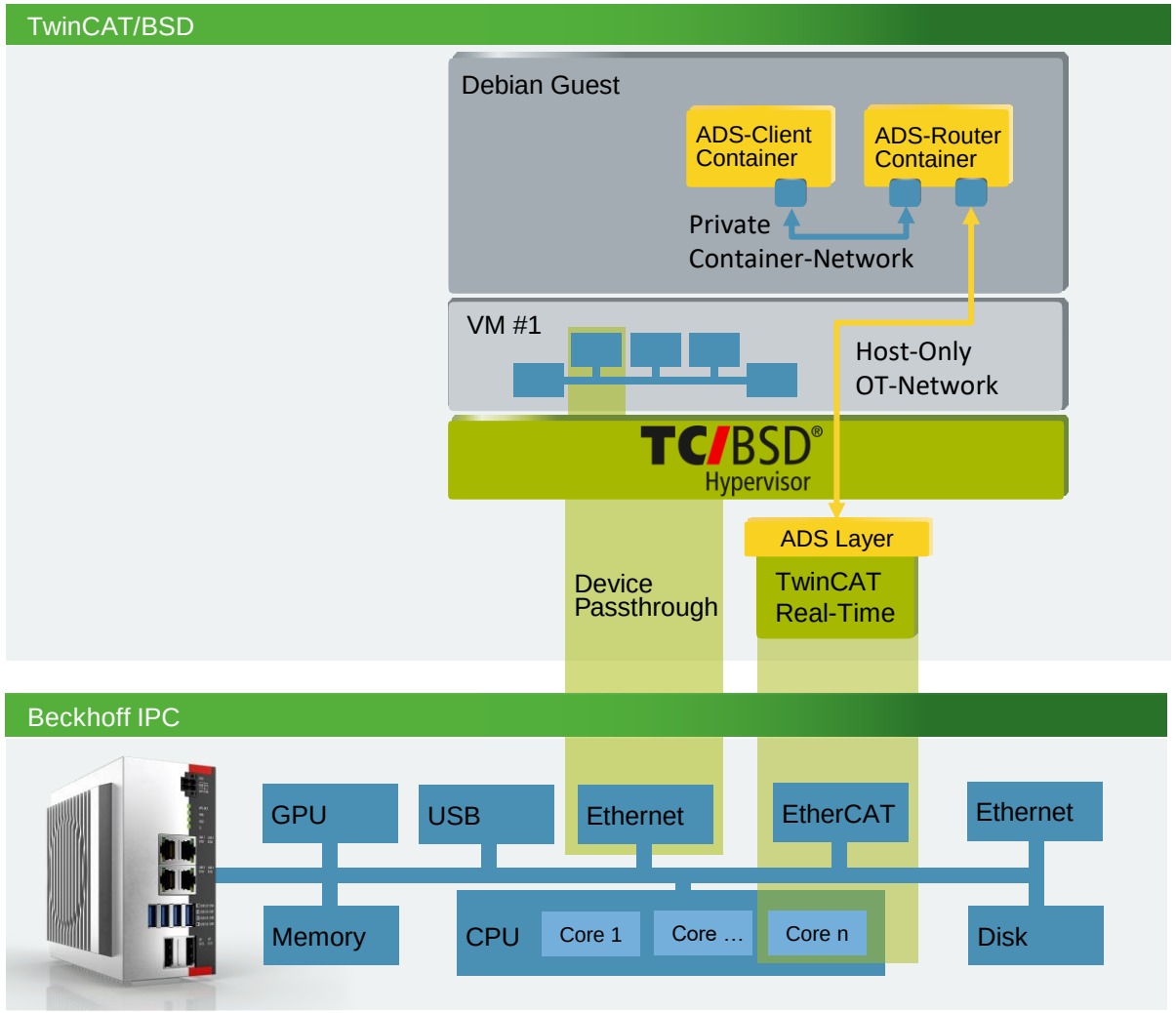
BECKHOFF

1. Setting up Tc/BSD for the VM
2. Installing the Linux guest
3. Installing Docker and other necessary programs
4. Building a Docker container image from a C# application
5. **Final Set up and running the container**

Final Steps

BECKHOFF

1. Edit routes for the AdsRouterConsole container
2. Add the AdsRouterContainer to the routing table of the host's TwinCAT router.
3. Run the containers.



Creating the Router Table for the AdsRouterConsole Container

BECKHOFF

1. Return to AdsRouterConsoleApp directory

```
cd $HOME/TF6000_ADS_DOTNET_V5_Samples-main/Sources/RouterSamples/AdsRouterConsoleApp/
```

2. Copy router table sample from the “src” directory to the current directory.

```
sudo cp ./src/settings-bridged-network.env ./settings.env
```

3. Open the file with a text editor

```
sudo nano settings.env
```

Cont.

BECKHOFF

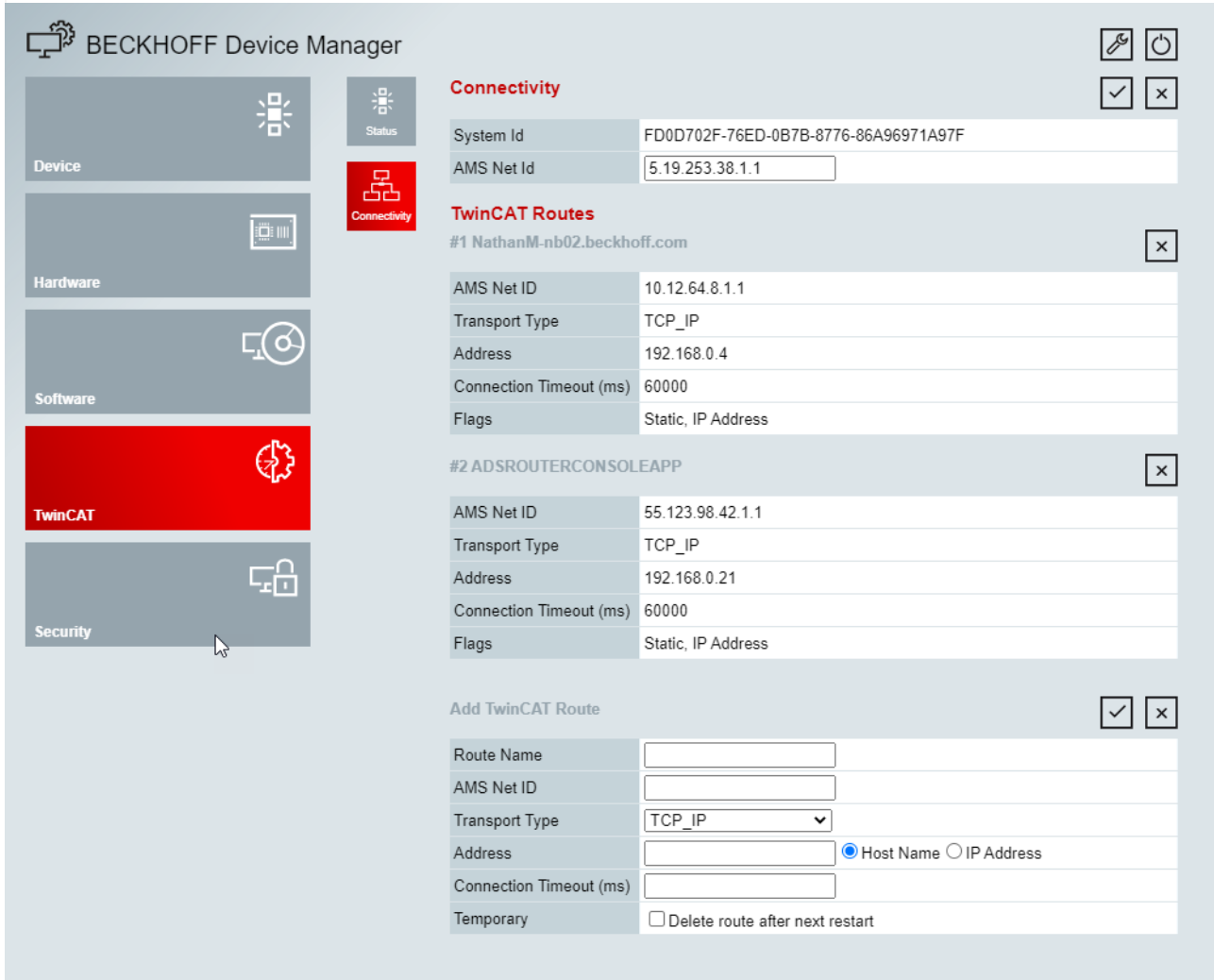
bhyve

```
  GNU nano 5.4                                    settings-host-network.env
# Alternative settings via Environment Variables
# using config.AddEnvironmentVariables("ENV_") in ConfigureAppConfiguration

# Basic ADS-Router config which has to match StaticRoutes on TwinCAT-Host systems
ENV_AmsRouter__Name=AdsRouterConsoleApp
ENV_AmsRouter__NetId=55.123.98.42.1.1

ENV_AmsRouter__LoopbackIP=172.17.0.2
ENV_AmsRouter__LoopbackPort=48900

ENV_AmsRouter__LoopbackExternalSubnet=172.17.0.0/16

# Indexed List of remote connections to TwinCAT Hosts
# First TwinCAT-Host
ENV_AmsRouter__RemoteConnections__0__Name=TwinCAT-Host
ENV_AmsRouter__RemoteConnections__0__Address=192.168.0.15
ENV_AmsRouter__RemoteConnections__0__NetId=5.19.253.38.1.1

# Verbose log output
ENV_Logging__LogLevel__Default=Debug
```

Adding the AdsRouterConsole Container to the Host Routing Table

BECKHOFF

1. Get the guest's IP address
   1. ip addr
2. On your ENG PC, connect to the Host's Web Server: https://<ipaddress>
3. Go to the Device Manager then Navigate to the Network page under TwinCAT
4. Enter the routing information of the container and save.



Run the Containers

BECKHOFF

1. Make sure TwinCAT is in run mode on the host and the PLC program is started.
   1. Use `doas TcSysExe.exe --run` to restart in run mode
2. Start the AdsRouterConsole container
   1. `sudo docker run -it --rm --name adsrouter --env-file=settings.env --ip 172.17.0.2 -p 48898:48900 ads-router-console`

```
      External Port:     bf02
info: TwinCAT.Ads.AdsRouterService.RouterService[0]
      Loopback IP:       172.17.0.2
info: TwinCAT.Ads.AdsRouterService.RouterService[0]
      Loopback Port:     bf04
info: TwinCAT.Ads.AdsRouterService.RouterService[0]
      LoopbackExternals: 172.17.0.0/16
info: TwinCAT.Ads.AdsRouterService.RouterService[0]
info: TwinCAT.Ads.AdsRouterService.RouterService[0]
      Configured routes:
info: TwinCAT.Ads.AdsRouterService.RouterService[0]
      ==================
info: TwinCAT.Ads.AdsRouterService.RouterService[0]
        TwinCAT-Host, 5.19.253.38.1.1, 192.168.0.15
info: TwinCAT.Ads.AdsRouterService.RouterService[0]
info: TwinCAT.Ads.AdsRouterService.RouterService[0]
      ApplicationPath: /app/AdsRouterConsoleApp.dll
      BaseDirectory: /app/
      CurrentDirectory: /app

info: TwinCAT.Ads.AdsRouterService.RouterService[0]
      ASPNETCORE_ENVIRONMENT: Production

info: TwinCAT.Ads.AdsRouterService.RouterService[0]
      AmsTcpIpRouter Status changed to 'Starting'
dbug: TwinCAT.Ads.AdsRouterService.RouterService[0]
      Starting remote listeners for 'RouterExternalChannel' ...
dbug: TwinCAT.Ads.AdsRouterService.RouterService[0]
      Starting remote listening on 172.17.0.2:48898
info: TwinCAT.Ads.AdsRouterService.RouterService[0]
      AmsTcpIpRouter listening to external Endpoint: 172.17.0.2:48898.
dbug: TwinCAT.Ads.AdsRouterService.RouterService[0]
      [AmsTcpIpRouter] 'RouterExternalChannel' Start Listener
info: TwinCAT.Ads.AdsRouterService.RouterService[0]
      Loopback Listener 'RouterLoopbackChannel' (172.17.0.2:48900) starting ...
info: TwinCAT.Ads.AdsRouterService.RouterService[0]
      AmsTcpIpRouter Status changed to 'Started'
dbug: TwinCAT.Ads.AdsRouterService.RouterService[0]
      [AmsTcpIpRouter] 'RouterLoopbackChannel' Start Listener
info: Microsoft.Hosting.Lifetime[0]
      Application started. Press Ctrl+C to shut down.
```

```
      Content root path: /app
dbug: Microsoft.Extensions.Hosting.Internal.Host[2]
      Hosting started
```

Cont.

BECKHOFF

1. Switch to a new tty console
    1. Use Ctrl+Alt+F# to switch to up to 6 consoles. (i.e. Ctrl+Alt+F2 will switch to console 2. We are in console 1 by default)
2. Start the AdsClientConsole container.
    1. docker run -it --rm --env "AmsConfiguration:LoopbackAddress=172.17.0.2" --env "AmsConfiguration:LoopbackPort=48900" ads-cli-client -v 5.76.88.215.1.1 'INT' 'MAIN.nCounter' '16'

```
nathan@debian:~$ sudo docker run -it --rm --env "AmsConfiguration:LoopbackAddress=172.17.0.2" --env "AmsConfiguration:LoopbackPo
rt=48900" ads-cli-client -v 5.19.253.38.1.1 'INT' 'MAIN.nCounter' '16'
2022-04-13T20:05:53.8710781Z: AmsConfiguration:LoopbackAddress=172.17.0.2
2022-04-13T20:05:53.9196185Z: AmsConfiguration:LoopbackPort=48900
2022-04-13T20:05:53.9260131Z: AMS router endpoint set to: 172.17.0.2:48900
2022-04-13T20:05:53.9526227Z: ADS client will connect to ADS service: 5.19.253.38.1.1:851
2022-04-13T20:05:54.4119105Z: ADS client connected to ADS service: 5.19.253.38.1.1:851
2022-04-13T20:05:54.4232478Z: Expected size of type int is 2 bytes
2022-04-13T20:05:54.4336554Z: Converted value 16 of type int into buffer: 10-00
2022-04-13T20:05:54.4362536Z: write buffer: 10-00
2022-04-13T20:05:55.1547760Z: Will write symbol MAIN.nCounter of type int with value 16
2022-04-13T20:05:55.1921902Z: Symbol successfully written
nathan@debian:~$
```

Cont.

BECKHOFF

- To read from TwinCAT remove the final argument:

```
docker run -it --rm --env "AmsConfiguration:LoopbackAddress=172.17.0.2" --env "AmsConfiguration:LoopbackPort=48900" ads-cli-client -v 5.76.88.215.1.1 'INT' 'MAIN.nCounter'
```

```
nathan@debian:~$ sudo docker run -it --rm --env "AmsConfiguration:LoopbackAddress=172.17.0.2" --env "AmsConfiguration:LoopbackPo
rt=48900" ads-cli-client -v 5.19.253.38.1.1 'INT' 'MAIN.nCounter'
2022-04-13T20:10:02.5552685Z: AmsConfiguration:LoopbackAddress=172.17.0.2
2022-04-13T20:10:02.6006648Z: AmsConfiguration:LoopbackPort=48900
2022-04-13T20:10:02.6056431Z: AMS router endpoint set to: 172.17.0.2:48900
2022-04-13T20:10:02.6272625Z: ADS client will connect to ADS service: 5.19.253.38.1.1:851
2022-04-13T20:10:02.7987503Z: ADS client connected to ADS service: 5.19.253.38.1.1:851
2022-04-13T20:10:02.8049098Z: Expected size of type int is 2 bytes
2022-04-13T20:10:02.8114469Z: Will read symbol MAIN.nCounter of type int with size 2
2022-04-13T20:10:02.8147244Z: Expected size of type int is 2 bytes
2022-04-13T20:10:03.0768996Z: Received data: E7-60
2022-04-13T20:10:03.0819450Z: Buffer data:System.Byte[]
2022-04-13T20:10:03.0854531Z: Converted data: 24807
24807
```

Wrap-up

BECKHOFF

- In the event that the vm crashes, or is exited improperly, it will be necessary to destroy the current instance from the host.
- The debian-vm.sh script has a function for removing the configurations it added and destroy the current instance of the vm. Call it with the destroy argument:

```
doas sh debian-vm.sh destroy
```

Wrap-up Cont.

BECKHOFF

- Next delete the lines we added to /etc/rc.conf and /etc/pf.conf.d/bhf
- Then enter the command

```
zfs destroy zroot/Debian-vm_disk0
```