



Wi-Fi automation lab

Pre-lab tasks

Andreas Koksrud / Telenor Bedrift

Co-authors:

Kjetil Teigen Hansen & François Vergès

References / inspiration

<https://github.com/CiscoDevNet/yangsuite/>

<https://postman.com>

https://docs.ansible.com/ansible/latest/collections/cisco/ios/ios_facts_module.html

<https://www.wifireference.com/2020/01/14/viewing-network-telemetry-from-the-catalyst-9800-with-grafana/>

<https://grafana.com/grafana/dashboards/13462-device-health-monitoring/>

<https://grafana.com/grafana/dashboards/12468-catalyst-9800-client-stats/>

<https://wirelessisfun.wordpress.com/2020/12/10/network-telemetry-data-and-grafana-part-1-the-advanced-netconf-explorer/>

<https://python.org>

<https://codeium.com>

<https://canonical.com/multipass>



Copyright

© Andreas Koksrud 2025. All rights reserved. This presentation is provided for educational and informational purposes only. You may distribute and learn from this presentation, but commercial use or any form of monetization is strictly prohibited without prior written consent

Any slides marked Kjetil Teigen Hansen or the Conscia logo will also have full or co-ownership by Kjetil

Any slides marked François Vergès or the SemFio logo will also have full or co-ownership by François



Prerequisites

- Cisco Meraki account (<https://dashboard.meraki.com>)
- Juniper MIST account (<https://manage.mist.com>)
- Postman account (<https://postman.com>)
- Complete the pre-lab exercises (this document) before the deep dive labs
- Bring an Ethernet dongle if you don't have built-in port
- (optional) Bring an extra screen to show lab guide (or prepare to Alt-Tab)



Communications

- WebEx space: Wi-Fi automation lab
- Please help each other
- Sharing is caring 😊



Agenda

Pre-lab

- Choose your hypervisor
- Install Ubuntu Server w/Docker
- Install Postman
- Install VS Code
- (optional) install 9800-CL

Day 1

- Sort out pre-lab task problems
- Get to know the lab environment
- Connect VS Code to Ubuntu
- Install and explore Ansible
- Explore Python automation
- Install and explore YANG Suite
- Explore Postman
- Install and explore Grafana

Day 2

- In-depth explore a topic of choice
 - Grafana / TIG-stack
 - Grafana Cloud
 - Ansible
 - Python
 - Cloud vendor automation (MIST or Meraki)

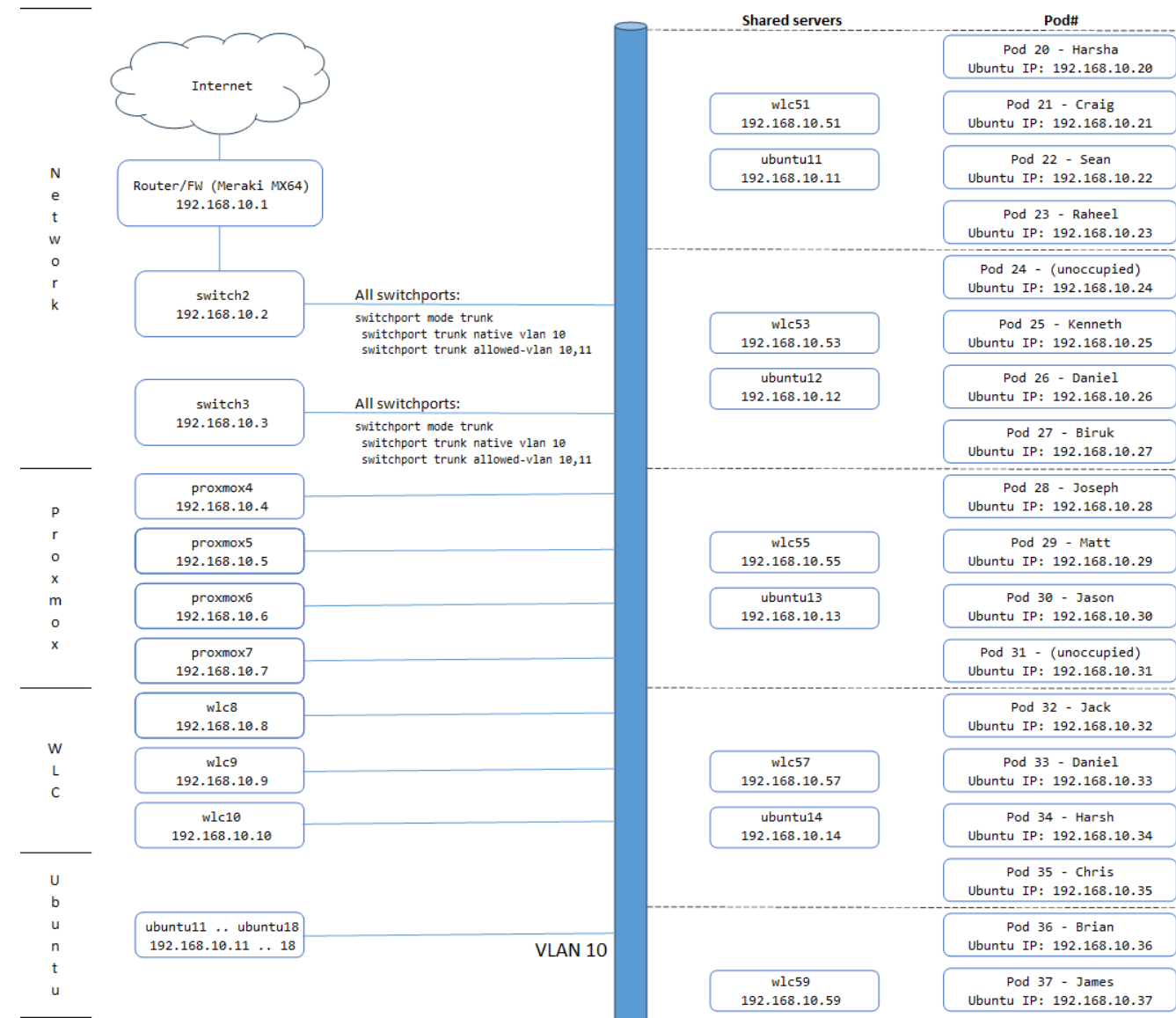


Scope

- In scope
 - Getting started with various systems/languages/solutions for lab purposes
 - Set up your own Ubuntu Linux server on your own laptop. It will be possible to use a shared server if you do not want or have possibility to install an Ubuntu VM on your laptop
 - Some nice examples to try various aspects of automation
 - Inspiring you to explore deeper on your own
- Out of scope topics
 - Git
 - Learning the languages (Ansible, Python, InfluxQL, etc)
 - Learning Linux
 - Deploying the systems for production use
 - Troubleshooting WLC/AP connection



Wi-Fi Automation deep dive - Lab topology



- Each student have an assigned Pod number. When using static IP, their Ubuntu Server should have the same last octet as the Pod#
- 2 students share a preconfigured WLC, as assigned in the topology map
- Everyone is connected to VLAN 10
- All switchports on all switches are trunk ports with VLAN 10 as native, and allowed vlan as VLAN 10 and 11
- For Wi-Fi clients on your SSIDs you can use VLAN 11 to not fill up VLAN 10

Login to shared devices

User: devnet-adm

Pass: ChangeMe2025!

IP Plan (VLAN 10)

Static adm IPs: 192.168.10.1 - 19

Ubuntu (per pod): 192.168.10.20 - 50

WLCs (shared): 192.168.10.51 - 70

DHCP range: 192.168.10.71 - 250

IP Plan (VLAN 11)

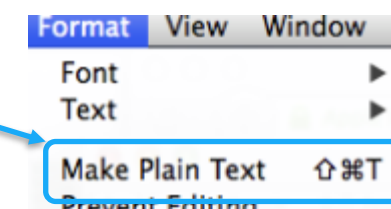
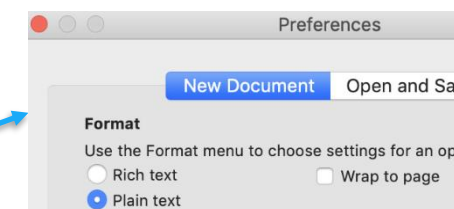
Static adm IPs: 192.168.11.1 - 10

DHCP range: 192.168.10.11 - 250



Notes for Mac users - TextEdit

- TextEdit (the default text editor on OS X) use rich text formatting by default. This is not a great idea when copy-pasting stuff to the command line, so there is a couple of things you must keep in mind
 - If you want, you can change a setting in TextEdit to make all new documents plain text
 - To change the current document to plain text go to Format -> Make Plain Text
 - Shortcut is: Shift-Cmd-T



Building your own lab

- Based on feedback from running this deep dive on previous events, if you are going to build and troubleshoot the full lab setup, it will take most or all of the time (2x 3h)
- The deep dive itself will have pre-built and pre-configured+tested WLAN Controllers (1 WLC per 2-4 students) for you to use. If you want to install your own WLC by following the optional part of this pre-lab guide that is OK, but we strongly encourage you to use the deep dive time to do actual automation exercises and not troubleshooting your WLC
- There will also be pre-built Ubuntu servers to use, running on our servers, for those who want to use that instead of their own laptop.
- If you have a mini computer or an old laptop to spare, you could install Proxmox on that and use for your VMs. Unfortunately it is outside the scope of this deep dive and this pre-lab guide to give detailed instructions on the Proxmox installation itself, outside what is described in task 1d. It is also outside the scope of the deep dive to help troubleshoot your Proxmox. But our servers run on that, so we are somewhat familiar with it and might answer some simple questions 😊
- We will also provide pre-built images with a finished installation of Ubuntu Server for Hyper-V and Virtualbox
- We encourage you to install your own Ubuntu Server, since it will give you more experience building the items yourself when you get home or need it in some other situation.



Pre-lab task #1: Choose your Hypervisor

- The lab itself will use 9800 WLCs that already run on our servers, which have Proxmox Virtual Environment installed. That is a Level 1 hypervisor (run directly on the hardware)
- This chapter will have separate instructions for using Hyper-V (1a), VirtualBox (1b), Multipass/qemu (1c) and Proxmox (1d). Corresponding instructions will also be given in chapter 2 (a/b/c/d) for Ubuntu Server installation, and chapter 6 (a/b/d) for WLC 9800-CL installation
- On Mac, 9800-CL will not be possible, so Mac users will either need a separate server, or (preferably) use our pre-built (and tested) 9800-CL installations
- Hyper-V (on Windows), VirtualBox and VMWare Workstation and Fusion are examples of level 2 hypervisors. That mean that they are running on top of another OS. They will be slower than level 1 hypervisors like Proxmox, Nutanix, VMWare ESXi, Hyper-V server etc. There are some differences between how long the "path to the hardware" is, which might explain some differences in speed using the alternatives.
- **!!! Note !!!**
 - You should turn OFF power save stuff on your laptop, as you might run into unexpected problems if your laptop enters power saving mode
 - You should always do a shutdown of your VMs before shutting down or putting your laptop in sleep or hibernation mode. Especially the 9800-CL do NOT like if the host OS sleeps, and might be corrupt so you have to reinstall the WLC



Pre-lab task #1: Choose your Hypervisor

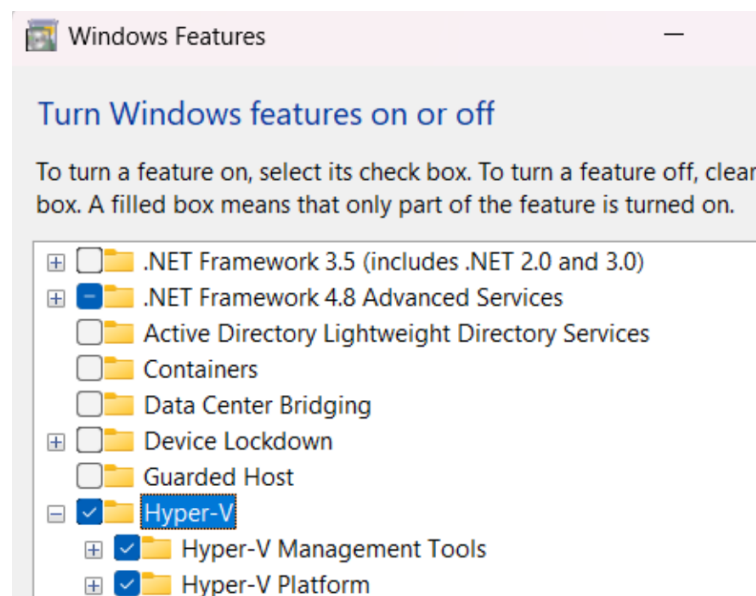
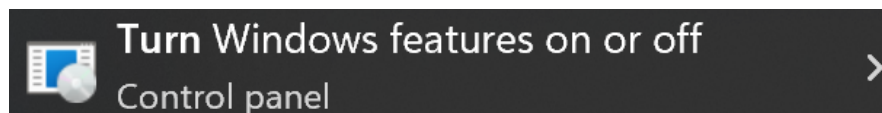
- My experience is that on Windows, virtual machines in Hyper-V are faster than in VirtualBox. This might be due to suboptimal config, differences in hardware, or that Hyper-V have a shorter "path to the hardware"
- If you have no preference, this is my recommendation for this deep dive:
 - If you use Windows: Install Ubuntu Server in Hyper-V
 - If you use Mac: Install Ubuntu Server using Multipass (w/qemu)
 - WLC 9800-CL: Use our shared pre-built & tested servers referring to your Pod# in the Lab Topology map
- For the rest of this chapter, and for chapter 2, follow the instructions for your chosen hypervisor

!!! Note - Many of these instructions require admin privileges. You should be in control of what you do so you don't break anything. We can give no guarantees. The instructions and screenshots are from the working solution on our own laptops, but we have no way to tell if it works on all setups and which conflicting software you might have. If you do not want to do this on your laptop, you can use our pre-built and tested Ubuntu Servers and WLCs !!!



Pre-lab task #1a: Hyper-V

- If you want to use Hyper-V on Windows as your hypervisor, follow this task. When done, jump to task 2a, for how to install Ubuntu on Hyper-V.
- Hyper-V can also run WLC 9800-CL. If you want to try this, you can follow the steps in the optional task 6a
- Full instructions on how to enable/install Hyper-V, or to troubleshoot this, is outside the scope of this lab guide. But in short, go to "Turn Windows Features on or off" and enable Hyper-V

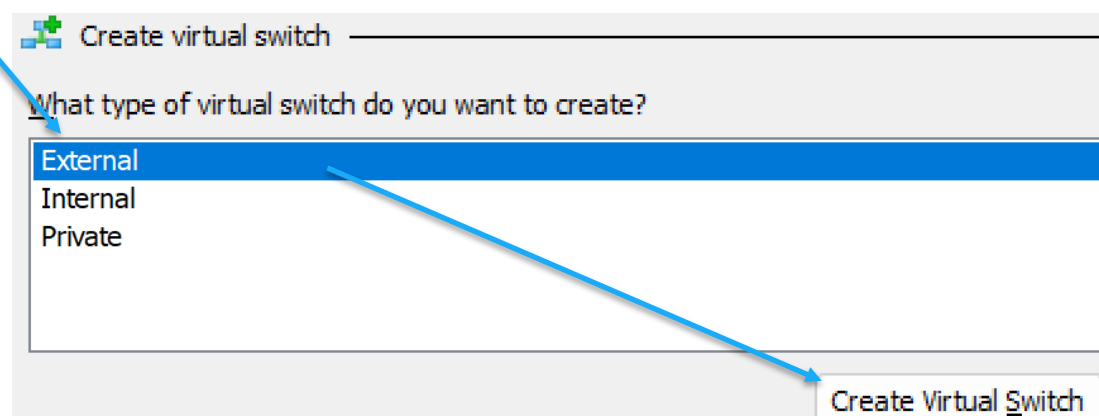
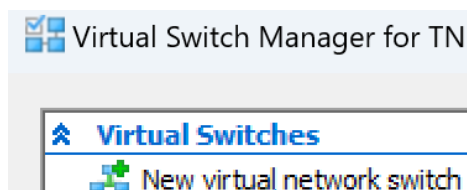
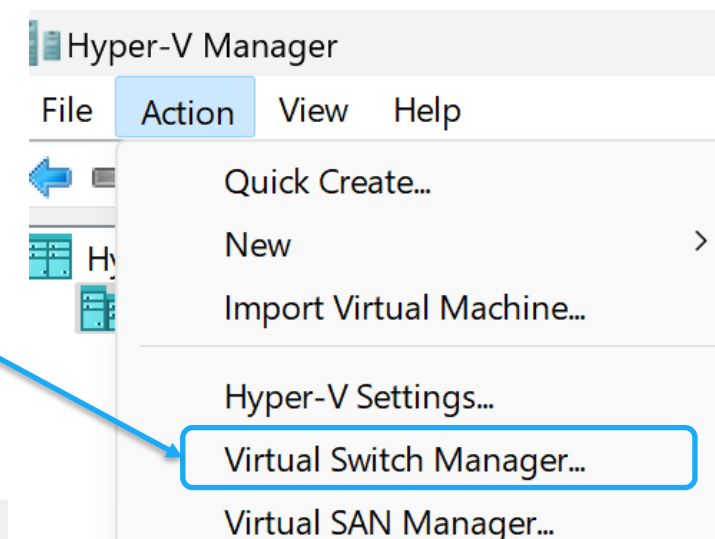
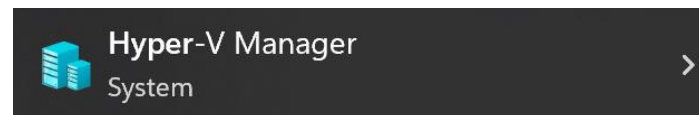


- Reference: <https://learn.microsoft.com/en-us/virtualization/hyper-v-on-windows/quick-start/enable-hyper-v#enable-the-hyper-v-role-through-settings>



Pre-lab task #1a: Hyper-V - Virtual Switch Manager

- Start by opening Hyper-V manager from the Start Menu
- To create virtual switches (Networks) to connect your Ubuntu and 9800-CL, select "Virtual Switch Manager..."
- You want to create an "External" switch. This is similar to "Bridged Adapter" in VirtualBox.



Pre-lab task #1a: Hyper-V - Virtual Switch Manager

- Create the external switch as shown on the screenshot

- Use a descriptive name

- Select your wired ethernet adapter

- Enable this setting

- Click OK

Virtual Switch Properties

Name:
External (LAN)

Notes:

Connection type
What do you want to connect this virtual switch to?

☒ External network:
Intel(R) Ethernet Connection (17) I219-LM

☒ Allow management operating system to share this network adapter

☐ Internal network
☐ Private network

OK Cancel Apply



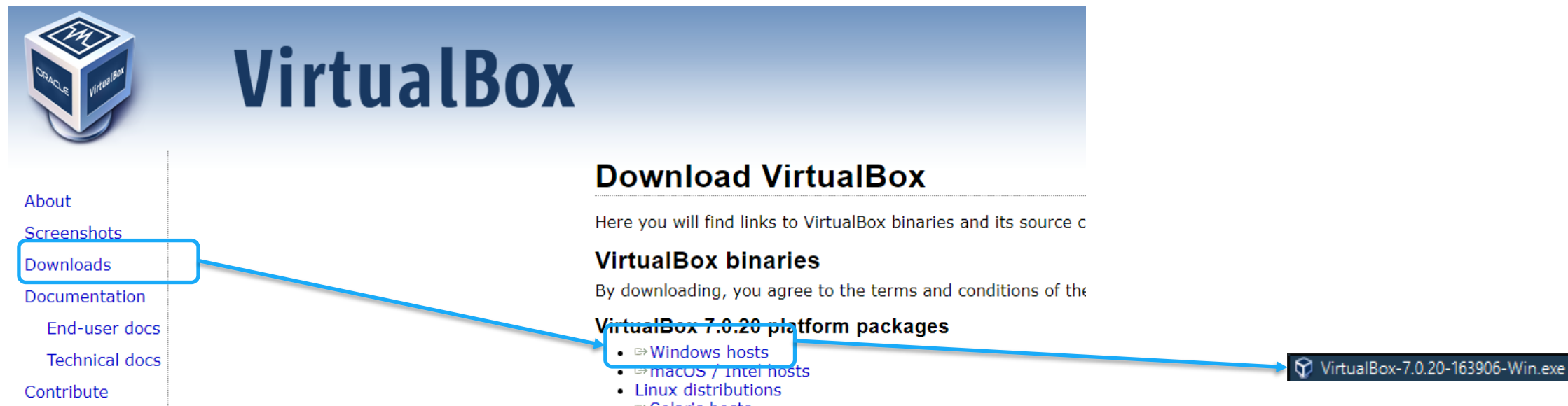
Pre-lab task #1b: VirtualBox

- If you want to use VirtualBox on Windows as your hypervisor, follow this task. When done, jump to task 2b, for how to install Ubuntu on VirtualBox
- VirtualBox can also run WLC 9800-CL. If you want to try this, you can follow the steps in the optional task 6b
- RightCtrl + F switches between full-screen and window
- Tap RightCtrl to "free" input from guest



VirtualBox installation

<https://virtualbox.org/>



The screenshot shows the VirtualBox website. On the left, a navigation menu includes 'About', 'Screenshots', 'Downloads' (highlighted with a blue box), 'Documentation', 'End-user docs', 'Technical docs', and 'Contribute'. A blue arrow points from the 'Downloads' link to the 'VirtualBox binaries' section. In this section, another blue box highlights the 'Windows hosts' link in the 'VirtualBox 7.0.20 platform packages' list. A second blue arrow points from this link to the download file 'VirtualBox-7.0.20-163906-Win.exe'.

Download VirtualBox

Here you will find links to VirtualBox binaries and its source c

VirtualBox binaries

By downloading, you agree to the terms and conditions of the

VirtualBox 7.0.20 platform packages

- [Windows hosts](#)
- [macOS / Intel hosts](#)
- [Linux distributions](#)
- [Solaris hosts](#)

[VirtualBox-7.0.20-163906-Win.exe](#)

- Needs admin privileges
- Not all steps are shown here, it is mostly "Next-next-next"
- As default folder for VMs, I use "C:\Virtual Machines", you can use whatever, but screenshots here will show this path



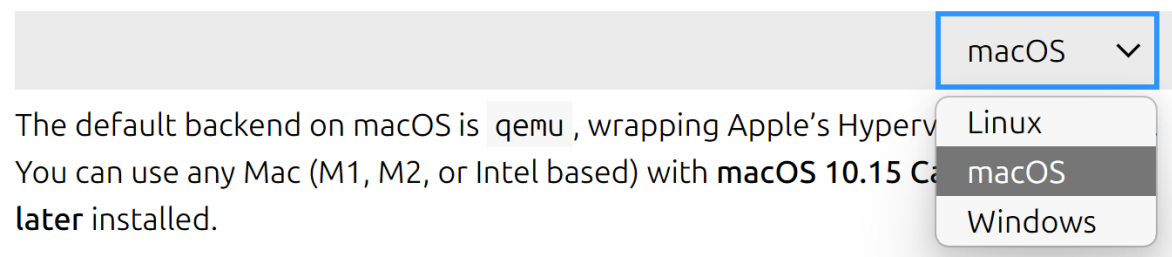
VM Networking

- Here is a general overview of different modes of networking
- **Bridged Adapter (used in this lab)**
 - Each VM gets an IP alongside your host, directly on the network
 - Switchports must accept multiple mac addresses from each host
 - For APs to connect to 9800, I prefer this rather than NAT'ing through your host
 - Also, for WLCs to connect to Ubuntu (for telemetry), I also prefer this over NAT'ing
- NAT Network (not covered in this lab)
 - Host IP is shared as outside IP
 - VMs get IP addresses on a NAT'ed network pool
 - NAT Network is created under File -> Tools -> Network Manager -> NAT Networks
 - You must create Port Forwarding rules (in the NAT Networks config) to allow traffic from host to guest VMs
- NAT (not covered in this lab)
 - Host IP is shared as outside IP
 - VM can only reach internet, not the other VMs
 - Host and guest can not reach each other on IP, only the VirtualBox console



Pre-lab task #1c: Multipass (Mac)

- On Mac we recommend using Canonical Multipass for deploying the Ubuntu Server. It will use qemu as backend. Follow this task to install Multipass and jump to task 2c, for how to install Ubuntu. Qemu should be enabled on Mac as default
- Full instructions on how to install Multipass, or to troubleshoot this, is outside the scope of this lab guide. But the tutorial in this link are very good:
<https://canonical.com/multipass/docs/install-multipass>
- Select "macOS" in the drop down menus



Pre-lab task #1c: Multipass (Mac)

- Verify that Multipass is installed and working

```
akoksrud:MacOS ~$ multipass --version
multipass 1.15.0+mac
multipassd 1.15.0+mac
```

- Once multipass is installed, you can launch an Ubuntu server with this command. You can test this if you want, or you can skip to section 2c where you will install the server to use in the deep dive with some more options like memory, disk, cpu, networking etc

```
akoksrud:MacOS ~$ multipass launch --name testserver
Launched: testserver
```

- To list your servers

```
akoksrud:MacOS ~$ multipass list
```

Name	State	IPv4	Image
ubuntu-devnet	Running	192.168.64.9	Ubuntu 24.04 LTS

- Start, connect to shell, stop and delete your servers

```
akoksrud:MacOS ~$ multipass start ubuntu-devnet
akoksrud:MacOS ~$ multipass shell ubuntu-devnet
akoksrud:MacOS ~$ multipass stop ubuntu-devnet
akoksrud:MacOS ~$ multipass delete ubuntu-devnet
akoksrud:MacOS ~$ multipass purge
```



Pre-lab task #1c: Multipass (Mac)

- Instruct Multipass to use your wired adapter as bridge adapter. Using wireless adapter with or without bridging is also possible, but for the deep dive lab it will be less room for error if you stick with wired for now

```
akoksrud:MacOS ~$ multipass networks
Name      Type      Description
en0       wifi      Wi-Fi
en4       ethernet  Ethernet Adapter (en4)
en5       ethernet  Ethernet Adapter (en5)

akoksrud:MacOS ~$ multipass set local.bridged-network=en4
```

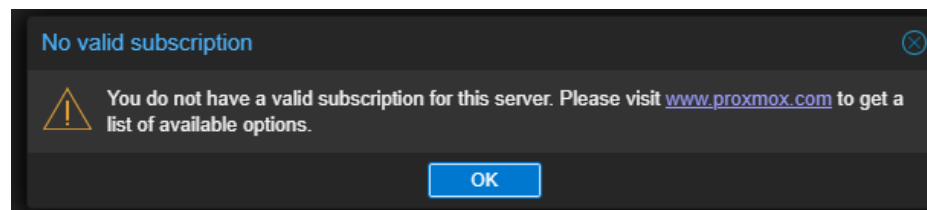
- Continue to section 2c where we will use the bridged adapter when we create a Ubuntu Server



Pre-lab task #1d: Proxmox (separate hardware)

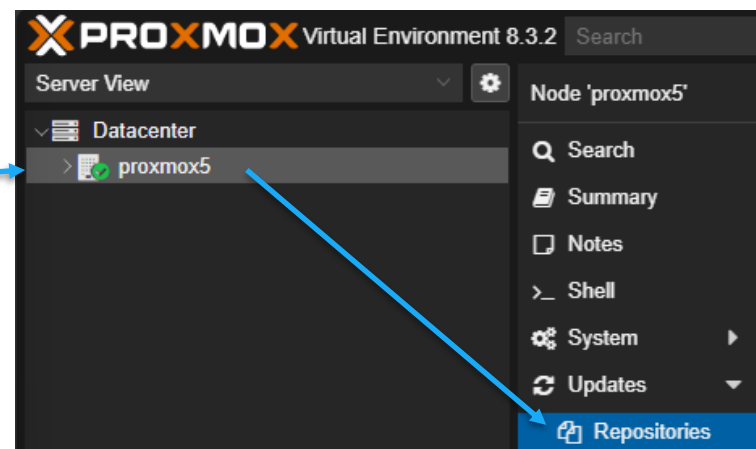


- If you want to use Proxmox as your hypervisor, follow this task. When done, jump to task 2d, for how to install Ubuntu on Proxmox
- The "portable datacenter" that run the student VMs in this lab, run on Proxmox Virtual Environment
- If you have an old laptop (with sufficient RAM, disk and CPU), or maybe an Intel NUC or other mini computer, this is an excellent alternative
- Full instructions on how to install Proxmox, or to troubleshoot this, is outside the scope of this lab guide. For installation instructions, see <https://proxmox.com/>
- For production use, licenses should be aquired. For lab work, support won't be that critical, so you could run without license. Proxmox itself is open source, but the company earns their well deserved cash by offering various levels of support agreements/licensing. You will get this box each time you log in to to the proxmox web page

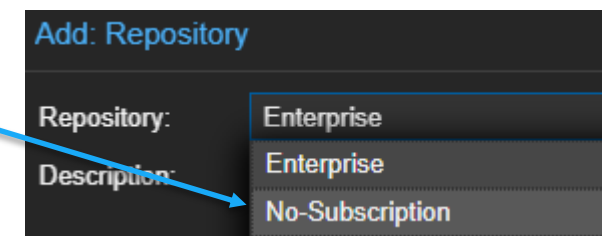
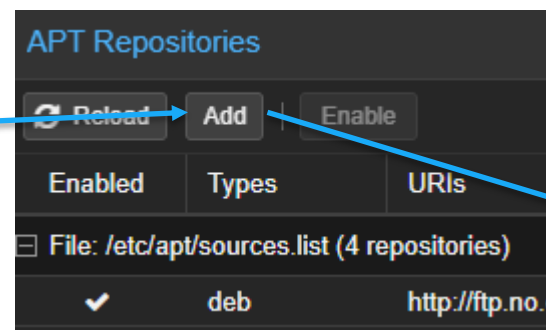


Pre-lab task #1d: Proxmox - Enable NoSub repo

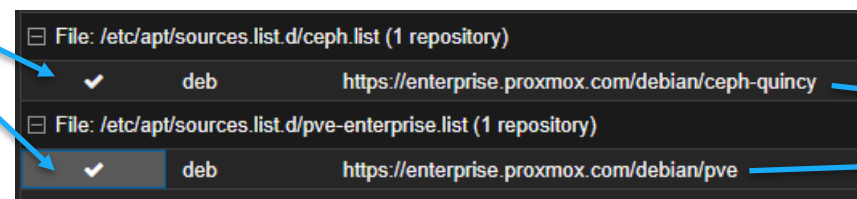
- To update/upgrade Proxmox (without subscription / Community Edition), go to your node {server name} -> Updates -> Repositories



- Then Add the No-Subscription repository



- Disable the two enterprise repositories
- You can then update from GUI or from CLI



```
root@proxmox5:~$ apt update
root@proxmox5:~$ apt upgrade
```



Pre-lab task #2: Ubuntu Server

- Like the first section, this is also separated into specific instructions for installing Ubuntu Server on
 - 2a. Hyper-V
 - 2b. VirtualBox
 - 2c. Parallels
 - 2d. Proxmox
- Follow the instructions for your hypervisor, and skip the others
- **!!! After the hypervisor-specific instructions, there are common instructions which are identical no matter which hypervisor you are using !!!**



Pre-lab task #2: Download Ubuntu Server

- Hyper-V, VirtualBox and Proxmox: Download Ubuntu Server 24.04 (default ISO image)

- <https://ubuntu.com/download/server>

Ubuntu Server

This is the default ISO image of the Ubuntu Server installer.

Download 24.04 LTS

- Parallels (Mac with Apple M-series CPU): Download "Ubuntu Server for ARM"

- <https://ubuntu.com/download/server/arm>

Ubuntu Server for ARM

Ubuntu 24.04.1 [LTS](#) includes support for the very latest ARM-based server systems powered by certified 64-bit processors.

Develop and test using over 50,000 software packages and runtimes — including Go, Java, Javascript, PHP, Python and Ruby — and deploy at scale using our complete scale-out management suite including MAAS and Juju. Ubuntu delivers server-grade performance on ARM, while fully retaining the reliable and familiar Ubuntu experience.

Ubuntu Server

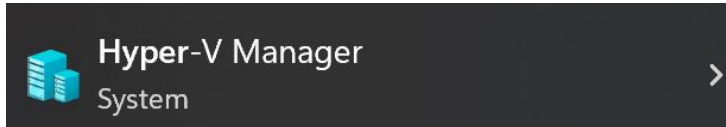
This is the default ISO image of the Ubuntu Server installer.

Download 24.04.1 LTS

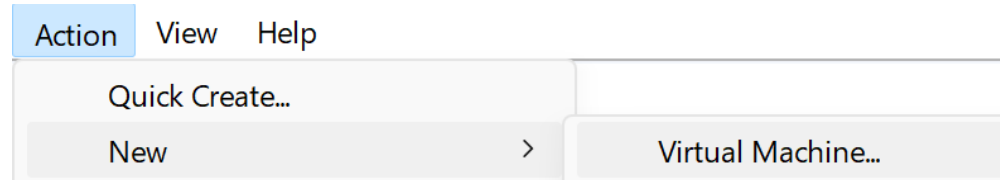


2a: Hyper-V: Create Virtual Machine

- Open "Hyper-V Manager"



- Action -> New -> Virtual Machine...



2a: Hyper-V: Create Virtual Machine

- Click past the first page. On page 2, specify a name. Click Next

Before You Begin	Choose a name and location for this virtual machine.
Specify Name and Location	The name is displayed in Hyper-V Manager. We recommend you identify this virtual machine, such as the name of the guest operating system.
Specify Generation	
Assign Memory	Name: <input type="text" value="ubuntu-devnet"/>

- Specify Generation: Generation 2

Before You Begin	Choose the generation of this virtual machine.
Specify Name and Location	<input type="radio"/> Generation 1 This virtual machine generation supports 32-bit and 64-bit guest operating systems and virtual hardware which has been available in all previous versions of Hyper-V.
Specify Generation	<input checked="" type="radio"/> Generation 2 This virtual machine generation provides support for newer virtualization firmware, and requires a supported 64-bit guest operating system.
Assign Memory	
Configure Networking	
Connect Virtual Hard Disk	

- Assign 4GB memory

Before You Begin	Specify the amount of memory to allocate to this virtual machine. The amount can range from 4 MB through 251658240 MB. To improve performance, we recommend you allocate a multiple of 1 MB.
Specify Name and Location	
Specify Generation	
Assign Memory	Startup memory: <input type="text" value="4096"/> MB <input checked="" type="checkbox"/> Use Dynamic Memory for this virtual machine.
Configure Networking	



2a: Hyper-V: Create Virtual Machine

- On Configure Networking, select the "External (LAN)" vSwitch you created in task 1a

Before You Begin

Specify Name and Location

Specify Generation

Assign Memory

Configure Networking

Each new virtual machine includes a network adapter. You can connect it to a virtual switch, or it can remain disconnected.

Connection: External (LAN)

- For Hard Disk, create a VHDX disk with 40GB

Specify Generation

Assign Memory

Configure Networking

Connect Virtual Hard Disk

Installation Options

Summary

☒ Create a virtual hard disk

Use this option to create a VHDX dynamically expanding

Name: ubuntu-devnet.vhdx

Location: C:\Virtual Machines\

Size: 40 GB (Maximum: 64 TB)

- Installation Options: Use the Ubuntu Server ISO image you downloaded. Then click "Finish"

Assign Memory

Configure Networking

Connect Virtual Hard Disk

Installation Options

☒ Install an operating system from a bootable image file

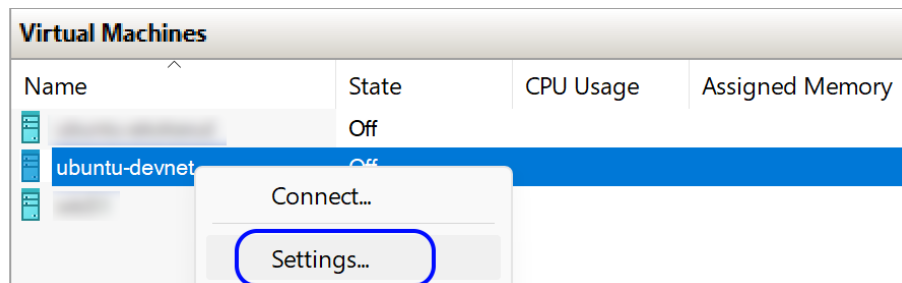
Media

Image file (.iso): C:\Virtual Machines\ubuntu-24.04-live-server-amd64.iso Browse...

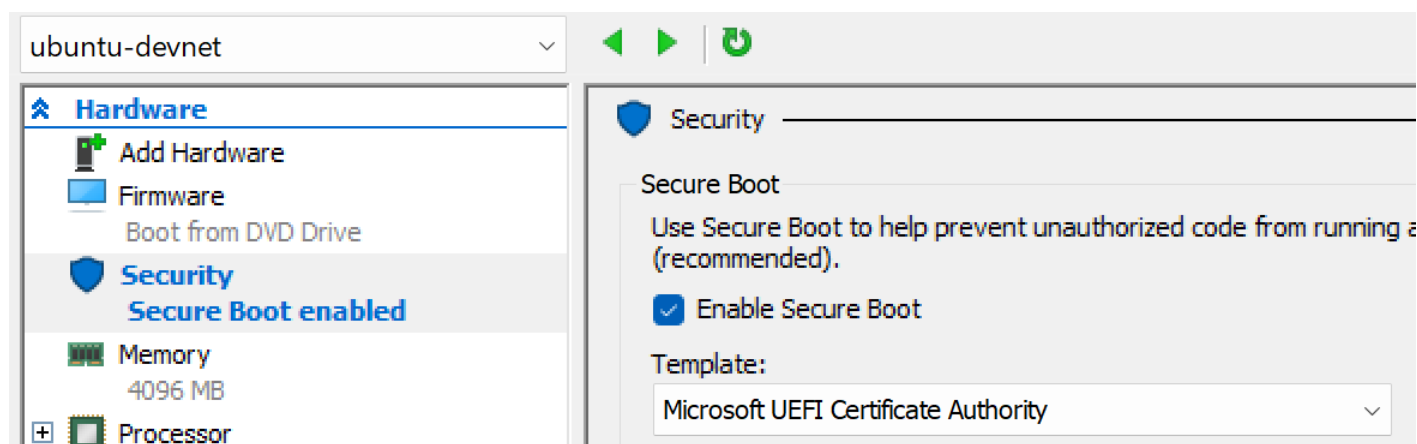


2a: Hyper-V: Create Virtual Machine

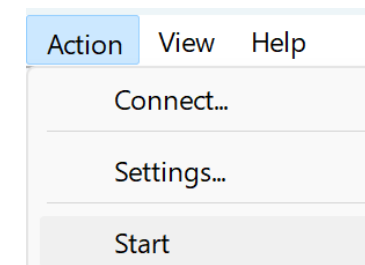
- Right click you new machine, and select "Settings..."



- Under "Security" select Template: Microsoft UEFI Certificate Authority

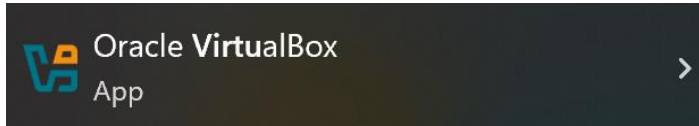


- Start your machine, then skip some slides forward to the common instructions ☺

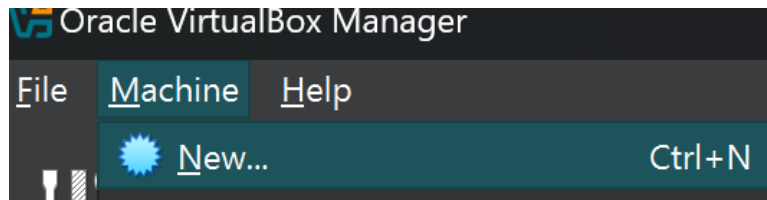


2b: VirtualBox: Create Virtual Machine

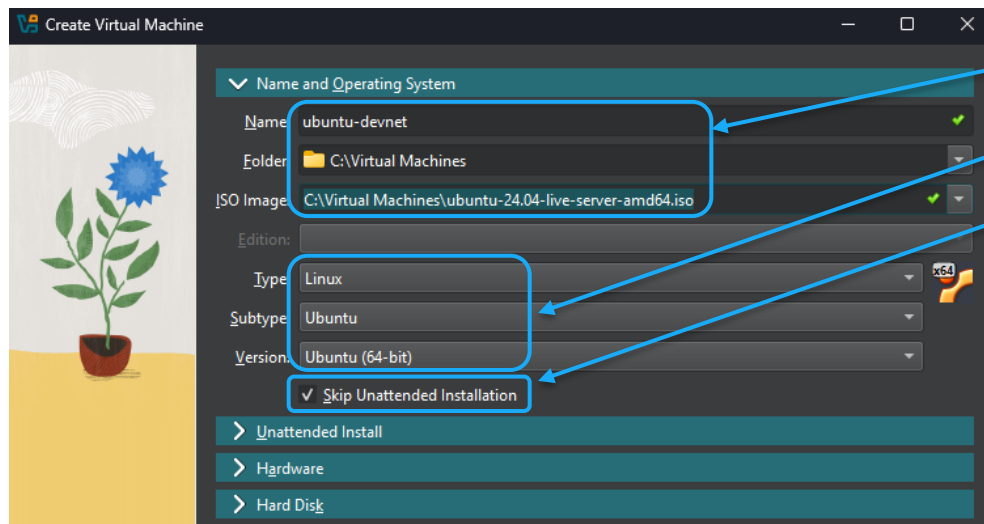
- Open VirtualBox



- Machine -> New



2b: VirtualBox: Create Virtual Machine



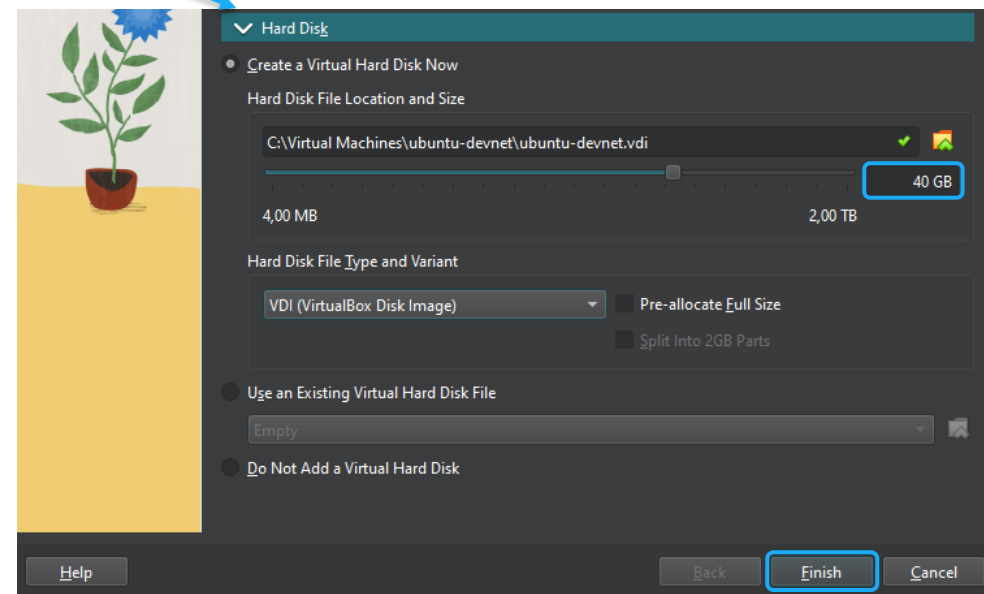
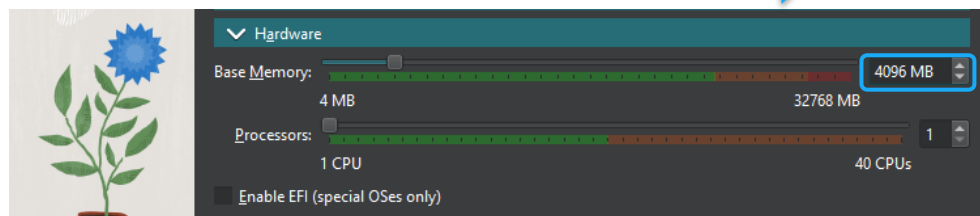
Set the name, folder and select the ISO that you downloaded

Select Type as shown

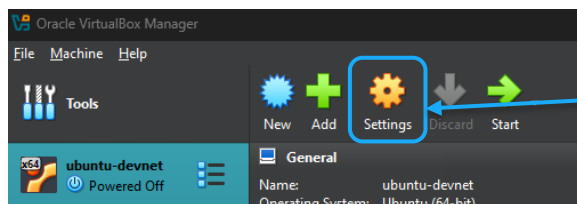
Check «Skip Unattended Installation»

Expand the «Hardware» section. Increase to 4096MB

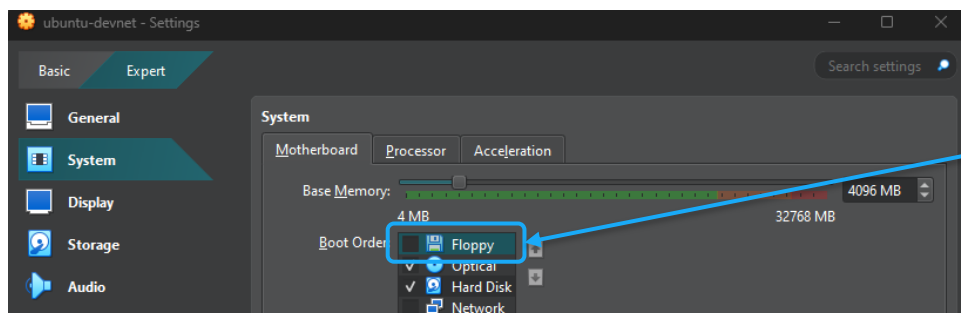
Expand the «Hard Disk» section. Increase to 40GB, then Finish



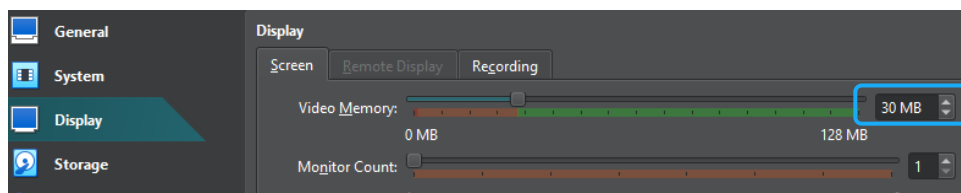
2b: VirtualBox: Create Virtual Machine



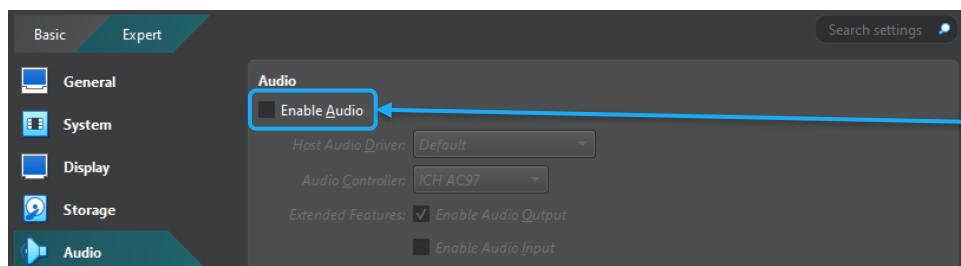
- Enter settings
- General: No change needed



- System: Remove «Floppy» checkbox. Keep the rest default



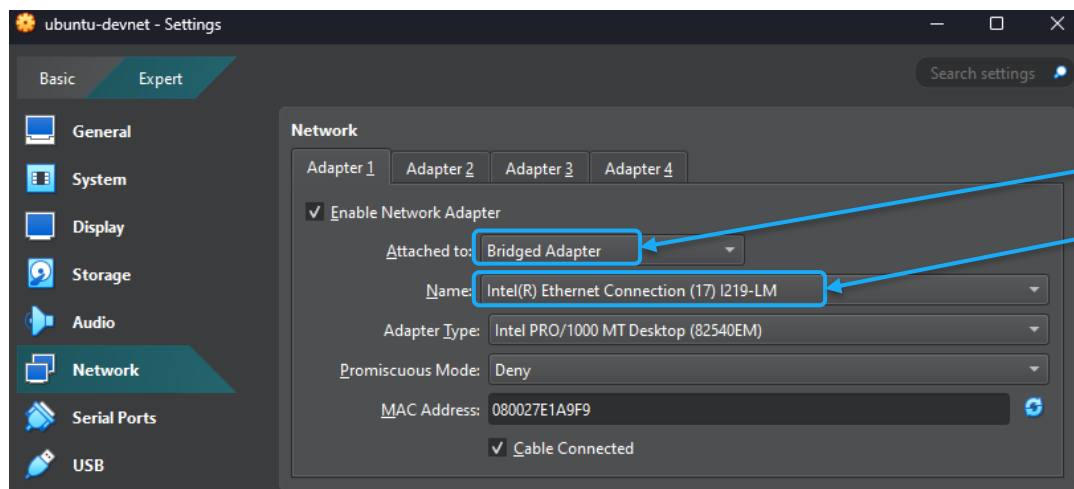
- Display: Ensure «Video Memory» is at least 30MB. Keep the rest default



- Storage: No change needed
- Audio: Remove «Enable Audio» checkbox



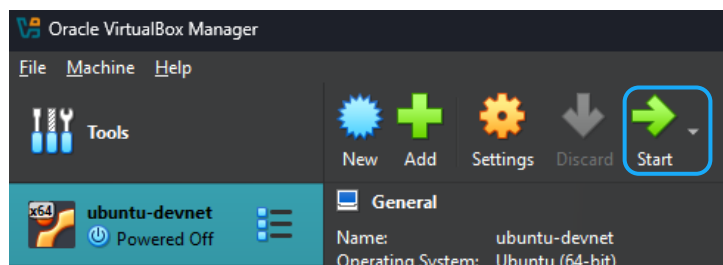
2b: VirtualBox: Create Virtual Machine



- Network:
 - Change «Attached to» for Adapter 1 to «Bridged Adapter»
 - Change «Name», select your Ethernet adapter
 - Keep the rest as default
 - Do not enable Adapter 2, 3 or 4



- Click OK to save the changes



- Start the VM
- (here you can also choose «Headless Start» at a later stage when your VM have got its static IP so you would only use SSH)
- Then, skip some slides forward to the common instructions ☺



2c: Multipass - Create VM

- First, create a file on your laptop, named "cloud-config.yaml"

```
akoksrud@MacOS:~/$ mkdir ubuntu-devnet
akoksrud@MacOS:~/$ cd ubuntu-devnet
akoksrud@MacOS:~/ubuntu-devnet$ nano cloud-config.yaml
```

~/ubuntu-devnet/cloud-config.yaml

```
users:
- default
- name: devnet-adm
  lock_passwd: false
  sudo: ALL=(ALL) NOPASSWD:ALL
  plain_text_passwd: ChangeMe2025!
  shell: /bin/bash
ssh_pwauth: True
package_update: true
package_upgrade: true
```

- From the same folder you created the YAML file, create the server using the following command

```
akoksrud@MacOS:~/ubuntu-devnet$ multipass launch --name ubuntu-devnet --bridged --memory 4G --disk 40G --cpus 2 --cloud-init cloud-config.yaml
Creating ubuntu-devnet /
Configuring ubuntu-devnet /
Starting ubuntu-devnet /
```

- List your VMs

```
akoksrud@MacOS:~/ubuntu-devnet$ multipass list
Name           State      IPv4           Image
ubuntu-devnet  Running   192.168.10.163 Ubuntu 24.04 LTS
```

If your IPv4 says "N/A", you must enter the DNS info in your hosts file manually. See next page.

Info will not show either, only throw a "Failed to resolve" error message.

Primarily seen when running Multipass on Windows. See next page for instructions

- Show VM info

```
akoksrud@MacOS:~/ubuntu-devnet$ multipass info ubuntu-devnet
Name:           ubuntu-devnet
State:          Running
(...)
```



2c: Multipass - Update hosts file (Windows)

- IF you get error message on multipass info, you need to update your hosts file. This is only seen on Windows. Multipass list will show "N/A" on IPv4 address

```
PS ~/ubuntu-devnet> multipass list
Name          State      IPv4      Image
ubuntu-devnet Running    N/A       Ubuntu 24.04 LTS

PS ~/ubuntu-devnet> multipass info ubuntu-devnet
info failed: ssh connection failed: 'Failed to resolve hostname ubuntu-devnet.mshome.net (No such host is known. )'
```

- Find the IP of the VM. On Windows you can open the console of the VM

```
PS ~/ubuntu-devnet> vmconnect.exe $env:computername ubuntu-devnet
(...new window will open...)
ubuntu-devnet login: devnet-adm
Password: ChangeMe2025!
(...cut...)
Memory usage: 7%          IPv4 address for eth1: 192.168.10.163
(...cut...)
devnet-adm@ubuntu-devnet:~$ exit
```

- Open your text editor (Notepad++ in the example) as administrator. Open the file `C:\Windows\System32\drivers\etc\hosts.ics` and add this line (change the IP to your IP, and the hostname to what multipass tries to look up)

```
192.168.10.163 ubuntu-devnet.mshome.net
```



2c: Multipass - Connect to shell

- Connect using SSH

```
akoksrud@MacOS:~/ubuntu-devnet$ ssh devnet-adm@192.168.10.163
devnet-adm@192.168.10.163's password: ChangeMe2025!
Welcome to Ubuntu 24.04.1 LTS (GNU/Linux 6.8.0-51-generic x86_64)
(...)
ubuntu@ubuntu-devnet:~$ exit
```

Change to YOUR Ubuntu IP

- Connect to the shell

```
akoksrud@MacOS:~/ubuntu-devnet$ multipass shell ubuntu-devnet
Welcome to Ubuntu 24.04.1 LTS (GNU/Linux 6.8.0-51-generic x86_64)
(...)
ubuntu@ubuntu-devnet:~$ exit
```

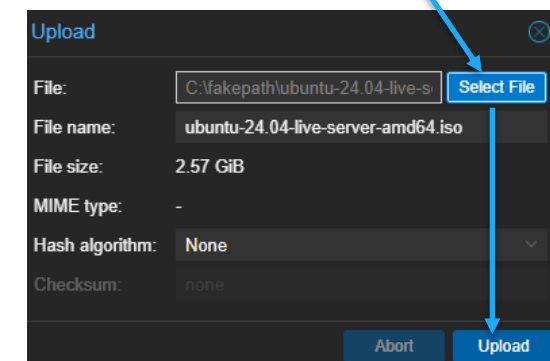
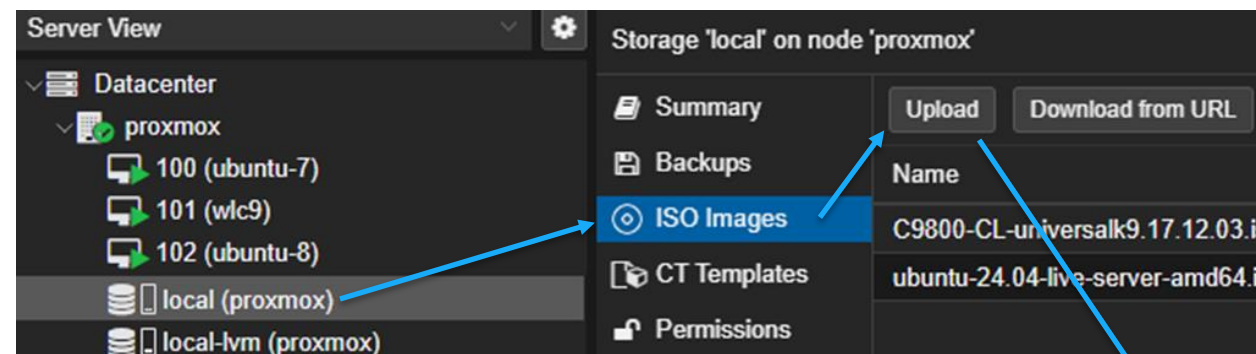
- Follow the relevant parts of the common Ubuntu Server installation instructions. You can of course skip the installation procedure itself, but be sure to do the parts regarding
 - Change keyboard layout
 - Change to static IP
 - Create and import SSH key
 - Clone the example git repository



2d: Proxmox - Create Virtual Machine

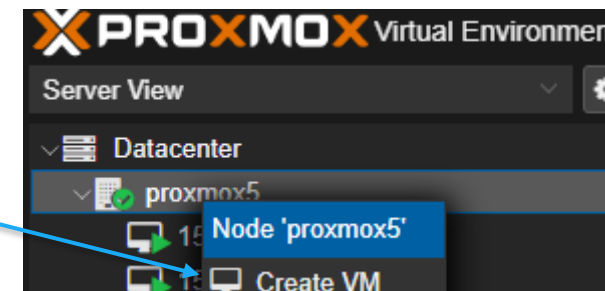
- Open the Proxmox website (<https://{IP}:8006>)
- Start by uploading the Ubuntu Server ISO to the Proxmox datastore:

Go to Datacenter>proxmox>local (proxmox)
Select "ISO images" and "Upload"



2d: Proxmox - Create Virtual Machine

- Right click your proxmox node, and "Create VM"

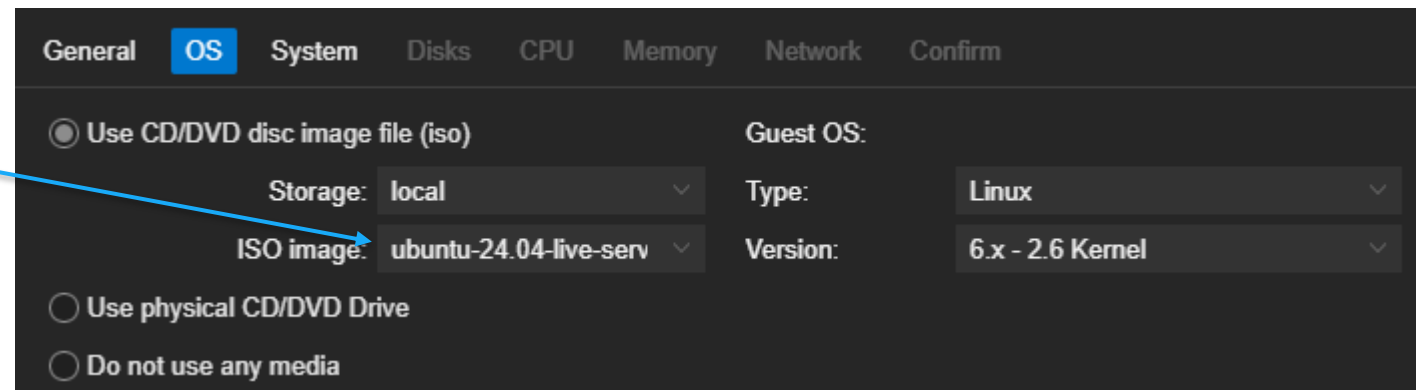


- Set the name and if it should start on Proxmox boot

A screenshot of the 'Create: Virtual Machine' dialog box in Proxmox VE. The 'General' tab is active. The 'Node' is set to 'proxmox5', 'VM ID' is '100', and 'Name' is 'ubuntu13'. The 'Start at boot' checkbox is checked. Other fields include 'Resource Pool' (empty), 'Start/Shutdown order' (set to 'any'), 'Startup delay' (set to 'default'), and 'Shutdown timeout' (set to 'default'). Blue arrows point from the text 'Set the name' to the 'Name' field and from 'and if it should start on Proxmox boot' to the 'Start at boot' checkbox.

2d: Proxmox - Create Virtual Machine

- Under OS, select the ISO image



General OS System Disks CPU Memory Network Confirm

☒ Use CD/DVD disc image file (iso) Guest OS:

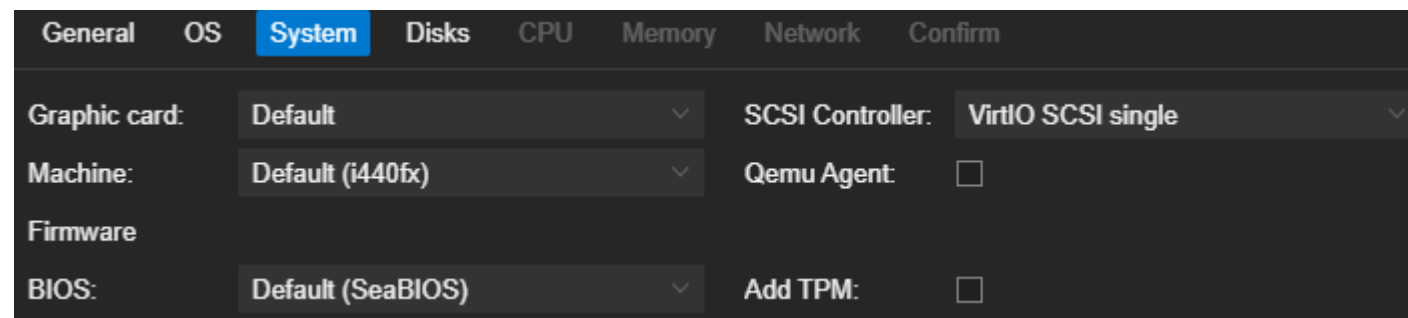
Storage: local Type: Linux

ISO image: ubuntu-24.04-live-serv Version: 6.x - 2.6 Kernel

☐ Use physical CD/DVD Drive

☐ Do not use any media

- Under System keep the defaults



General OS System Disks CPU Memory Network Confirm

Graphic card: Default SCSI Controller: VirtIO SCSI single

Machine: Default (i440fx) Qemu Agent: ☐

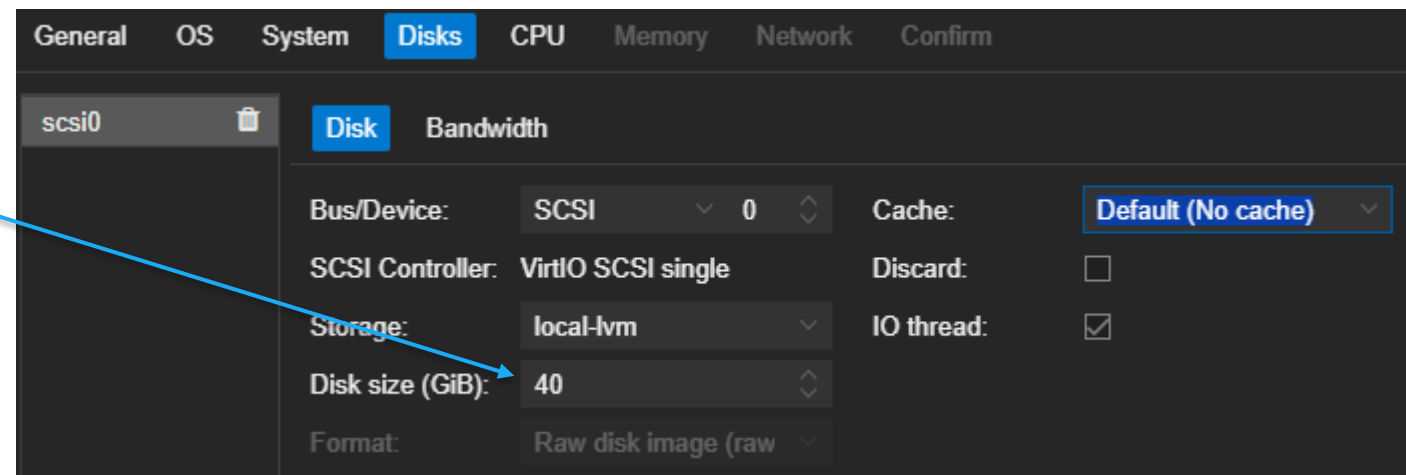
Firmware

BIOS: Default (SeaBIOS) Add TPM: ☐




2d: Proxmox - Create Virtual Machine



- Under Disks, change to 40GB




General OS System **Disks** CPU Memory Network Confirm

scsi0 

Disk Bandwidth


Bus/Device: SCSI 0  Cache: Default (No cache) 


SCSI Controller: VirtIO SCSI single

Storage: local-lvm 

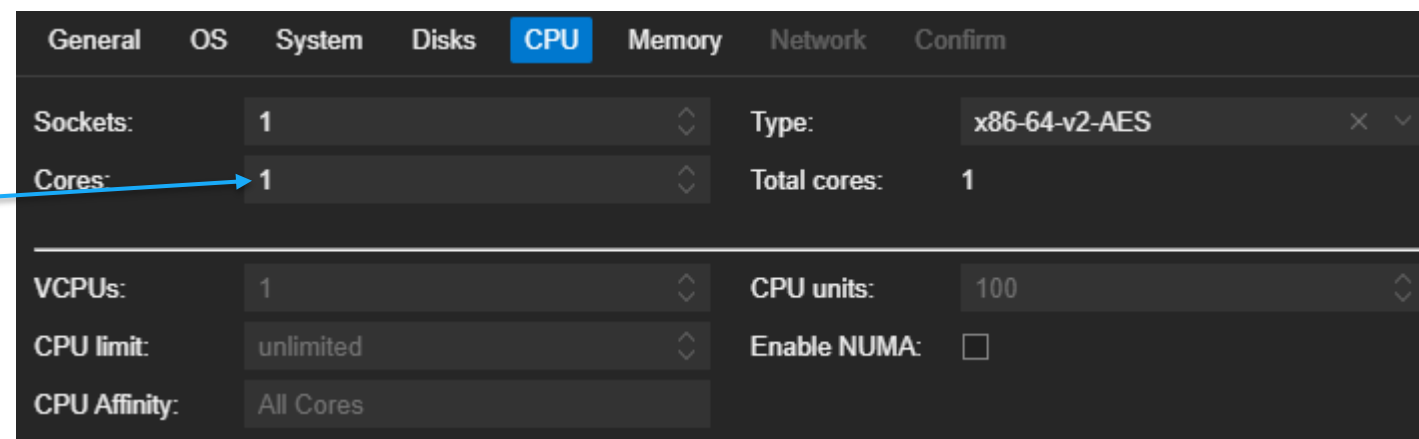
Discard: ☐

IO thread: ☒



Disk size (GiB): 40 


Format: Raw disk image (raw) 



- Under CPU keep the defaults, if you have lots of available resources you could increase the number of cores to 2 or 4




General OS System Disks **CPU** Memory Network Confirm

Sockets: 1  Type: x86-64-v2-AES 

Cores: 1  Total cores: 1

VCPU: 1  CPU units: 100 

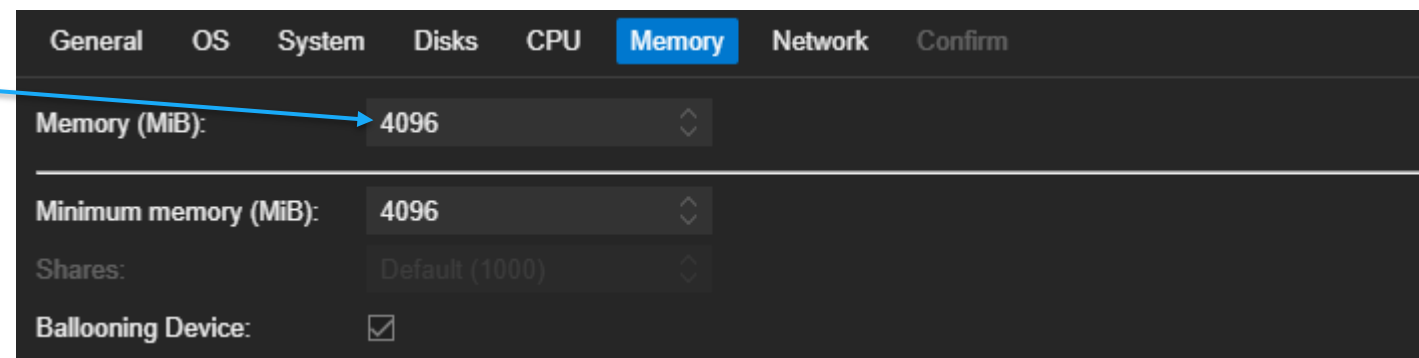
CPU limit: unlimited  Enable NUMA: ☐

CPU Affinity: All Cores



2d: Proxmox - Create Virtual Machine

- Under Memory, change to 4096MiB
If you have limited resources, it would work fine with 2048MiB as well



General OS System Disks CPU **Memory** Network Confirm

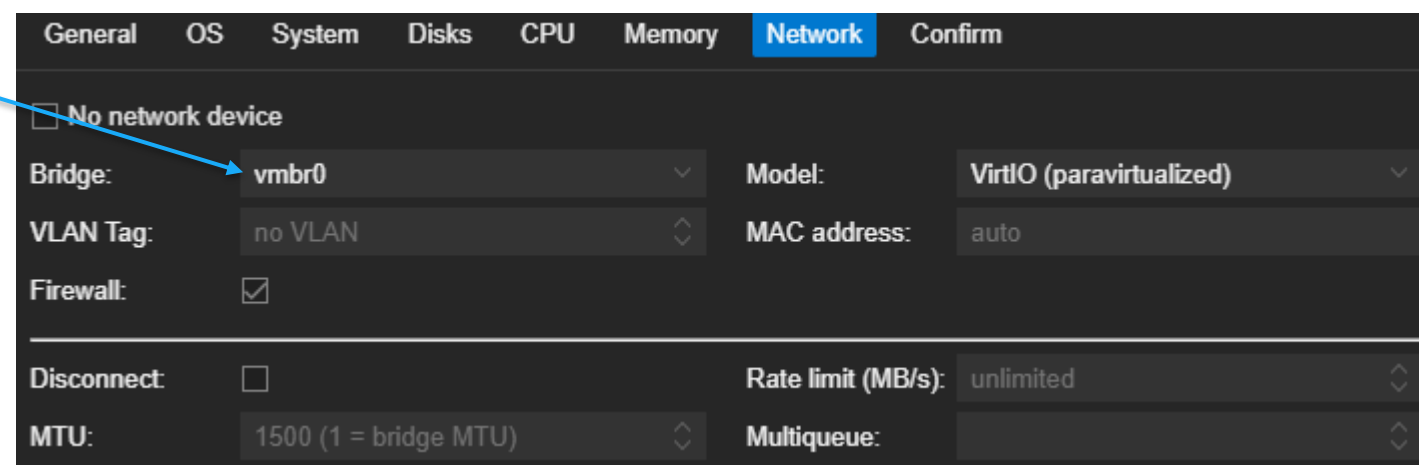
Memory (MiB): 4096

Minimum memory (MiB): 4096

Shares: Default (1000)

Ballooning Device: ☒

- Under Network, select vmbr0



General OS System Disks CPU Memory **Network** Confirm

☐ No network device

Bridge: vmbr0

Model: VirtIO (paravirtualized)

VLAN Tag: no VLAN

MAC address: auto

Firewall: ☒

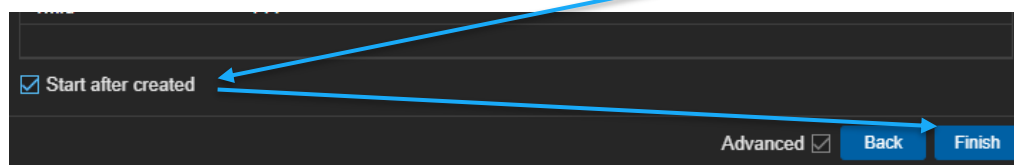
Disconnect: ☐

Rate limit (MB/s): unlimited

MTU: 1500 (1 = bridge MTU)

Multiqueue: [dropdown]

- Confirm and start your VM



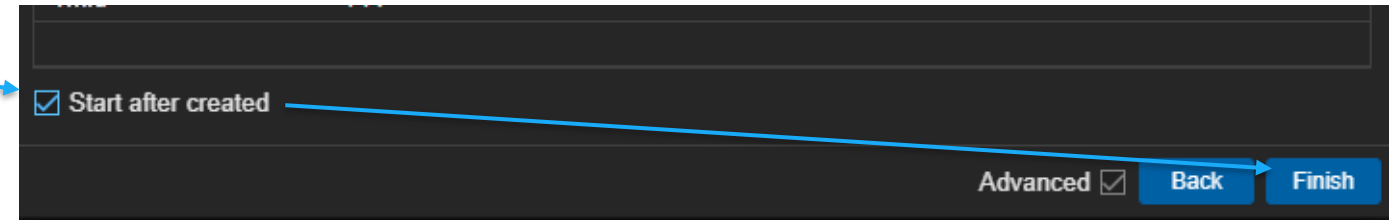
☒ Start after created

Advanced ☒ Back Finish

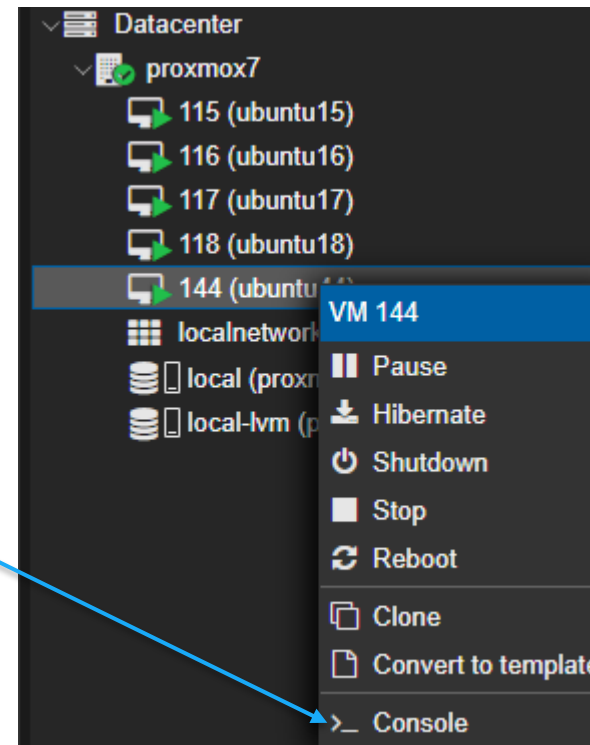


2d: Proxmox - Create Virtual Machine

- Confirm and start your VM



- Connect to the Console of your VM, to install the OS. Right click your VM and select "Console"



Ubuntu server install - Common instructions

- **!!! The following slides are common steps, regardless of the hypervisor you use to run Ubuntu Server !!!**
- If you have installed using a method which skips the installation procedure here (e.g. Multipass, or copying a pre-installed VM host etc), just skip the irrelevant slides, but be sure to at least consider the steps regarding language, ssh keys etc



Ubuntu server install

- Frequent updates to Ubuntu can make the following pages somewhat different, but the basics should be recognizable.
- If asked to update the installer you can do this, it won't really matter much

```
Willkommen! Bienvenue! Welcome! Добро пожаловать! Velkommen!

Use UP, DOWN and ENTER keys to select your language.

[ Asturianu          ]
[ Bahasa Indonesia  ]
[ Català            ]
[ Deutsch           ]
[ English           ]
[ English (UK)      ]
[ Español           ]
[ Français          ]
[ Galego            ]

Keyboard configuration

Please select your keyboard layout below, or select "Identify keyboard" to detect your layout automatically.

Layout: [ Norwegian ]
Variant: [ Norwegian ]

[ Identify keyboard ]

Choose the type of installation

Choose the base for the installation.

(X) Ubuntu Server
    The default install contains a curated set of packages that provide a comfortable experience for operating yo

( ) Ubuntu Server (minimized)
    This version has been customized to have a small runtime footprint in environments where humans are not expecte
    in.

Additional options

[ ] Search for third-party drivers
```

```
Installer update available

Version 24.10.1 of the installer is now available (24.04.1 is currently running).

[ Update to the new installer ]
[ Continue without updating  ]
[ Back                       ]
```

- Choose your own keyboard layout (unless you just love the norwegian æøå characters and funky placement of all the useful characters like dash and slash)
- To change keyboard layout later

```
sudo dpkg-reconfigure keyboard-configuration
```



Ubuntu server install - Network configuration

```
Network configuration

Configure at least one interface this server can use to talk
provides sufficient access for updates.

NAME      TYPE  NOTES
[ eth0    eth  -           ▶ ]
DHCPv4 ← 192.168.10.236/24
00:15:5d:0a:b3:09 / Unknown Vendor / Unknown Model

[ Create bond ▶ ]

[ Done ]
[ Back ]
```

Later, when you are connected to the lab network at the deep dive, you will change the IP address to a static IP. You will also want to do this when you have it in your own lab.

Leave the proxy address blank

```
Proxy configuration

If this system requires a proxy to connect to the internet, enter its details here.

Proxy address: 
```

Testing the network connection

```
Ubuntu archive mirror configuration

If you use an alternative mirror for Ubuntu, enter its details here.

Mirror address: http://no.archive.ubuntu.com/ubuntu/
You may provide an archive mirror to be used instead of the default.

This mirror location passed tests.

Get:1 http://no.archive.ubuntu.com/ubuntu noble InRelease [256 kB]
Get:2 http://no.archive.ubuntu.com/ubuntu noble-updates InRelease [126 kB]
Get:3 http://no.archive.ubuntu.com/ubuntu noble-backports InRelease [126 kB]
Fetched 508 kB in 1s (593 kB/s)
Reading package lists...

[ Done ]
[ Back ]
```



Ubuntu server install - Disk usage

- !!! When you encounter the disk usage screen, please take a little care and follow these steps !!!

- If not, you will only use about 50% of the allocated disk
- In the first screen, choose "Use an entire disk" (default)

Guided storage configuration

Configure a guided storage layout, or create a custom one:

(X) Use an entire disk

[VBOX_HARDDISK_VBe677ddd2-0fd66e9e local disk 40.000G ▼]

[X] Set up this disk as an LVM group

- At the next screen, select the "ubuntu-lv" option

- Press Enter, select Edit

- Change the "Size" to the max value and save

- (optional) IF you have forgotten this, here are the commands to extend the size later

Storage configuration

FILE SYSTEM SUMMARY

MOUNT POINT	SIZE	TYPE	DEVICE TYPE
[/	18.996G	new ext4	new LVM logical volume ▶]
[/boot	2.000G	new ext4	new partition of local disk ▶]

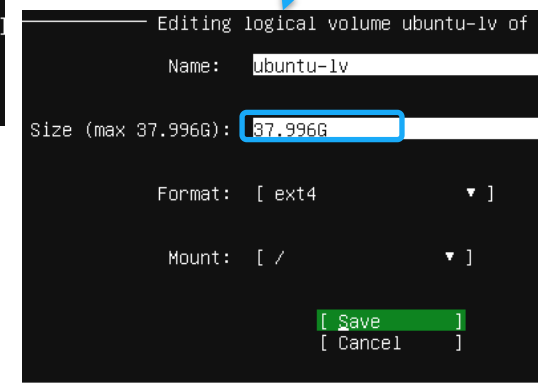
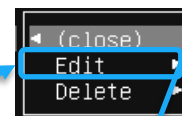
AVAILABLE DEVICES

DEVICE	TYPE	SIZE
[ubuntu-vg (new)	LVM volume group	37.996G ▶]
free space		19.000G ▶]

[Create software RAID (md) ▶]
[Create volume group (LVM) ▶]

USED DEVICES

DEVICE	TYPE	SIZE
[ubuntu-vg (new)	LVM volume group	37.996G ▶]
ubuntu-lv	new, to be formatted as ext4, mounted at /	18.996G ▶]
[VBOX_HARDDISK_VBe677ddd2-0fd66e9e	local disk	40.000G ▶]
partition 1	new, BIOS grub spacer	1.000M ▶]
partition 2	new, to be formatted as ext4, mounted at /boot	2.000G ▶]
partition 3	new, PV of LVM volume group ubuntu-vg	37.997G ▶]



```
devnet-adm@ubuntu-devnet:~$ sudo lvsdisplay | grep "LV Path"
LV Path          /dev/ubuntu-vg/ubuntu-lv
devnet-adm@ubuntu-devnet:~$ sudo lvextend -l +100%FREE /dev/ubuntu-vg/ubuntu-lv
devnet-adm@ubuntu-devnet:~$ df -h | grep "lv"
/dev/mapper/ubuntu--vg-ubuntu--lv 37G 14G 22G 39% /
devnet-adm@ubuntu-devnet:~$ sudo resize2fs /dev/mapper/ubuntu--vg-ubuntu--lv
```



Ubuntu server install

On the next pages, just keep the defaults and move on, until you reach this page

Profile configuration

Enter the username and password you will use to log in to the system. A password is still needed for sudo.

Your name:

Your servers name:
The name it uses when it talks to other computers

Pick a username:

Choose a password:

Confirm your password:

SSH Setup

You can choose to install the OpenSSH server package to enable ssh access to your machine.

☒ Install OpenSSH server

Import SSH identity:
You can import your SSH keys from GitHub or a file.

Import Username:

☒ Allow password authentication over SSH

Upgrade to Ubuntu Pro [Help]

Upgrade this machine to Ubuntu Pro for security updates on a much wider range of packages, until 2034. Assists with FedRAMP, FIPS, STIG, HIPAA and other compliance or hardening requirements.

[\[About Ubuntu Pro \]](#)

☐ Enable Ubuntu Pro

☒ Skip for now

You can always enable Ubuntu Pro later using the 'pro attach' command.

Featured Server Snaps [Help]

These are popular snaps in server environments. Select or deselect with SPACE, press ENTER to see more details of the package, publisher and versions available.

<input type="checkbox"/> microk8s	canonical✓	Kubernetes for workstations and appliances	▶
<input type="checkbox"/> nextcloud	nextcloud✓	Nextcloud Server - A safe home for all your data	▶
<input type="checkbox"/> wekan	xet7	The open-source kanban	▶
<input type="checkbox"/> kata-containers	katacontainers✓	Build lightweight VMs that seamlessly plug into the containers ecosystem	▶
<input type="checkbox"/> docker	canonical✓	Docker container runtime	▶
<input type="checkbox"/> canonical-livepatch	canonical✓	Canonical Livepatch Client	▶
<input type="checkbox"/> rocketchat-server	rocketchat✓	Rocket.Chat server	▶
<input type="checkbox"/> mosquito	mosquitto✓	Eclipse Mosquitto MQTT broker	▶
<input type="checkbox"/> etcd	canonical✓	Resilient key-value store by CoreOS	▶
<input type="checkbox"/> powershell	microsoft-powershell✓	PowerShell for every system!	▶
<input type="checkbox"/> sabnzbd	sabnzbd✓	SABnzbd	▶
<input type="checkbox"/> wormhole	snapcrafters✓	get things from one computer to another, safely	▶
<input type="checkbox"/> aws-cli	aws✓	Universal Command Line Interface for Amazon Web Services	▶
<input type="checkbox"/> google-cloud-sdk	google-cloud-sdk✓	Google Cloud SDK	▶
<input type="checkbox"/> slcli	softlayer✓	Python based SoftLayer API Tool.	▶
<input type="checkbox"/> doctl	digitalocean✓	The official DigitalOcean command line interface	▶
<input type="checkbox"/> conjure-up	canonical✓	Package runtime for conjure-up spells	▶
<input type="checkbox"/> postgresql10	cmd✓	PostgreSQL is a powerful, open source object-relational database system.	▶
<input type="checkbox"/> heroku	heroku✓	CLI client for Heroku	▶
<input type="checkbox"/> keepalived	keepalived-project✓	High availability VRRP/BFD and load-balancing for Linux	▶
<input type="checkbox"/> prometheus	canonical✓	The Prometheus monitoring system and time series database	▶
<input type="checkbox"/> juju	canonical✓	Juju - a model-driven operator lifecycle manager for K8s and machines	▶

After «Installation complete» and «Reboot Now» you will get this screen. You should continue by just pressing Enter

```
[FAILED] Failed unmounting /cdrom.
Please remove the installation medium, then press ENTER:
[FAILED] Failed unmounting /cdrom.
```



Ubuntu server install

- At first boot it will look like this

```
ubuntu-devnet login: devnet-adm
Password:
Welcome to Ubuntu 24.04 LTS (GNU/Linux 6.8.0-35-generic x86_64)

 * Documentation:  https://help.ubuntu.com
 * Management:    https://landscape.canonical.com
 * Support:       https://ubuntu.com/pro

System information disabled due to load higher than 1.0
Expanded Security Maintenance for Applications is not enabled.

35 updates can be applied immediately.
To see these additional updates run: apt list --upgradable

Enable ESM Apps to receive additional future security updates.
See https://ubuntu.com/esm or run: sudo pro status


The programs included with the Ubuntu system are free software;
the exact distribution terms for each program are described in the
individual files in /usr/share/doc/*/copyright.

Ubuntu comes with ABSOLUTELY NO WARRANTY, to the extent permitted by
applicable law.

To run a command as administrator (user "root"), use "sudo <command>".
See "man sudo_root" for details.
```

- Run these commands to upgrade all packages to the latest version

```
devnet-adm@ubuntu-devnet:~$ sudo apt update
devnet-adm@ubuntu-devnet:~$ sudo apt upgrade
```



Change from DHCP to static IP

- When you come to WLPC you would need to use a static IP in the lab range. Before WLPC you might just keep the server in DHCP mode, and come back to this slide during the deep dive.
- This will show one method to permanently change the IP of your server
 - Check your adapter name

```
devnet-adm@ubuntu-devnet:~$ netplan status | grep network
2: ens18 ethernet UP (networkd: ens18)
```

- Create a new file `/etc/netplan/99_config.yaml`

```
devnet-adm@ubuntu-devnet:~$ sudo nano /etc/netplan/99_config.yaml
devnet-adm@ubuntu-devnet:~$ sudo chmod 600 /etc/netplan/99_config.yaml
```

- Delete the auto-created `/etc/netplan/50-cloud-init.yaml`

```
devnet-adm@ubuntu-devnet:~$ sudo rm /etc/netplan/50-cloud-init.yaml
```

- Run "sudo netplan apply" to use the new config file

```
devnet-adm@ubuntu-devnet:~$ sudo netplan apply
```

- Check that you have the new IP

```
devnet-adm@ubuntu-devnet:~$ sudo netplan status
```

- Reboot the server and you're good to go ☺

```
devnet-adm@ubuntu-devnet:~$ sudo reboot now
```

!!! In YAML, indentation is REALLY important !!!
In this file I use 2, 4, 6 or 8 spaces (multiple of 2).
It would be valid to use multiple of 3 or 4 instead.

Get this value from
"netplan status". Enter
it into 99_config.yaml
Probable values:
- Proxmox: ens18
- Hyper-V: eth0
- VirtualBox: enp0s3

Change to YOUR
Ubuntu IP

```
/etc/netplan/99_config.yaml
network:
  version: 2
  ethernets:
    ens18:
      #      dhcp4: true
      dhcp6: false
      dhcp4: false
      addresses:
        - 192.168.10.{YOUR_POD_IP}/24
      routes:
        - to: default
          via: 192.168.10.1
      nameservers:
        addresses:
          - 1.1.1.1
```

Most Common errors:

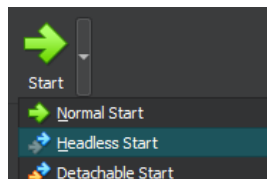
- Wrong file permissions (step 2)
- Errors in the indentation in the YAML file

To easily change between static and DHCP, I just comment/uncomment the lines. To change this to DHCP, just unncomment the "dhcp4: true" line, and comment the rest

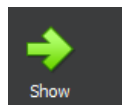


Connecting to the server using SSH

- Now would be a great time to connect to the Ubuntu server using your favourite SSH client. Like MobaXterm, SecureCRT, PuTTY, etc. You can also use the ssh CLI tool in your preferred shell
- The main reason for this is copy-paste, which you will use during these pre-lab tasks, and much more in the deep dive itself ☺ It might be possible to get good copy-paste behavior in the VirtualBox console, but the eventual instructions for that is still "to-be"☺
- (VirtualBox) If you have static IP and SSH up and running, the next time you reboot your Ubuntu Server, you could use "Headless Start" to prevent the console window pop-up.



- If you need the console when started headless, just use the "Show" button found where the Start button usually is before the VM is started



Create an SSH key -> Import to Ubuntu keystore

The reason for doing this, is to be able to log in from your laptop to your server with SSH from the terminal or from VS Code, without typing the password every time

- Start by opening Powershell (not CMD for this task) - on Mac you just use Terminal
- Then, logging in to your Ubuntu Server using SSH

```
PS C:\> ssh -l devnet-adm 192.168.10.7  
devnet-adm@ubuntu-13:~$ exit
```

- Create SSH key for your user

```
PS C:\> ssh-keygen -t ed25519
```

- Copy the public part of your key to Ubuntu

```
PS C:\> type ~\.ssh\id_ed25519.pub | ssh -l devnet-adm 192.168.10.7 "cat >> .ssh/authorized_keys"
```

Change to your server's IP

```
The authenticity of host '192.168.10.7 (192.168.10.7)' can't be established.  
ED25519 key fingerprint is SHA256:V+TRZ8hkGRQh+kcu10gUp+roD4ds00ulip5HAcoY96s.  
This key is not known by any other names  
Are you sure you want to continue connecting (yes/no/[fingerprint])? yes  
Warning: Permanently added '192.168.10.7' (ED25519) to the list of known hosts.  
devnet-adm@192.168.10.7's password:  
PS C:\> |
```

- Notice, when you SSH to your server now, you don't have to use a password

```
PS C:\> ssh -l devnet-adm 192.168.10.7  
Welcome to Ubuntu 24.04 LTS (GNU/Linux 6.8.0-38-generic x86_64)  
  
* Documentation:  https://help.ubuntu.com  
* Management:    https://landscape.canonical.com  
* Support:        https://ubuntu.com/pro  
  
System information as of Sat Jul 20 05:26:40 PM UTC 2024
```

- Reference: <https://code.visualstudio.com/docs/remote/ssh-tutorial>



Git clone the example files

```
devnet-adm@ubuntu-devnet:~$ cd ~
devnet-adm@ubuntu-devnet:~$ git clone https://github.com/akoksrud/wifi-automation
Cloning into 'wifi-automation'...
remote: Enumerating objects: 20, done.
remote: Counting objects: 100% (20/20), done.
remote: Compressing objects: 100% (15/15), done.
remote: Total 20 (delta 2), reused 15 (delta 1), pack-reused 0
Receiving objects: 100% (20/20), 15.61 KiB | 841.00 KiB/s, done.
Resolving deltas: 100% (2/2), done.
devnet-adm@ubuntu-devnet:~$ cd wifi-automation
devnet-adm@ubuntu-devnet:~/wifi-automation$ ls -l
total 52
drwxrwxr-x 2 devnet-adm devnet-adm 4096 Jul  1 18:38 examples
-rw-rw-r-- 1 devnet-adm devnet-adm 35149 Jul  1 18:38 LICENSE
drwxrwxr-x 2 devnet-adm devnet-adm 4096 Jul  1 18:38 presentation
-rw-rw-r-- 1 devnet-adm devnet-adm   63 Jul  1 18:38 README.md
drwxrwxr-x 2 devnet-adm devnet-adm 4096 Jul  1 18:38 solutions
devnet-adm@ubuntu-devnet:~/wifi-automation$
```

- Git will not be used for version control or backup in this lab, but it is highly recommended to use when building your own projects
- To update the contents of the cloned repository, do a git pull like this

```
devnet-adm@ubuntu-devnet:~/wifi-automation$ git pull origin main
```



Customizing your shell (optional)

- There are numerous ways to customize the shell as you like. One of them is using oh-my-bash, found on <https://github.com/ohmybash/oh-my-bash>

```
devnet-adm@ubuntu-devnet:~$ bash -c "$(curl -fsSL https://raw.githubusercontent.com/ohmybash/oh-my-bash/master/tools/install.sh)"
(...)
16:50:57 devnet-adm@ubuntu-devnet ~ → cd wifi-automation
16:51:38 devnet-adm@ubuntu-devnet ~ |main|→ nano ~/.bashrc
```

- Some benefits/differences by customizing the shell is getting info in the prompt about git branch, timestamps, etc. You can select a different theme in ~/.bashrc

"agnoster" theme:

```
devnet-adm@ubuntu-devnet ~/wifi-automation main
```

"powerbash10k" theme:

```
~ /wifi-automation git main ✓ ..... devnet-adm@ubuntu-devnet 16:12:46
```

- Another alternative is "Starship" found on <https://starship.rs>

```
devnet-adm@ubuntu-devnet:~$ curl -sS https://starship.rs/install.sh | sh
→ ~ wifi-automation git:(main) nano ~/.bashrc
→ ~ wifi-automation git:(main) source ~/.bashrc
devnet-adm in 🌐 ubuntu-devnet in wifi-automation on 🍷 main [!?] via 🦋 v3.12.3 (automation-venv)
✦ >
```

```
# Added:
eval "$(starship init bash)"
```

- You can also just modify the "PS1" variable in your ~/.bashrc file
This will customize the info and colors, but not give git/python/etc status. See <https://bash-prompt-generator.org/>

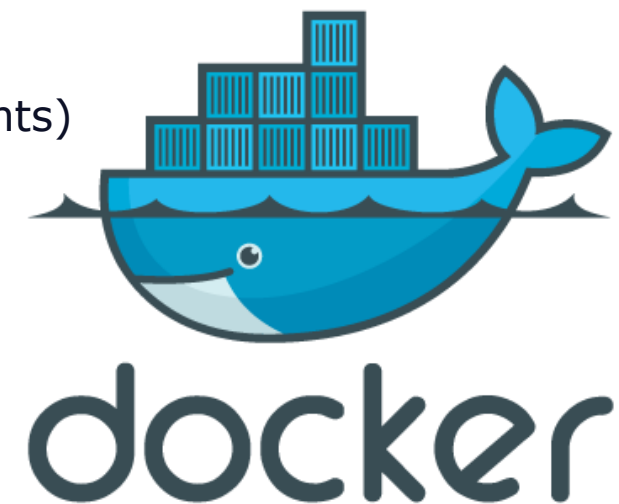
```
devnet-adm@ubuntu-devnet:~$ nano ~/.bashrc
devnet-adm@ubuntu-devnet:~$ source ~/.bashrc
[16:50:57] devnet-adm@ubuntu-devnet in ~ $
```

```
# Replaced the current PS1= section with
PS1='\[\e[93m\]\t\[\e[0m\] \[\e[32m\]\u@\h\[\e[37m\] in \[\e[94m\]\w\[\e[0m\] \[\e[37m\]\$\[\e[0m\] '
```



Pre-lab task #3: Install Docker on the Ubuntu Server

- Installing Docker on Ubuntu Server
- Docker is a platform for running containers (isolated/controlled environments)
- As Docker containers we will run
 - Telegraf, InfluxDB and Grafana to visualize Telemetry data from IOS-XE
 - You can also run YANG Suite to browse YANG models of your WLC
- We will use "docker-compose" for our container setups
 - All settings for a container specified in a YAML file
- To test the Docker installation, we will run
 - hello-world, a very small container that outputs "Hello from Docker!" if Docker is correctly installed
 - alpine, a very small footprint linux container



Installing docker

- Copy-paste should be OK from this slide, but it doesn't always like if you copy-paste the full content at once, so just do it in some separate chunks

```
# Add Docker's official GPG key:
sudo apt update
sudo apt install ca-certificates curl gnupg
sudo install -m 0755 -d /etc/apt/keyrings
curl -fsSL https://download.docker.com/linux/ubuntu/gpg | sudo gpg --dearmor -o /etc/apt/keyrings/docker.gpg
sudo chmod a+r /etc/apt/keyrings/docker.gpg

# Add the repository to Apt sources:
echo \
  "deb [arch="$(dpkg --print-architecture)" signed-by=/etc/apt/keyrings/docker.gpg] https://download.docker.com/linux/ubuntu \
  "$(. /etc/os-release && echo "$VERSION_CODENAME")" stable" | \
  sudo tee /etc/apt/sources.list.d/docker.list > /dev/null
sudo apt update

# Install Docker:
sudo apt-get install docker-ce docker-ce-cli containerd.io docker-buildx-plugin docker-compose-plugin

# Add current user to docker group, to be able to run containers without sudo
sudo groupadd docker
sudo usermod -aG docker $USER
sudo reboot
```



Docker first run

```
devnet-adm@ubuntu-devnet:~$ docker run hello-world
```

```
Hello from Docker!  
This message shows that your installation appears to be working correctly.  
(... cut for brevity ...)  
For more examples and ideas, visit:  
https://docs.docker.com/get-started/
```

```
devnet-adm@ubuntu-devnet:~$
```

```
devnet-adm@ubuntu-devnet:~$ docker pull alpine
```

```
Using default tag: latest  
latest: Pulling from library/alpine  
ec99f8b99825: Pull complete  
Digest: sha256:b89d9c93e9ed3597455c90a0b88a8bbb5cb7188438f70953fede212a0c4394e0  
Status: Downloaded newer image for alpine:latest  
docker.io/library/alpine:latest  
devnet-adm@ubuntu-devnet:~$ docker run -it alpine
```

```
/ # cat /etc/os-release  
NAME="Alpine Linux"  
ID=alpine  
VERSION_ID=3.20.3  
PRETTY_NAME="Alpine Linux v3.20"  
HOME_URL="https://alpinelinux.org/"  
BUG_REPORT_URL="https://gitlab.alpinelinux.org/alpine/aports/-/issues"  
/ # apk add nano  
(... cut for brevity ...)  
/ # nano /tmp/test.txt  
/ # exit
```

- The hello-world container is a small container, that just helps you check if your Docker installation is working
- Alpine is a very slim Linux distro. As an empty container, it requires about 8MB disk space. A lot of pre-built containers (like we will use later) are based on the alpine image.
- We will first pull Alpine, then run it and connect to its terminal using the "-it" option
- When connected to the Alpine container, we can do regular Linux stuff, and also install apps using apk. As an example we will install nano, edit a text file, and exit the container
- Containers are deleted after exiting, unless permanent files are specified (we will use that later)



Pre-lab task #4: Install Postman on your laptop

- Download and follow the instructions on
 - <https://www.postman.com/downloads/>



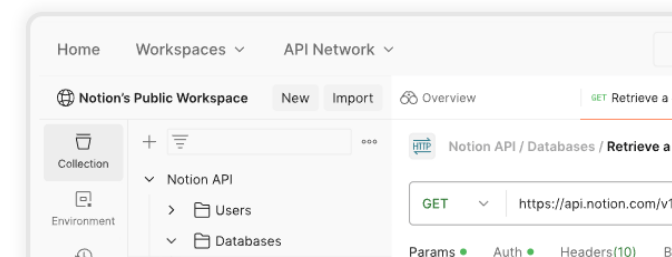
Download Postman

Download the app to get started using the Postman API Platform today. Or, if you prefer a browser experience, you can try the web version of Postman.

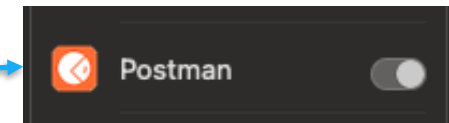
The Postman app

Download the app to get started with the Postman API Platform.

 Windows 64-bit



- If running on Mac, be sure that Postman have access to the Local Network System Settings -> Privacy & Security -> Local Network



Special thanks to Nick Cuypers for this little trick 😊



Create a Postman account

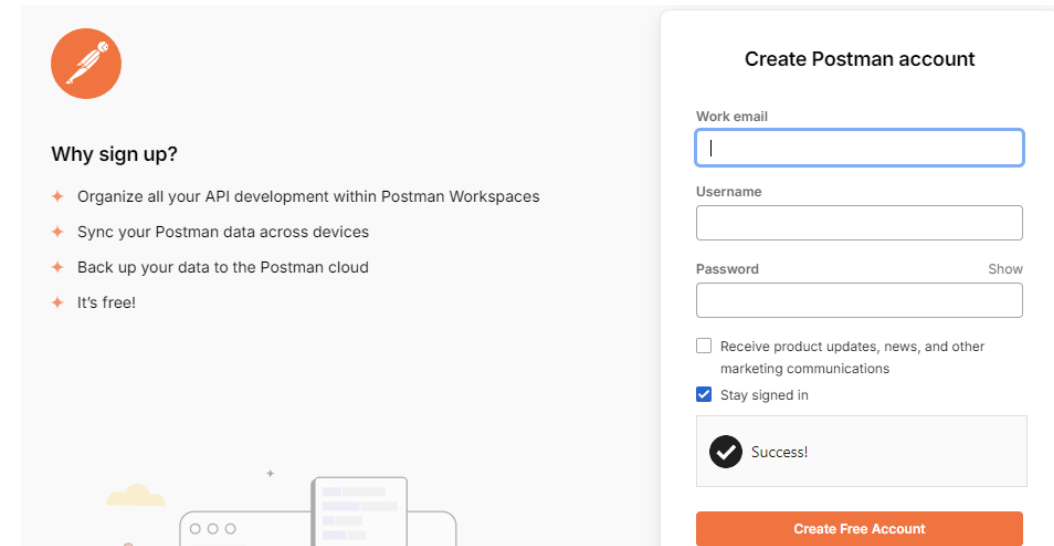
- Create a Postman account

[Sign Up for Free](#)

- After creating the account, I would always recommend to enable 2FA

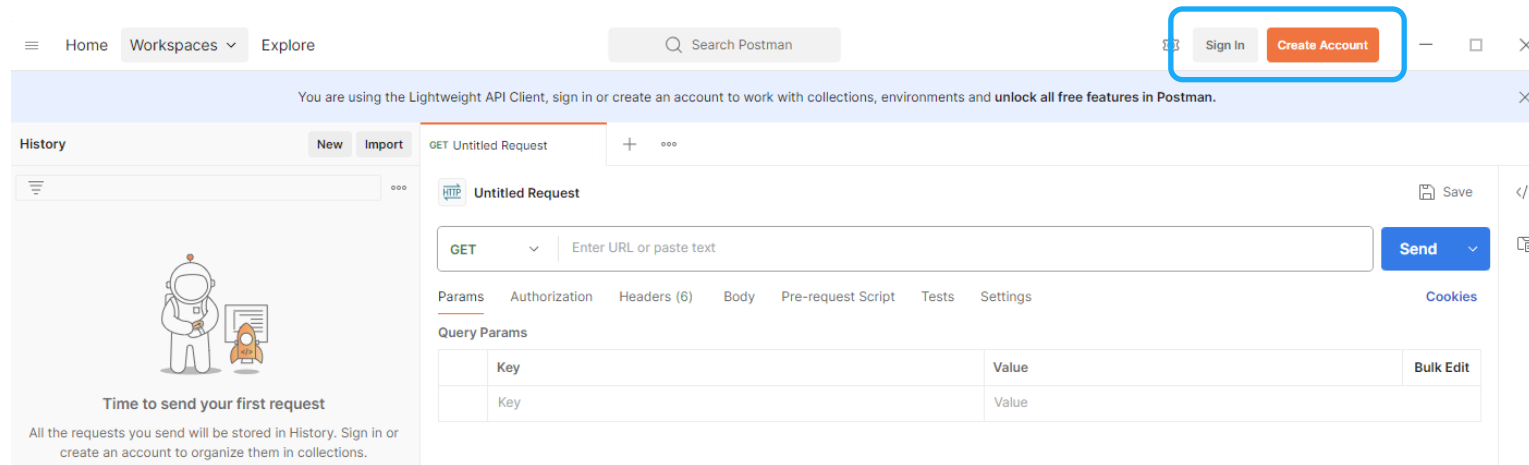
Two-factor authentication **ACTIVE**

Add an extra layer of security to your account by enabling two-factor authentication.
[Regenerate recovery codes](#)



The image shows the Postman account creation process. On the left, a 'Why sign up?' section lists benefits: organizing API development, syncing data across devices, backing up data to the cloud, and noting it's free. On the right, the 'Create Postman account' form includes fields for 'Work email', 'Username', and 'Password' (with a 'Show' toggle). It also has checkboxes for 'Receive product updates, news, and other marketing communications' (unchecked) and 'Stay signed in' (checked). A 'Success!' message with a checkmark is displayed below the form, and a 'Create Free Account' button is at the bottom.

- Log in to the Postman app using your account



The image shows the Postman app interface. At the top, there's a navigation bar with 'Home', 'Workspaces', and 'Explore' tabs, and a search bar. A 'Sign In' button and a 'Create Account' button are highlighted with a red box. Below the navigation bar, a message states: 'You are using the Lightweight API Client, sign in or create an account to work with collections, environments and unlock all free features in Postman.' The main area is divided into 'History' on the left and a 'GET Untitled Request' editor on the right. The 'History' section shows a 'Time to send your first request' message. The 'GET Untitled Request' editor has a 'Send' button and a 'Bulk Edit' button. Below the editor, there's a table for 'Query Params' with columns 'Key' and 'Value'.

Key	Value
Key	Value



Pre-lab task #5: Install VS Code on your laptop

- Download from <https://code.visualstudio.com/>
- Some recommended extensions
 - Ansible
 - Python
 - Remote - SSH
 - Codeium
- A word of caution - beware of fake plugins/extensions
 - <https://www.bleepingcomputer.com/news/security/malicious-vscode-extensions-with-millions-of-installs-discovered/>



VS Code

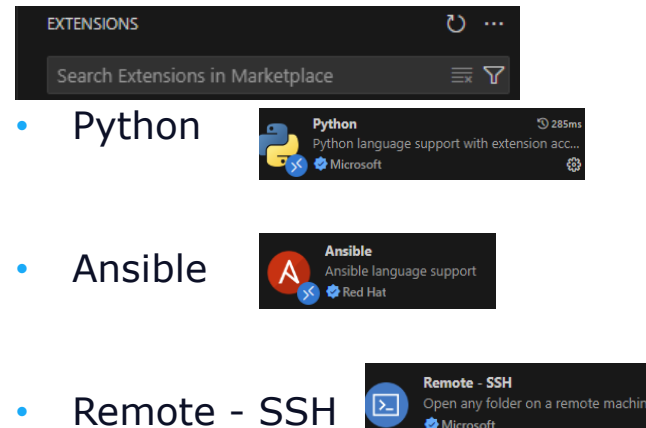
Explorer (files etc)

Command palette

Source control (Git):

- If you press here you are asked to log in etc. You don't have to do this to complete the lab, it will not be used.

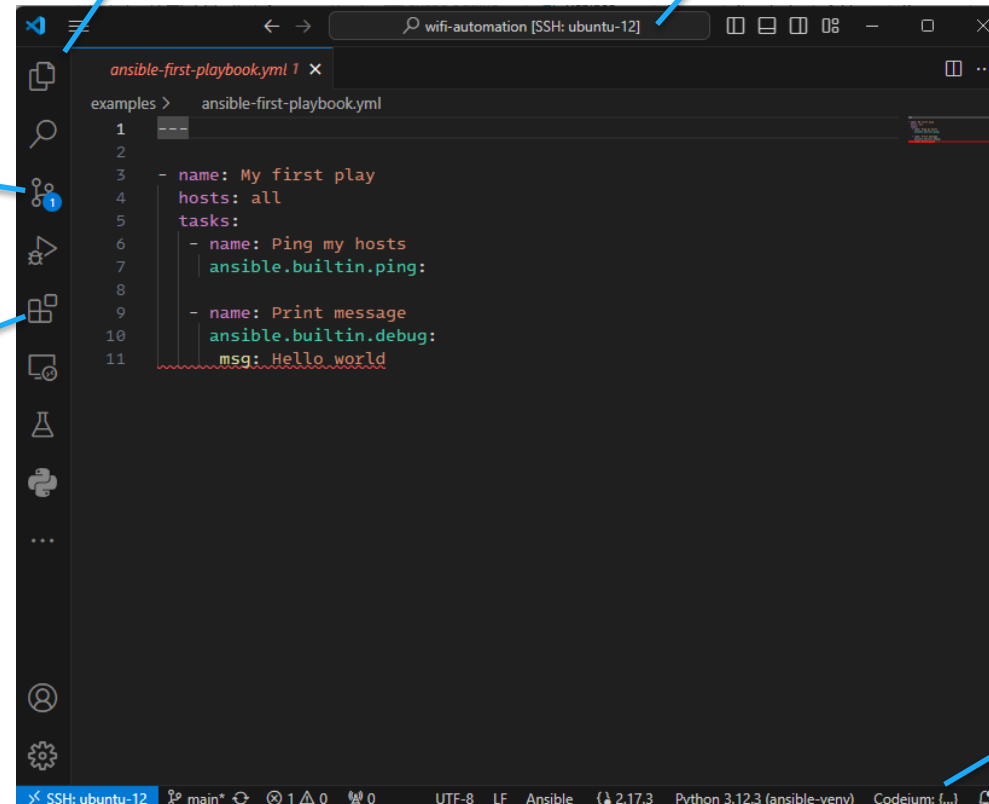
Extensions:



EXTENSIONS

Search Extensions in Marketplace

- Python
Python language support with extension acc...
Microsoft
- Ansible
Ansible language support
Red Hat
- Remote - SSH
Open any folder on a remote machine
Microsoft

Some active extension
(Codeium, GitHub Copilot, etc)Language (Python/Ansible/etc)
Python environment

Pre-lab task #6: 9800-CL (optional)

- In this task we will install Cisco 9800-CL. Instructions are provided for Hyper-V (6a), VirtualBox (6b) and Proxmox (6d)
- It is not possible to run 9800-CL on Apple Silicon (M-series Apple CPUs)
- No instructions will be supplied from our side for getting a 9800 to run on older Macs with Intel silicon, but you should be able to get it to work using VirtualBox (which is supported on Intel silicon Macs). Or you can follow this blog post from Francois:
 - <https://semfionetworks.com/blog/setup-cisco-catalyst-9800-controller-on-your-laptop/>
- In the deep dive sessions, we encourage you to not spend time troubleshooting your WLC, why an AP doesn't join, why function X or Y doesn't work, etc. Based on experience from previous deep dives, this will steal most of the time that you should use on trying out various automation exercises. Unless your WLC works right out of the box, use the assigned shared WLCs, which already have joined APs.
- **!!! Note !!!**
 - The virtual WLC is NOT a big fan of the host computer entering sleep mode. So you should do a "power off" on the 9800 VM before sleeping your laptop



Pre-lab task #6: 9800-CL (optional)

- Start by downloading the software

- <https://software.cisco.com>

- "Access downloads"

- Log in

- 9800-CL version 17.15.2 is used in this lab. Download the ISO image

- Any recent version should work, but if we use the same version the APs don't have to download software if changing WLCs

- If you do not have access to Cisco Software, you can use the shared 9800 wlc9 (see topology map)

- Example uses Ultra-low specs

- RAM: 6144MB

- Processors: 2

- HDD: 16GB

Access downloads >

Login to view your download history

LOG IN NOW

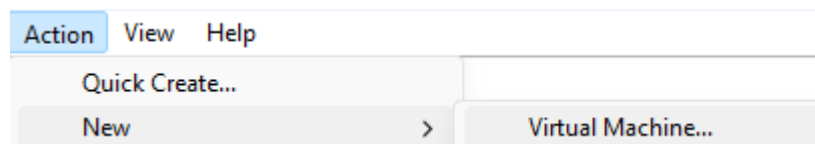
Cisco Catalyst 9800 Wireless Controller for Cloud - Hyper-V / ESXi 29-Nov-2024 1629.01 MB
/ KVM
C9800-CL-universalk9.17.15.02.iso

Table 1. Supported Profile Configurations

Profiles	CPUs	RAM	APs	Clients
Ultra-Low	2 vCPUs	6 GB	100	1,000
Small	4 vCPUs	8 GB	1000	10,000
Medium	6 vCPUs	16 GB	3000	32,0000
Large	10 vCPUs	32 GB	6000	64,0000

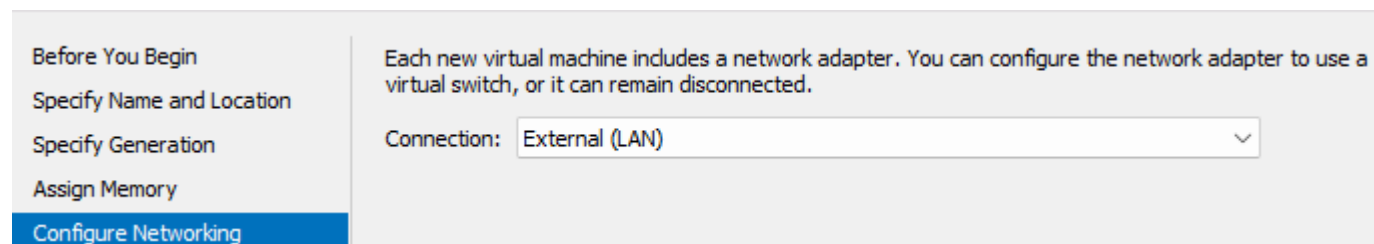
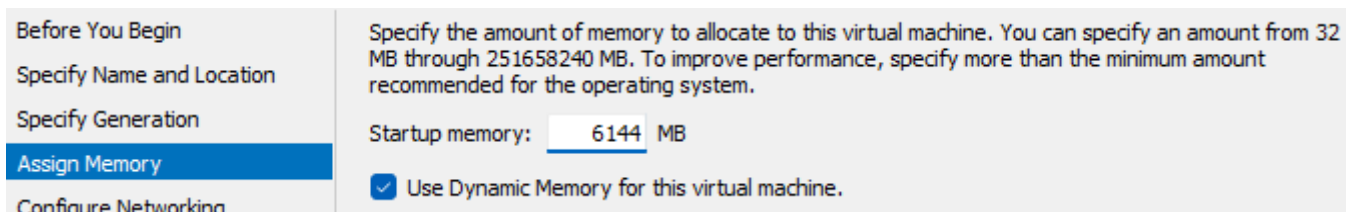
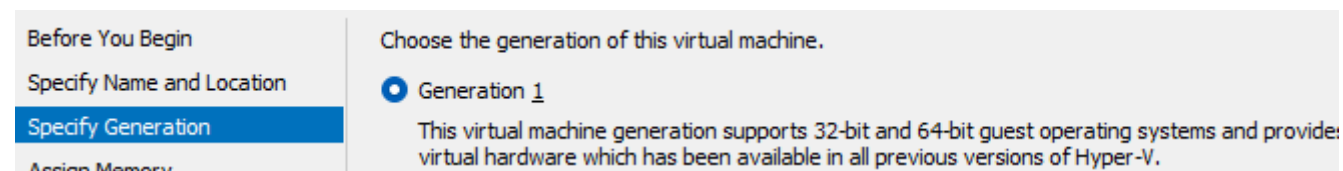
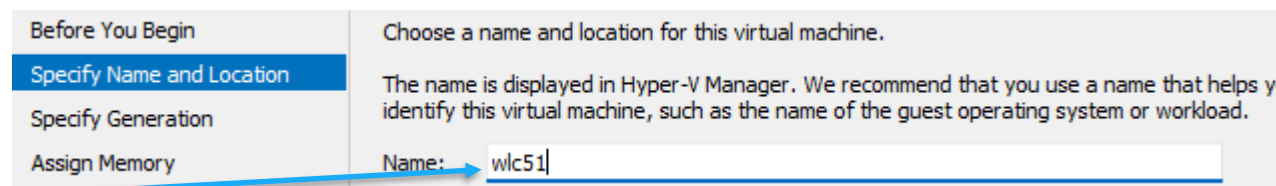


6a.Create VM (Hyper-V)



Change the name to reflect your pod number. E.g.

wlc11
wlc12
wlc13
etc



6a.Create VM (Hyper-V)

Specify Name and Generation

Specify Generation

Assign Memory

Configure Networking

Connect Virtual Hard Disk

Summary

☐ Create a virtual hard disk
Use this option to create a VHDX dynamically expanding virtual hard disk.

Name:

Location:

Size: GB (Maximum: 64 TB)

☐ Use an existing virtual hard disk
Use this option to attach an existing virtual hard disk, either VHD or VHDX format.

Location:

☒ Attach a virtual hard disk later
Use this option to skip this step now and attach an existing virtual hard disk later.

Open Powershell and issue these commands to rename the network adapters to the same as inside the 9800:

```
PS C:\> Rename-VMNetworkAdapter -VMName wlc51 -Name "Network Adapter" -NewName GigabitEthernet1
PS C:\> Set-VMNetworkAdapterVlan -VMName wlc51 -VMNetworkAdapterName GigabitEthernet1 -Untagged
```

You can get the current settings by using these commands

```
PS C:\> Get-VMNetworkAdapterVlan -VMName wlc51
PS C:\> Get-VMNetworkAdapterVlan -ManagementOS
```

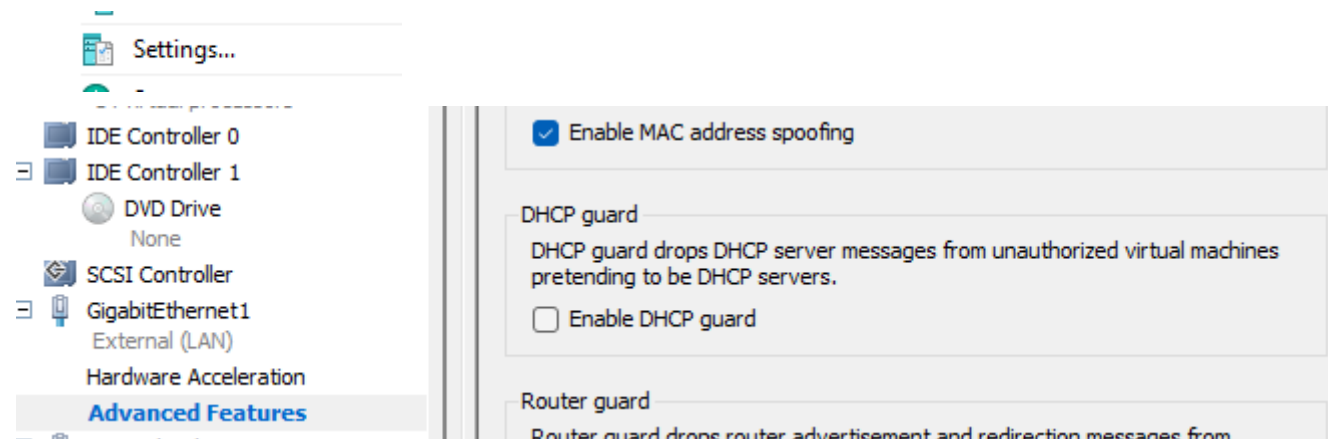
With this configuration (Untagged) we can only use the Native VLAN on GigabitEthernet1, which is VLAN 10. Hyper-V is not so easy (or at least not as well-documented) as VirtualBox or VMWare to create tagged switches. We have a solution, but currently I have not been able to figure out how to access the native VLAN with the host OS at the same time. It should be possible, I am happy for any suggestions. A workaround is to use Gig1 for untagged access to VLAN 10, and Gig2 for VLAN 11. **The following commands will make the Trunk work, but currently breaks the host OS access to the network, so be warned ;-)**

However - Using FlexConnect you will not need to have VLAN 11 accessible from the WLC, as the APs themselves will drop the traffic to VLAN 11

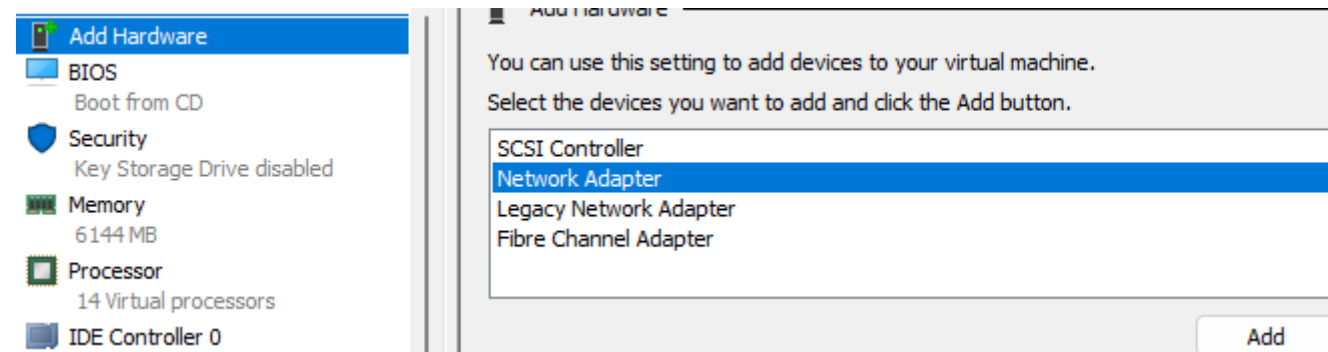
```
PS C:\> Set-VMNetworkAdapterVlan -VMName wlc51 -VMNetworkAdapterName GigabitEthernet1 -Trunk -AllowedVlanIdList "10-11" -NativeVlanId 10
PS C:\> Set-VMNetworkAdapterVlan -ManagementOS -VMNetworkAdapterName "External (LAN)" -Trunk -AllowedVlanIdList "10-11" -NativeVlanId 10
(to revert, use the following command, as well as the Untagged command for Gig1 from previous slide)
PS C:\> Set-VMNetworkAdapterVlan -ManagementOS -VMNetworkAdapterName "External (LAN)" -Untagged
```



6a.Create VM (Hyper-V)



Add the second adapter, and rename it using PowerShell

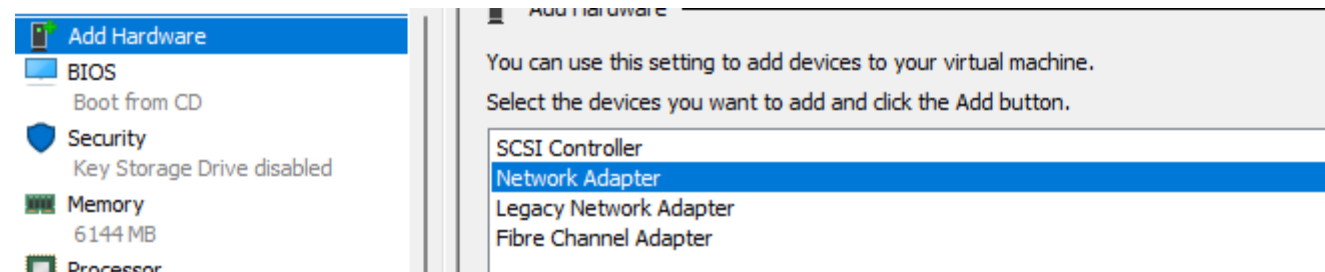


```
PS C:\> Rename-VMNetworkAdapter -VMName wlc51 -Name "Network Adapter" -NewName GigabitEthernet2
PS C:\> Get-VMNetworkAdapterVlan -VMName wlc51
```

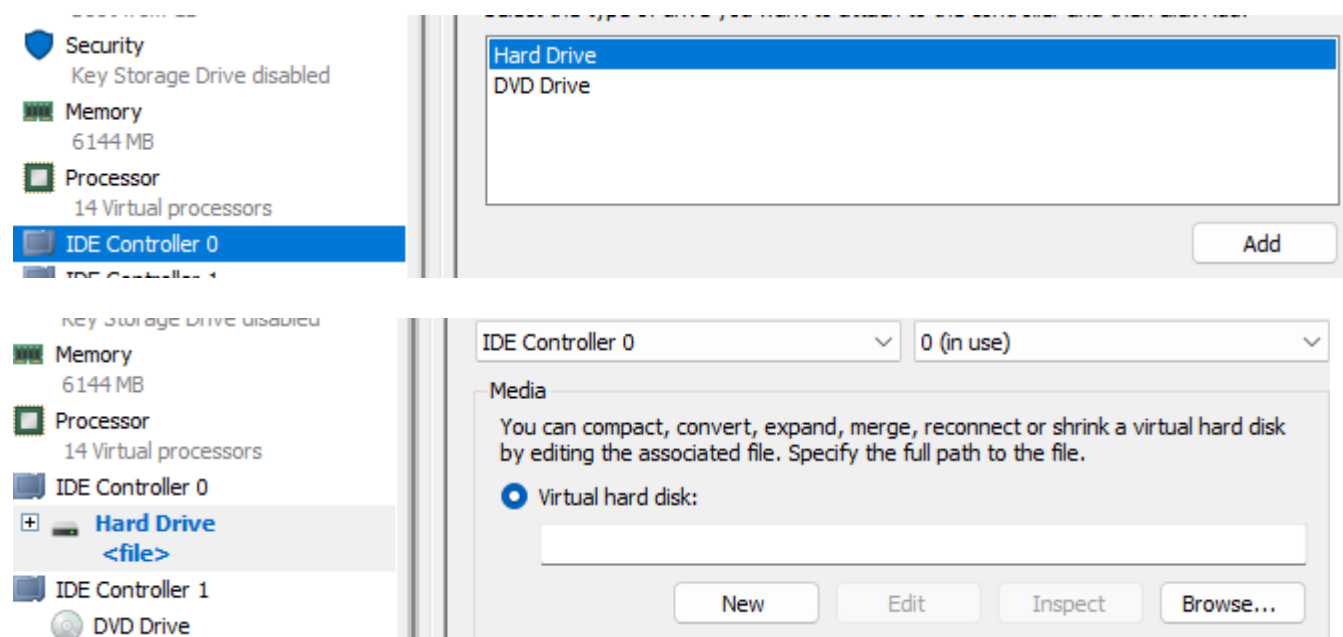


6a.Create VM (Hyper-V)

Add the third adapter, and rename it using PowerShell

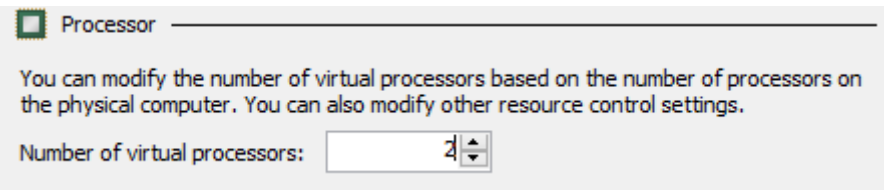
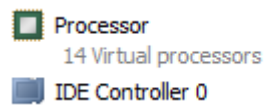


```
PS C:\> Rename-VMNetworkAdapter -VMName wlc51 -Name "Network Adapter" -NewName GigabitEthernet3
PS C:\> Get-VMNetworkAdapterVlan -VMName wlc51
```



6a.Create VM (Hyper-V)

Change the number of CPUs to 2



6a.Create VM (Hyper-V)

The screenshot displays the Hyper-V VM creation wizard with the 'Specify Name and Location' step selected. The left sidebar shows the progression: 'Before You Begin', 'Choose Disk Format', 'Choose Disk Type', 'Specify Name and Location', and 'Configure Disk'. The main area contains the following information:

- Before You Begin**
- Choose Disk Format**
- Choose Disk Type**
- Specify Name and Location**
- Configure Disk**

What type of virtual hard disk do you want to create?

☒ Fixed size

This type of disk provides better performance and is recommended for servers running a workload with high levels of disk activity. The virtual hard disk file that is created initially uses the specified size and does not change when data is deleted or added.

Specify the name and location of the virtual hard disk file.

Name:

Location:

You can create a blank virtual hard disk or copy the contents of an existing physical disk.

☒ Create a new blank virtual hard disk

Size: GB (Maximum: 2.040 GB)

☐ Copy the contents of the specified physical disk:

14 virtual processors

- IDE Controller 0
 - Hard Drive **wlc51.vhd**
- IDE Controller 1
 - DVD Drive **C9800-CL-universalk9.1...**

☒ Image file:

☐ Physical CD/DVD drive:



6b. Create VM (VirtualBox)

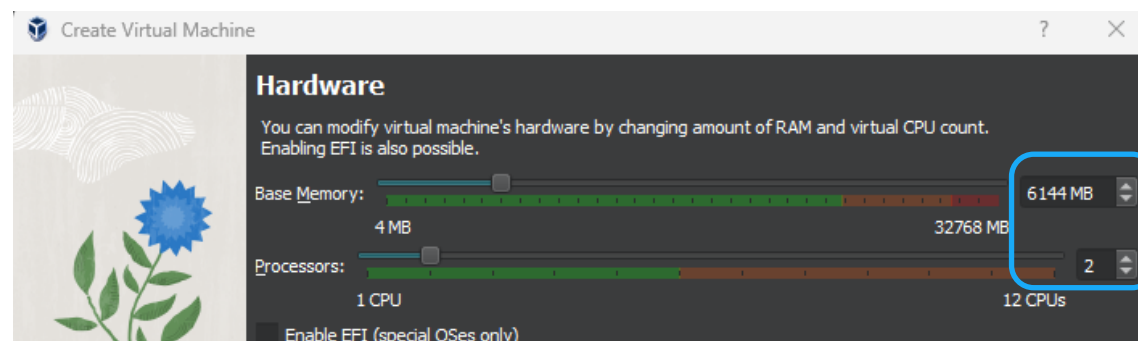
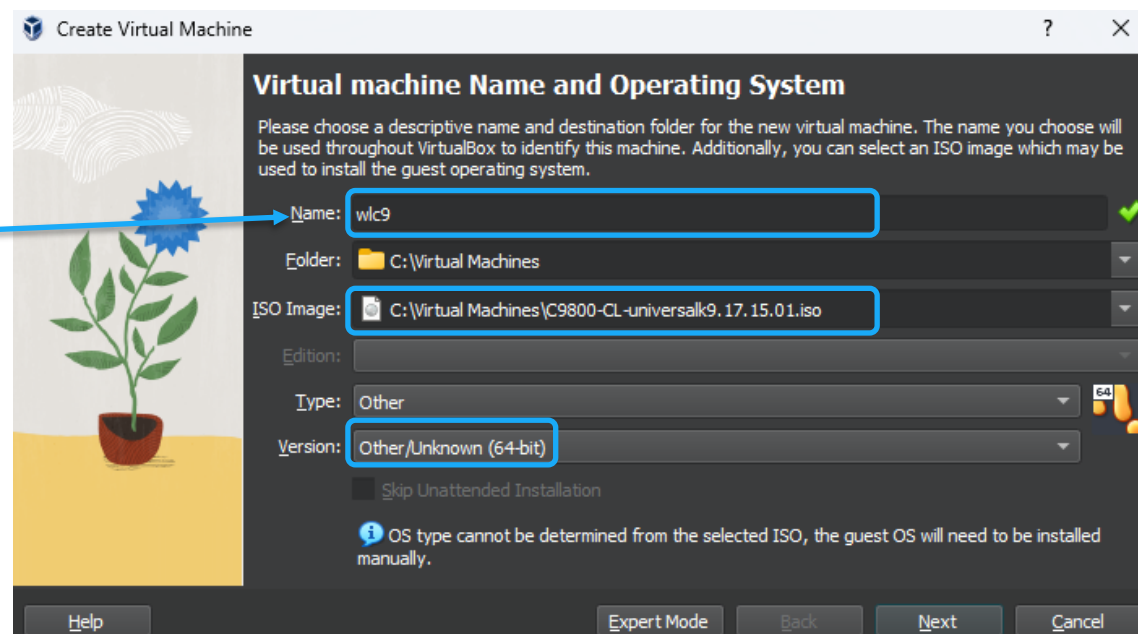
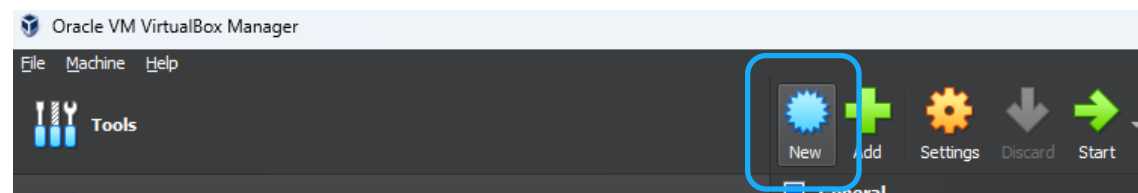
Change the name to reflect your pod number. E.g.

wlc11

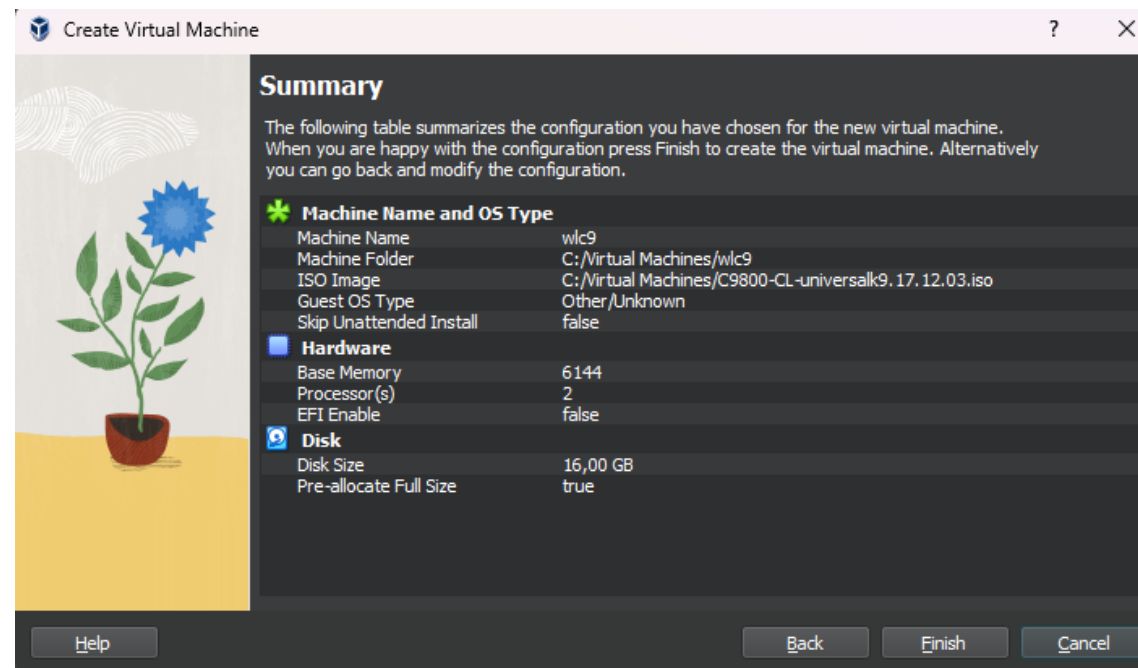
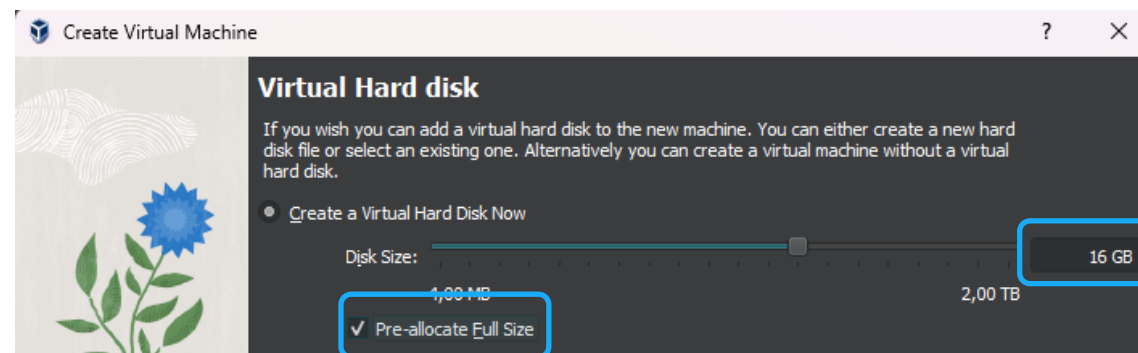
wlc12

wlc13

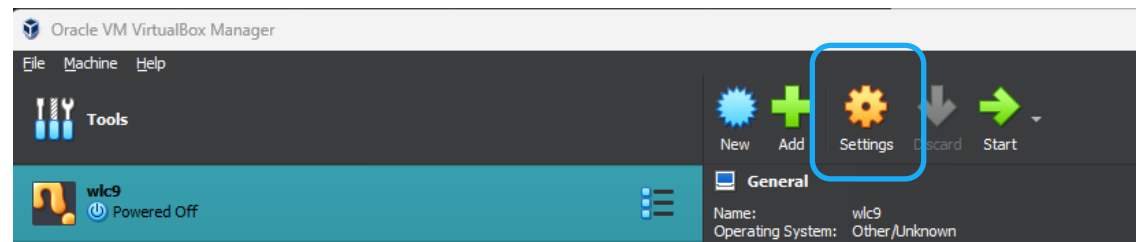
etc



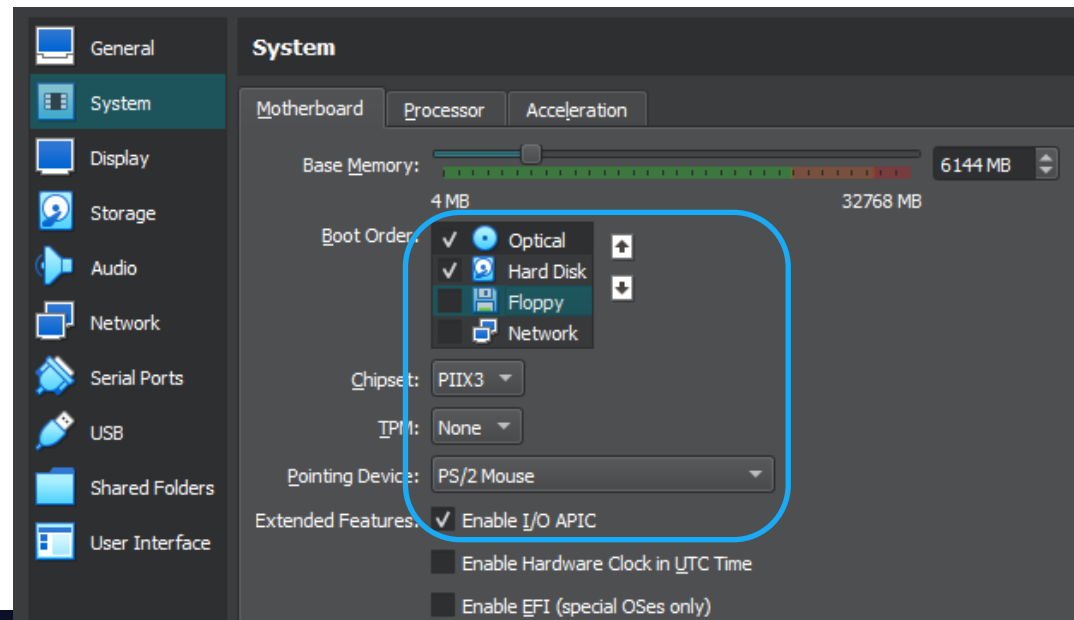
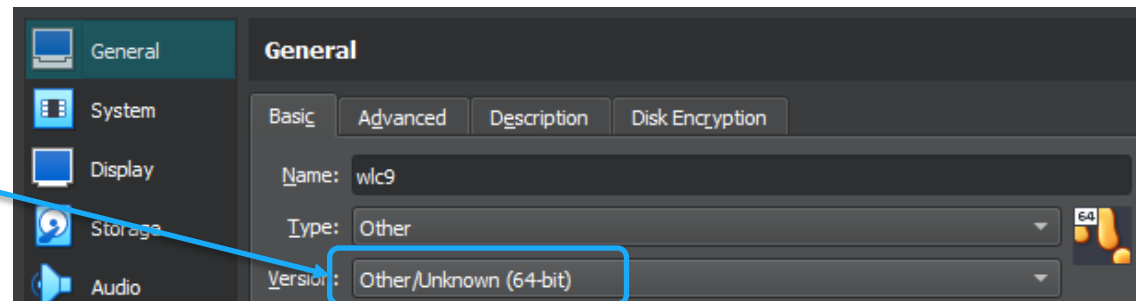
6b. Create VM (VirtualBox)



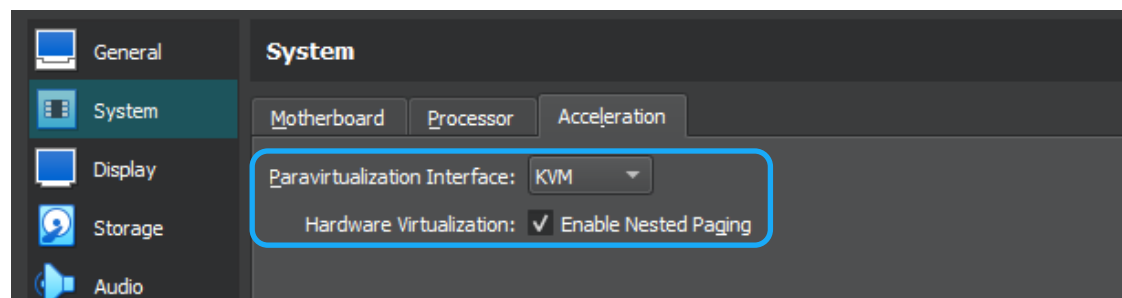
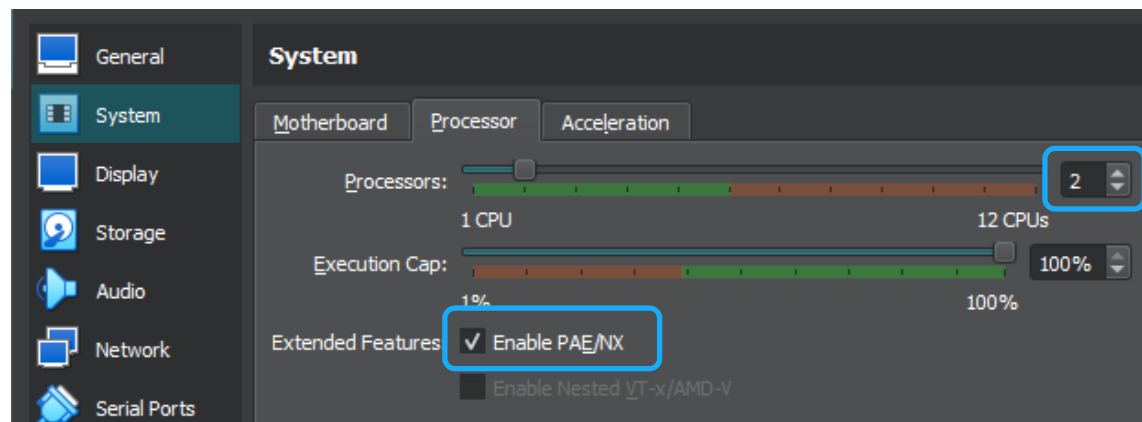
6b. Modify VM (VirtualBox)



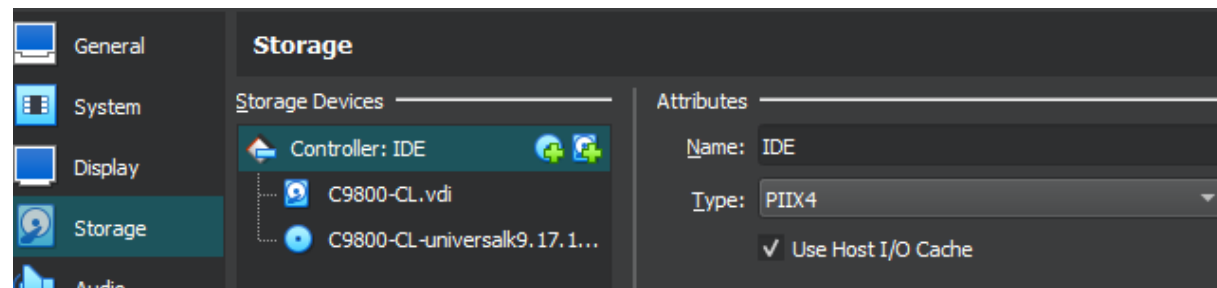
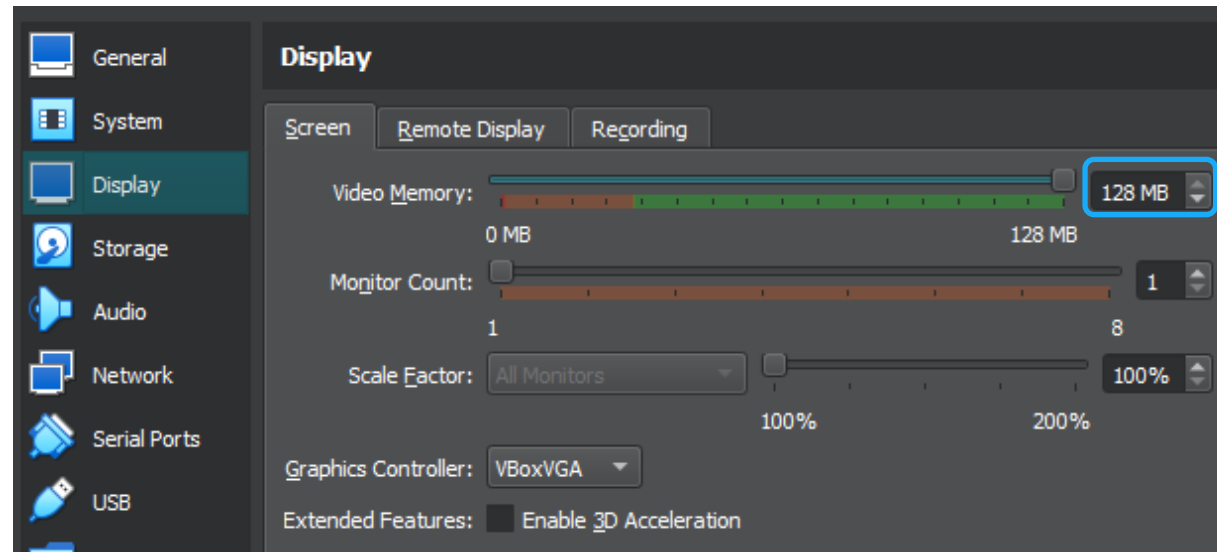
Important



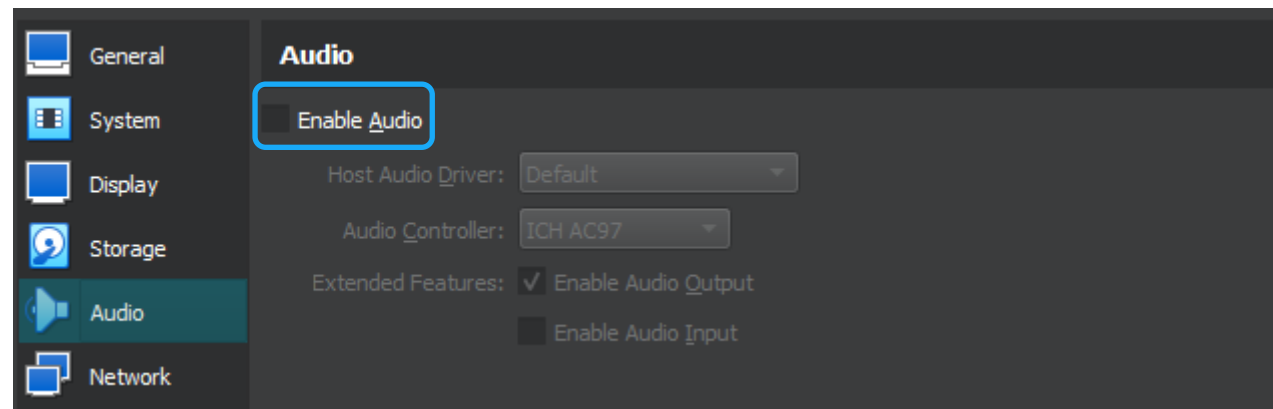
6b. Modify VM (VirtualBox)



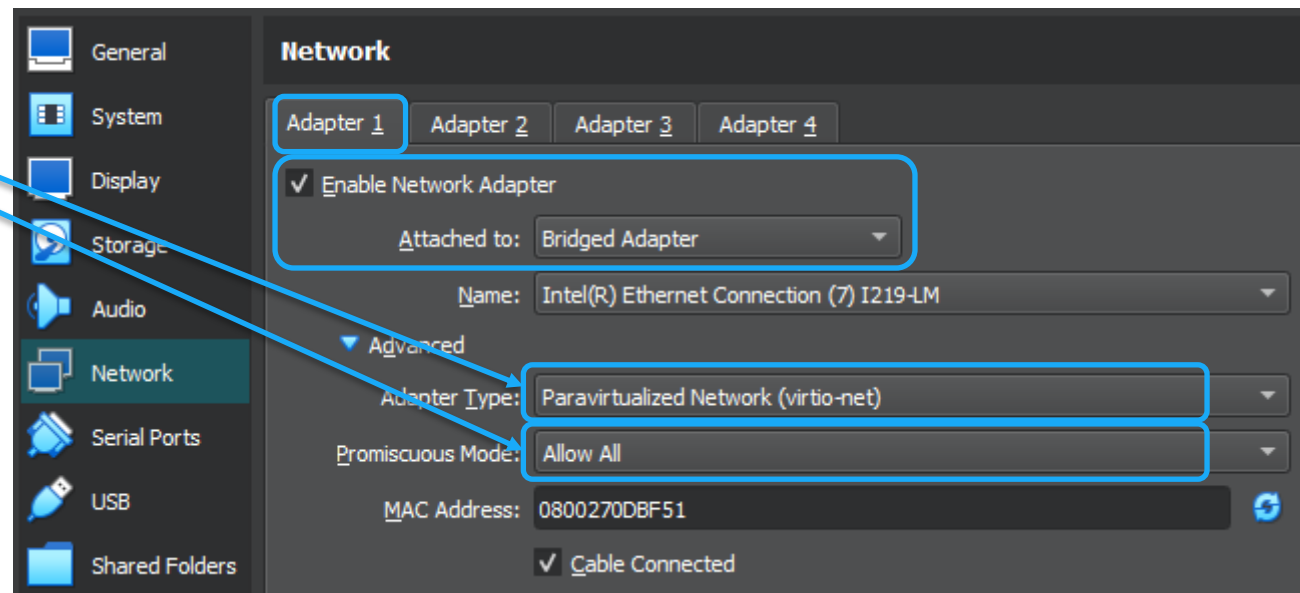
6b. Modify VM (VirtualBox)



6b.Modify VM (VirtualBox)

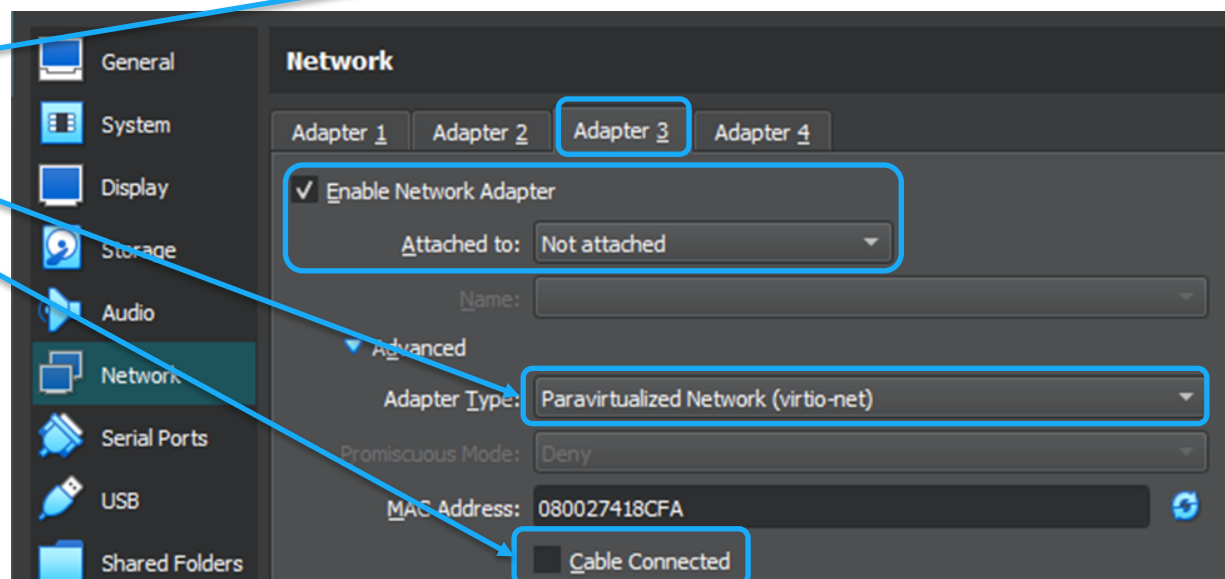
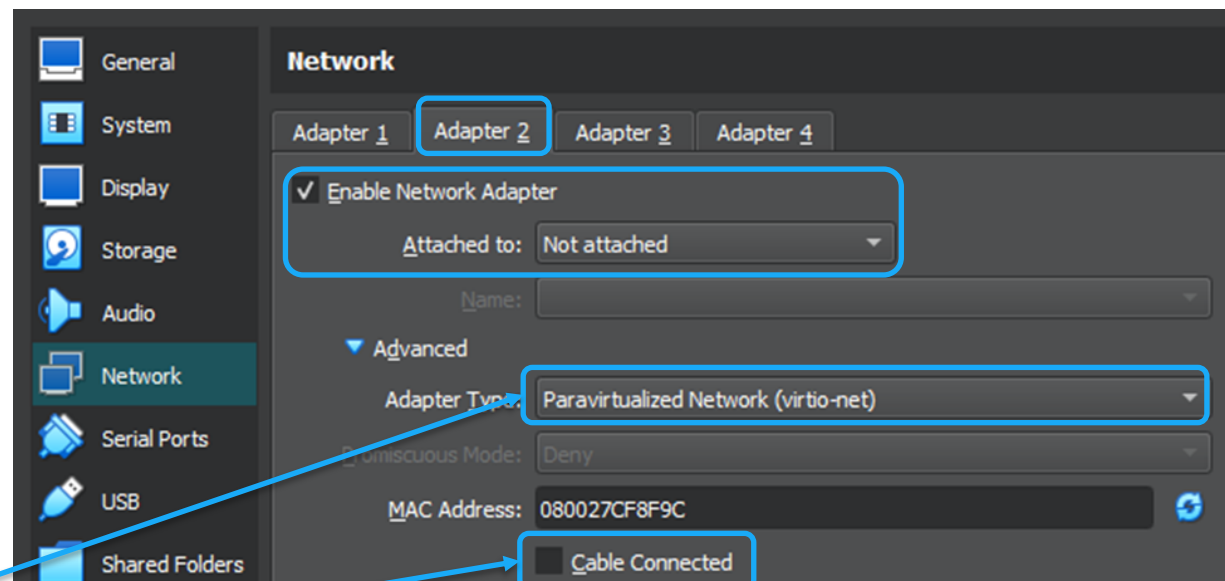


Important

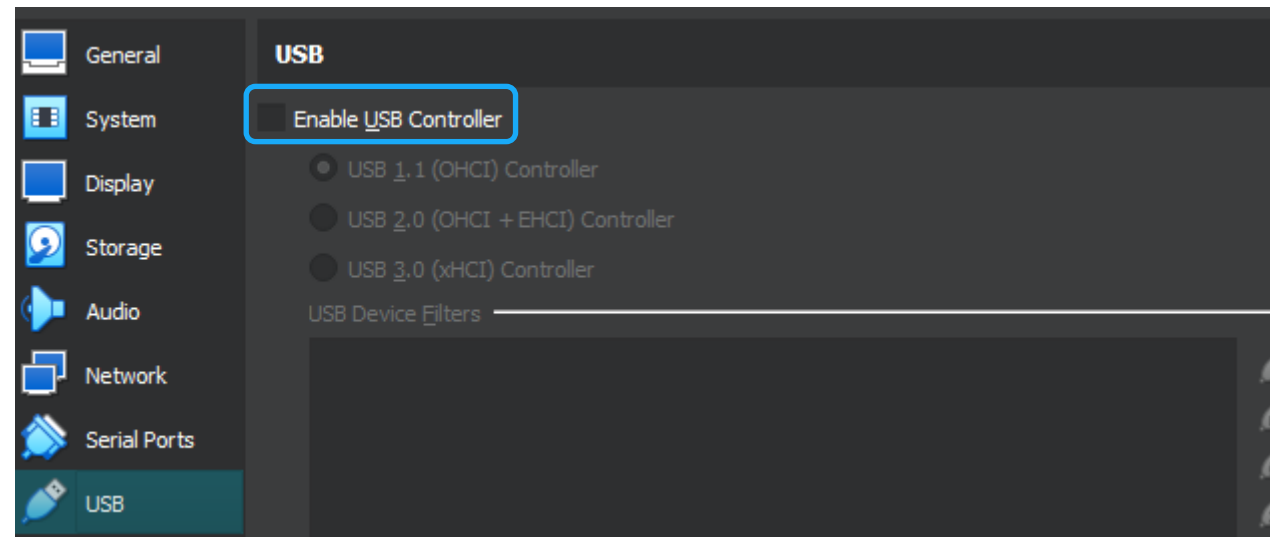
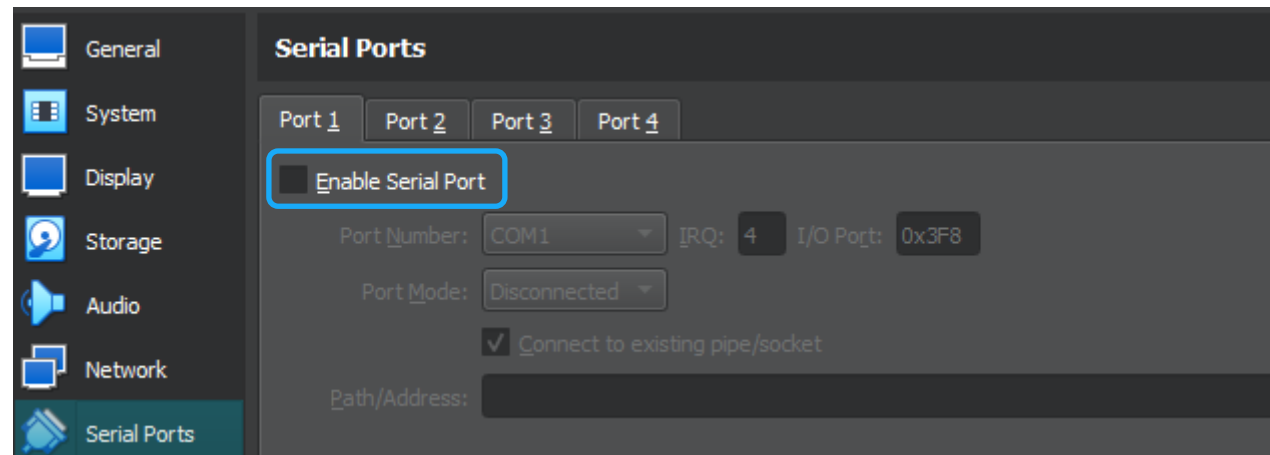


6b. Modify VM (VirtualBox)

Important

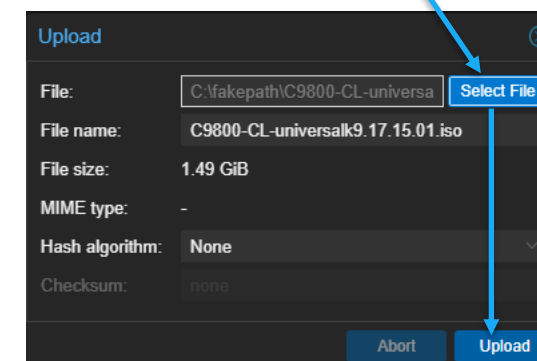
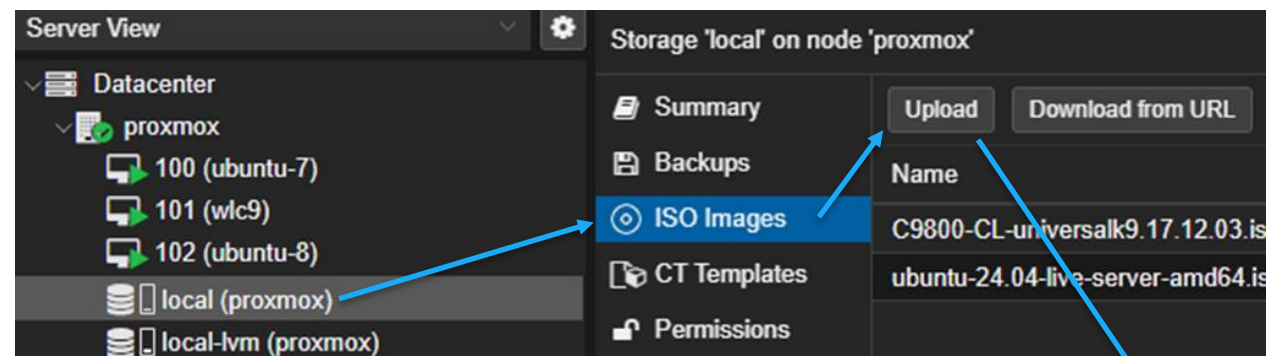


6b. Modify VM (VirtualBox)



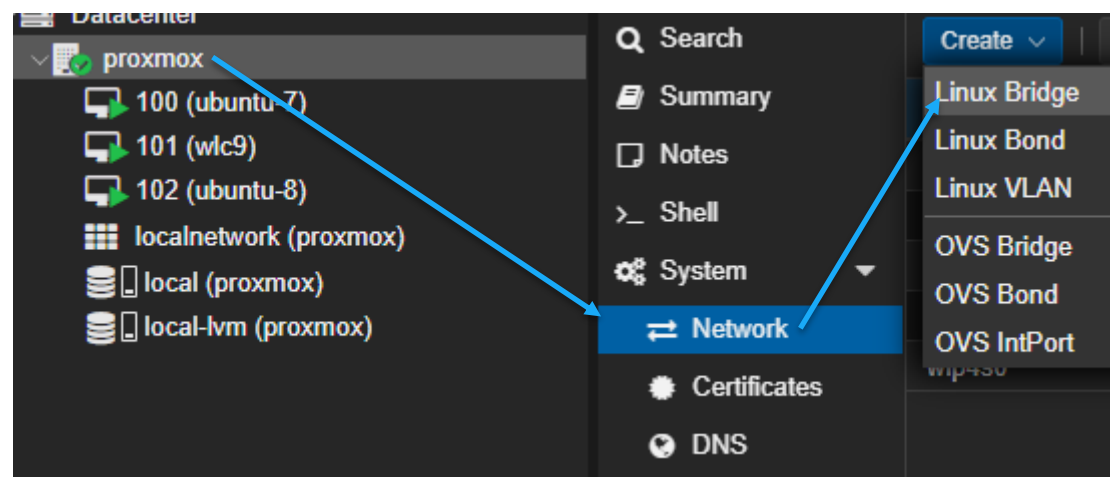
6d.Create VM (Proxmox)

First upload the ISO you will use.
Go to Datacenter>proxmox>local (proxmox)
Select "ISO images" and "Upload"



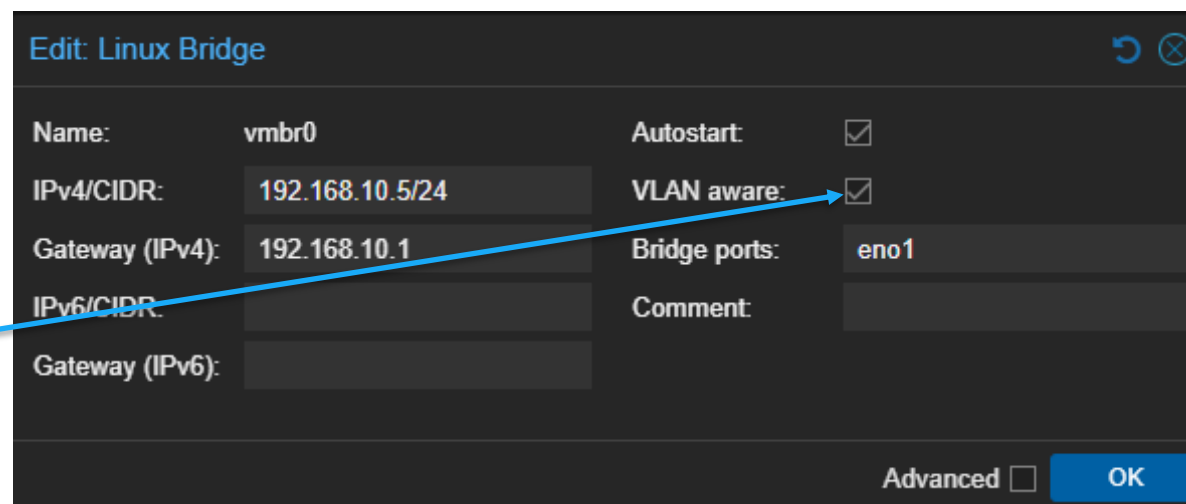
6d.Create VM (Proxmox)

Find and modify the network bridge "vmbro" (it should already exist and be bound to the wired network adapter) which we will use to get trunking and tagged VLANs to work



The IP and binding to the ethernet adapter should already be in place

The only change you need to do is
Select "VLAN aware": Enabled



Reboot the proxmox node after changing the Network adapter



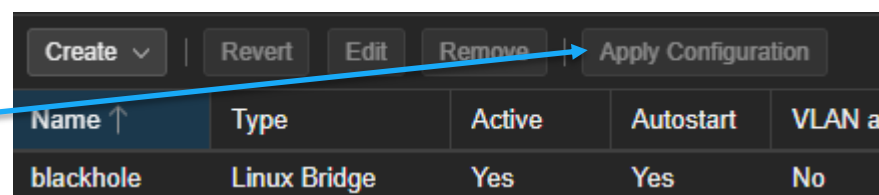
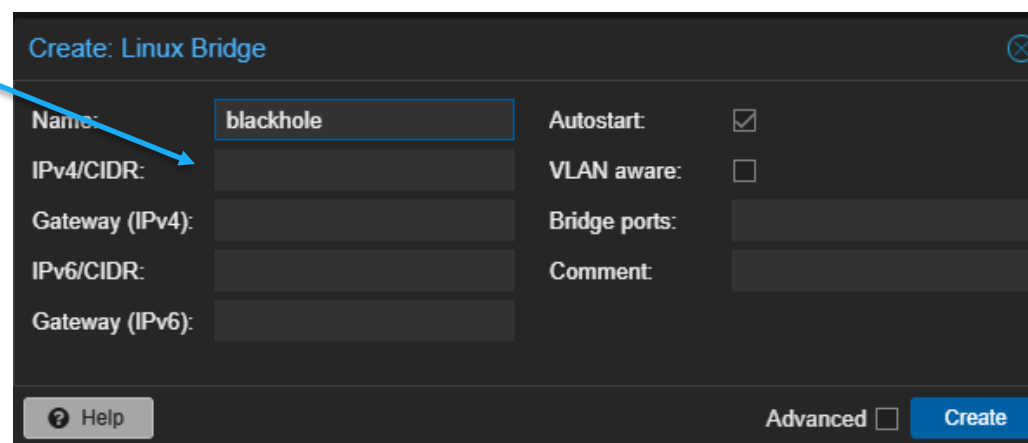
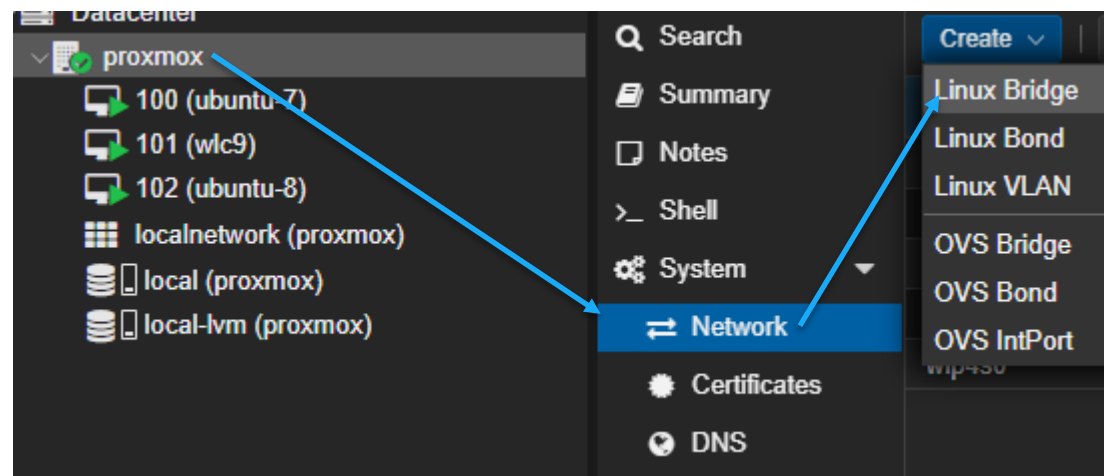
6d.Create VM (Proxmox)

Create a "blackhole" adapter, that can be assigned to ports that should exist but should not reach anywhere in your setup

The creation is similar to the bridge adapter from previous step, just don't put anything into the IP, gateway or bridge ports fields

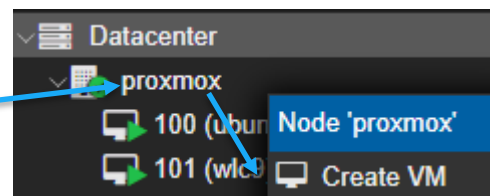
Examples of where I use blackhole adapters, are for the HA sync port, out of bands port etc. If/when you will use these ports in the setup they must be assigned another port

After creating the adapter, press "Apply Configuration"



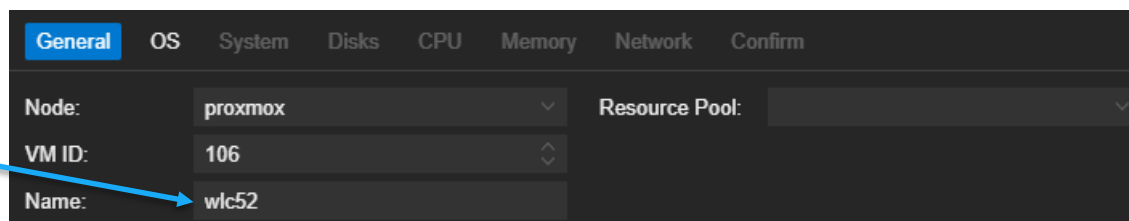
6d.Create VM (Proxmox)

Create the VM by right-clicking "proxmox" and then "Create VM"

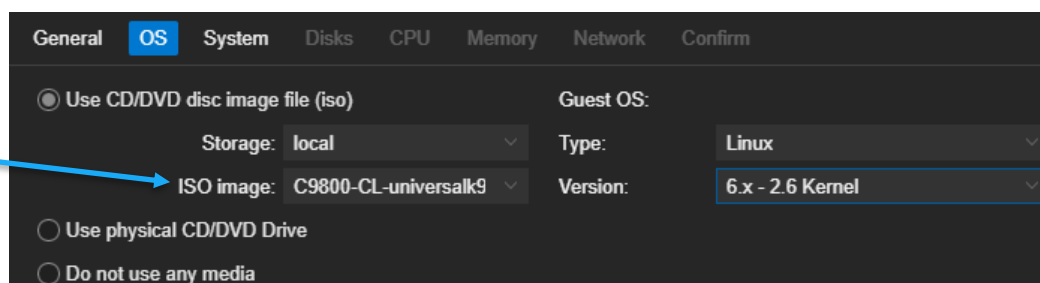


Name the VM

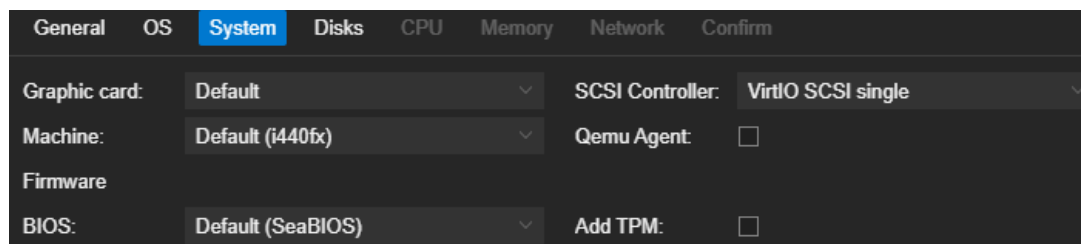
The VM will get the next free VM ID number (you can change if you want)



Select the ISO



Keep the System parameters at the default values



6d.Create VM (Proxmox)

Change the disk size to 16GiB

General OS System **Disks** CPU Memory Network Confirm

scsi0

Disk Bandwidth

Bus/Device: SCSI 0 Cache: Default (No cache)

SCSI Controller: VirtIO SCSI single Discard: ☐

Storage: local-lvm IO thread: ☒

Disk size (GiB): 16

Change CPU to 2 cores

General OS System Disks **CPU** Memory Network Confirm

Sockets: 1 Type: x86-64-v2-AES

Cores: 2 Total cores: 2

Change to 6GiB RAM (6144MiB)

General OS System Disks CPU **Memory** Network Confirm

Memory (MiB): 6144

Select the "vmbr0" Network Bridge, and use vmxnet3 (or VirtIO parvirtualized which also should work)

General OS System Disks CPU Memory **Network** Confirm

☐ No network device

Bridge: vmbr0 Model: VMware vmxnet3

VLAN Tag: no VLAN MAC address: auto

Firewall: ☒



6d.Create VM (Proxmox)

Confirm, but do not start the VM yet

Click the unstarted VM and go to "Hardware"

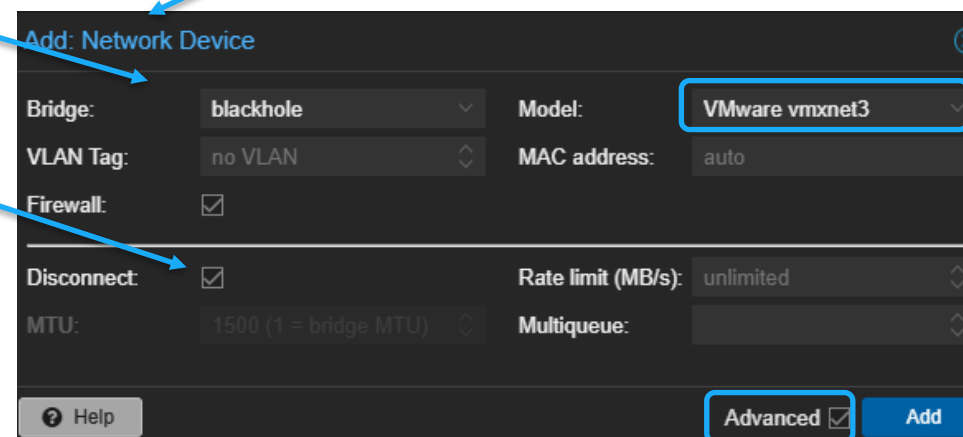
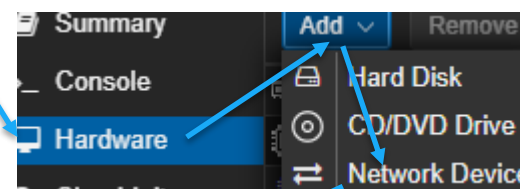
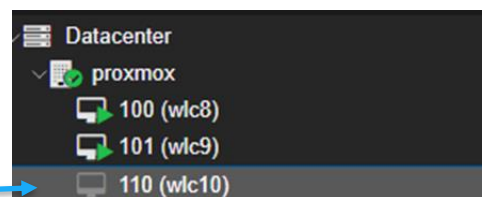
"Add" and "Network Device", to create the second and third network adapters

Use the "blackhole" network adapter you created earlier

Make the adapters 2 and 3 Disconnected

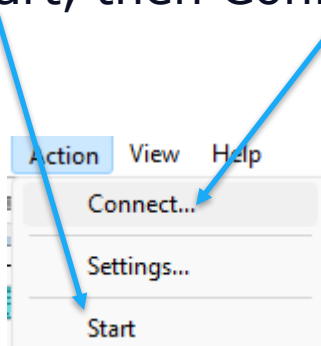
You should now have 3 "Network Device" adapters:

Network Device (net0)	vmxnet3=BC:24:11:95:AF:FB,bridge=vbr0,firewall=1
Network Device (net1)	vmxnet3=BC:24:11:A4:A6:CA,bridge=blackhole,firewall=1,link_down=1
Network Device (net2)	vmxnet3=BC:24:11:9D:46:B2,bridge=blackhole,firewall=1,link_down=1

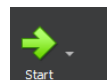


WLC common instructions - Start the 9800-CL VM

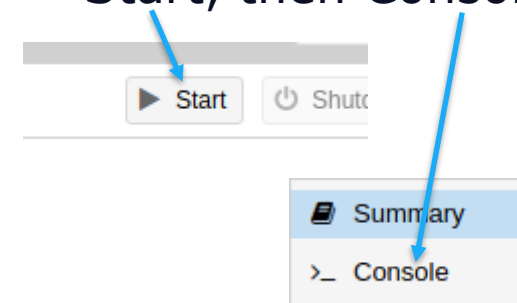
Hyper-V:
Start, then Connect



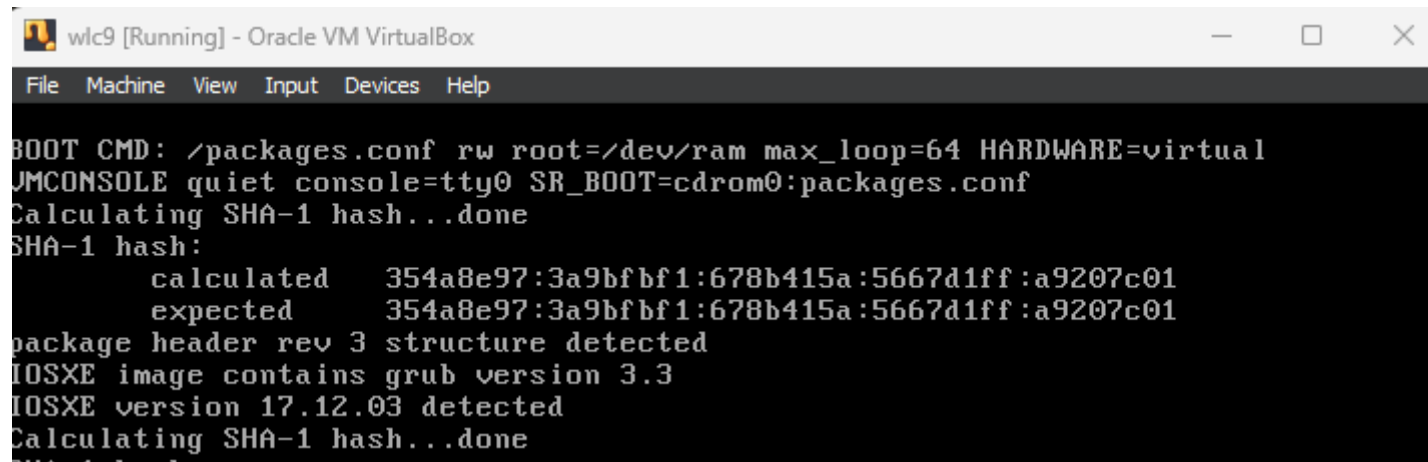
VirtualBox:
Start



Proxmox:
Start, then Console



Run VM

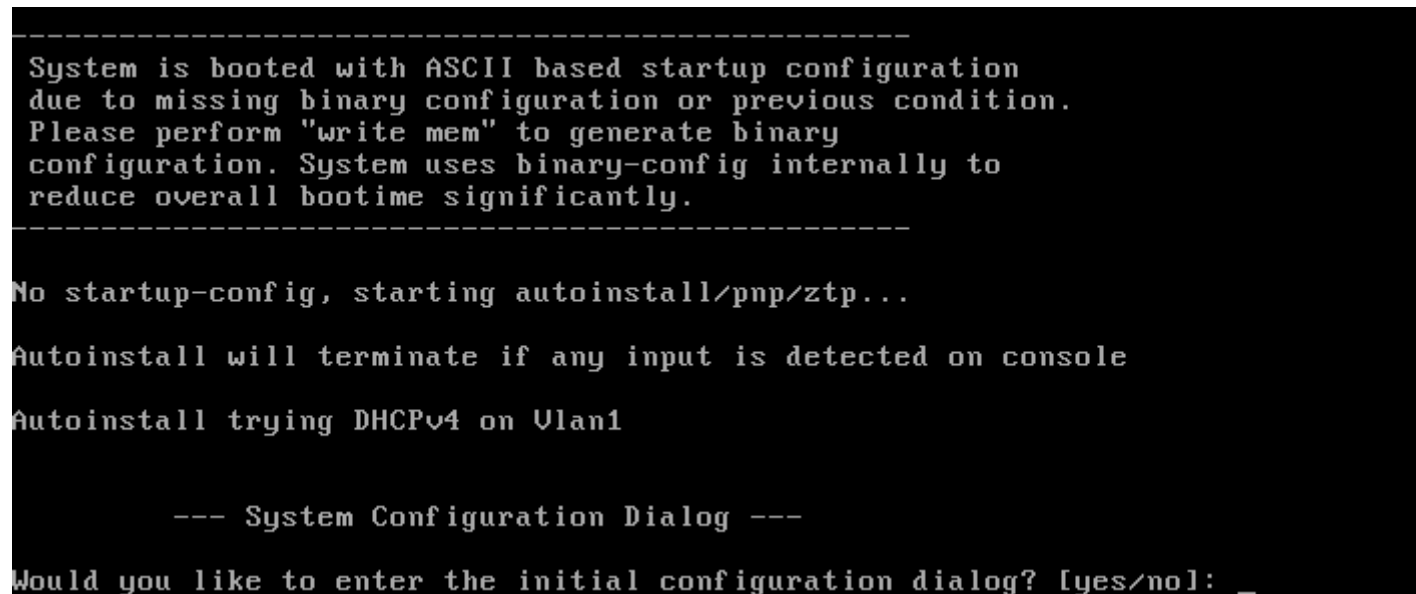


```
wlc9 [Running] - Oracle VM VirtualBox
File Machine View Input Devices Help

BOOT CMD: /packages.conf rw root=/dev/ram max_loop=64 HARDWARE=virtual
UMCONSOLE quiet console=tty0 SR_BOOT=cdrom0:packages.conf
Calculating SHA-1 hash...done
SHA-1 hash:
    calculated    354a8e97:3a9bfbf1:678b415a:5667d1ff:a9207c01
    expected      354a8e97:3a9bfbf1:678b415a:5667d1ff:a9207c01
package header rev 3 structure detected
IOSXE image contains grub version 3.3
IOSXE version 17.12.03 detected
Calculating SHA-1 hash...done
SHA-1 hash:
    calculated    354a8e97:3a9bfbf1:678b415a:5667d1ff:a9207c01
    expected      354a8e97:3a9bfbf1:678b415a:5667d1ff:a9207c01
```

Take some minutes to boot...

It looks like this when first boots are completed and ready for config:



```
-----
System is booted with ASCII based startup configuration
due to missing binary configuration or previous condition.
Please perform "write mem" to generate binary
configuration. System uses binary-config internally to
reduce overall boottime significantly.
-----

No startup-config, starting autoinstall/pnp/ztp...

Autoinstall will terminate if any input is detected on console

Autoinstall trying DHCPv4 on Vlan1

--- System Configuration Dialog ---

Would you like to enter the initial configuration dialog? [yes/no]: _
```



Configure WLC

Type "no" to enter initial configuration dialog

```
--- System Configuration Dialog ---  
Would you like to enter the initial configuration dialog? [yes/no]: _
```

Enter your enable secret twice (from topology sheet or create your own)

```
The enable secret is a password used to protect  
access to privileged EXEC and configuration modes.  
This password, after entered, becomes encrypted in  
the configuration.  
-----  
secret should be of minimum 10 characters and maximum 32 characters with  
at least 1 upper case, 1 lower case, 1 digit and  
should not contain [cisco]  
-----  
Enter enable secret: *****  
Confirm enable secret: *****
```

Enter "0" to return to CLI without saving

```
The following configuration command script was created:  
  
enable secret 9 $9$Ceibr8MFw2y9W.$NG1kD.b/c2u6Y0992/VkxN..04UDSQVV0HowSM1M1Qo  
!  
end  
  
[0] Go to the IOS command prompt without saving this config.  
[1] Return back to the setup without saving this config.  
[2] Save this configuration to nvram and exit.  
  
Enter your selection: 0_
```



Initial config

- Enter these lines manually to prepare the WLC for SSH connection
- Replace the text {pod-number} with your pod number

Full input commands

```
enable
configure terminal
hostname wlc{pod-number}
ip domain name automationlab.local
crypto key generate rsa modulus 4096
!
vlan 10
!
interface vlan 10
ip address 192.168.10.{pod-number} 255.255.255.0
no shutdown
!
interface GigabitEthernet1
switchport trunk allowed vlan 10,11
switchport trunk native vlan 10
switchport
switchport mode trunk
!
aaa new-model
username devnet-adm privilege 15 algorithm-type sha256 secret ChangeMe2025!
enable algorithm-type sha256 secret ChangeMe2025!
end
write mem
reload
```

Abbreviated commands (pro/lazy version)

(yes it is possible to abbreviate even more if you want)

```
en
conf t
ho wlc{pod-number}
ip dom name automationlab.local
cr key gen rsa mod 4096
!
vlan 10
!
int vl 10
ip add 192.168.10.{pod-number} 255.255.255.0
no shut
!
int g1
swi trun all vlan 10,11
swi trun nat vlan 10
swi
swi mo trun
!
aaa new
user devnet-adm priv 15 algo sha2 sec ChangeMe2025!
en algo sha2 sec ChangeMe2025!
end
wr
reload
```



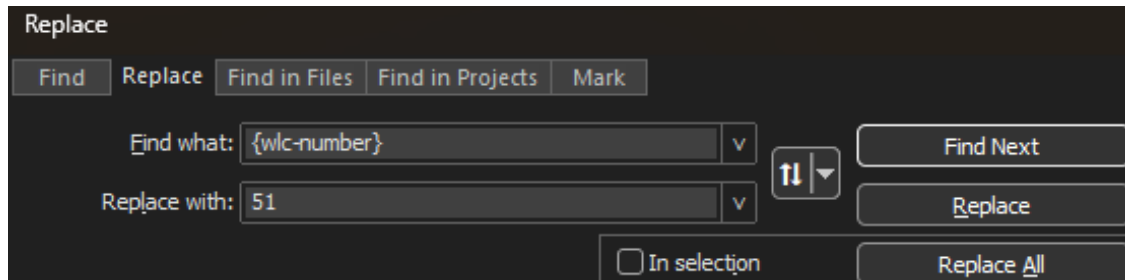
Copy this to Notepad++

Replace {wlc-number} with number

Then paste in conf mode on 9800:

Initial config

- Copy the config script , and paste into the WLC (Log in, and enter "enable" mode first)
- Replace the text {wlc-number} with your pod number



- It should be 3 occurrences
 - SSID name: wlan lab-network 17 lab-network{wlc-number}
 - Mobility group: wireless mobility group name automationlab{wlc-number}
 - RF-group: wireless rf-network automationlab{wlc-number}
- You should change/add your local country under the "wireless country" section
- The last line should be pasted alone after the rest is finished

```

conf t
vlan 10
name management
!
interface Vlan 1
shutdown
!
interface vlan 10
description management
interface GigabitEthernet1
description uplink
!
ip default-gateway 10.10.10.1
ip route 0.0.0.0 0.0.0.0 10.10.10.1
aaa authentication login default local
aaa authentication enable default enable
!
interface GigabitEthernet1
description Out-of-Band
shutdown
!
interface GigabitEthernet1
description HA-sync
shutdown
!
vlan 11
name lab-network
!
do terminal length 0
ip dhcp bootp ignore
no service psm
no psm
service tcp-keepalives-in
service tcp-keepalives-out
ntp server 8.pool.ntp.org
ntp server 1.pool.ntp.org
service timestamps debug datetime msec localtime
service timestamps log datetime msec localtime
service password-encryption
aaa login-lab comen
aaa authorization exec default local
!
! MFC US Timezone Settings (Phoenix, Arizona - Mountain Standard Time all year)
clock timezone MST -7 0
! MFC EU Timezone Settings (Europe/Dublin - CET (UTC+1) in winter and CEST (UTC+2) in summer)
clock timezone CET 1 0
! Clock Summer Time CEST recurring last Sun Mar 2:00 last Sun Oct 3:00
ip http server
ip http authentication local
ip http secure-server
no logging console
no logging monitor
logging buffered 4096
configuration mode exclusive auto
errdisable recovery cause all
errdisable recovery interval 600
ip name-server 2.2.2.2 8.8.8.8
ip ssh time-out 15
!
line con 0
exec-timeout 5 0
logging synchronous
stopbits 1
line vty 10
exec-timeout 0 0
privilege level 15
logging synchronous
transport input ssh
!
wireless mobility group name automationlab{wlc-number}
wireless mobility group 0
!
wireless rf-network automationlab{wlc-number}
wireless client max-user-login 3
ap dot11 180 shutdown
ap dot11 2 shutdown
!
! Country codes to support WLC US and EU, change when used locally
wireless country US
wireless country NO
no ap dot11 2 shutdown
no ap dot11 5 shutdown
!
wireless management interface vlan10
wireless profile flex lab-flex-profile
wireless wlan 10
vlan-name 10
vlan-id 10
vlan-id 11
!
ap profile lab-ap-profile
ap user username devnet-ade password 0 ChangeMe2025! secret 0 ChangeMe2025!
ssh
!
wireless tag site lab-site-tag
no local-site
flex-profile lab-flex-profile
ap-profile lab-ap-profile
!
wlan lab-network 17 lab-network{wlc-number}
radio policy dot11 180
security ft user-the-oh
user require
security user psk set-key ascii 0 ChangeMe2025!
no security user user psk
no shutdown
!
wireless profile policy lab-policy
no central authentication
no central dhcp
no central switching
http-ty-caching
http-ty-caching
vlan 11
no shutdown
!
wireless tag policy lab-policy-tag
wlan lab-network policy lab-policy
!
ap filter name lab-filter-rule
!
ap name-regen
tag policy lab-policy-tag
tag of default-of-tag
tag site lab-site-tag
ap filter priority 100 filter-name lab-filter-rule
!
password encryption aes
key config-key password-encrypt
PasswordForEncryptionAutomationLab
PasswordForEncryptionAutomationLab
!
!
! The last line should be pasted alone, after the rest is finished
wireless config wlc-csc key-size 2048 signature-algo sha256 password 0 CertPasswordAutomationLab

```



Prepare WLC for API connections

- Enable NETCONF and RESTCONF. If you want to create separate users for that it is OK, but you can use the "devnet-adm" user for this lab

```
netconf-yang
restconf
ip http secure-server
aaa authorization exec default local
```

- You will not use this in these labs, but for your reference, here is how to create read-only users for NETCONF and RESTCONF

```
username restconf-read privilege 1 secret ChangeMe2025!
username netconf-read privilege 1 secret ChangeMe2025!
exit
request platform software yang-management nacm populate-read-rules privilege 1
```

- Note about the last command
 - Some times it will take a couple of minutes before the command is "ready to run". Wait a minute and try again if you get an error

```
wlc30#request platform software yang-management nacm populate-read-rules privilege 1
The process for the command is not responding or is otherwise unavailable

wlc30#request platform software yang-management nacm populate-read-rules privilege 1
% Error: Currently unable to process request
```



Pre-lab task summary

- You should now be ready for the deep dive labs, by having installed
 - Ubuntu Server w/Docker in a hypervisor of your choice (VirtualBox, Hyper-V, Parallels, Proxmox, etc)
 - Postman
 - VS Code w/some extensions
 - (optional) 9800-CL in a hypervisor of your choice (VirtualBox, Hyper-V, Proxmox)

