Build your own Wi-Fi automation lab

Pre-lab tasks

Andreas Koksrud / Telenor Bedrift

Kjetil Teigen Hansen / Conscia

telenor

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

Copyright

Prerequisites

- Cisco.com account with access to downloading WLC (9800-CL at https://software.cisco.com)
- Postman account (https://postman.com)
- Windows laptop with administrative privileges
  - For now, we will include some preliminary instructions for Mac
  - What will definitely NOT work on Mac, is running the C9800 WLAN Controller
- Complete the pre-lab exercises (this document) before the deep dive labs

- Without these, parts of the Deep Dive might be difficult to complete, or steps might differ severely

Communications

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

Agenda

Pre-lab

- Install VirtualBox
- Install Cisco 9800-CL
- Install Ubuntu Server w/Docker
- Install Postman
- Install VS Code

Day 1

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

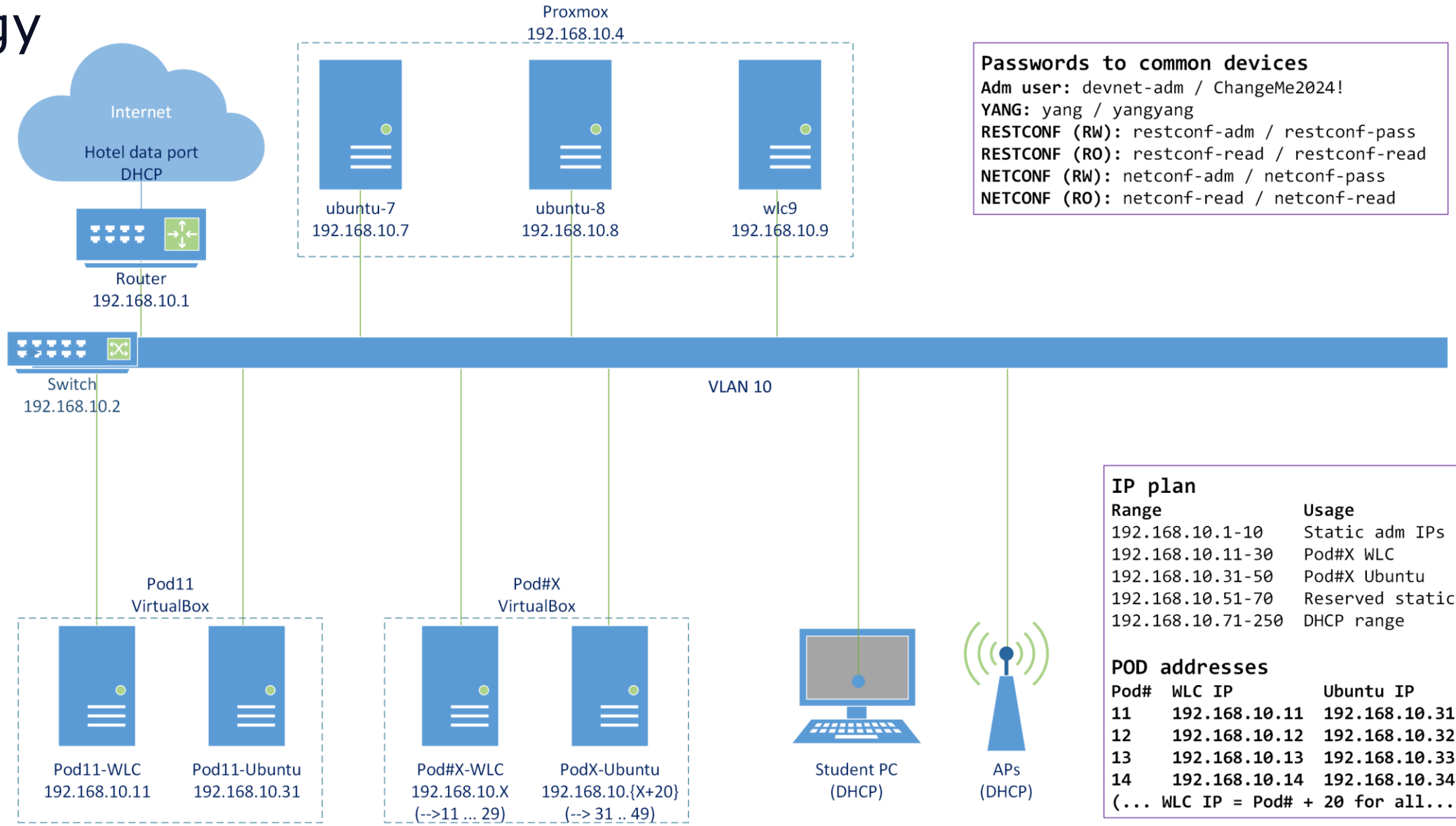
Day 2

- In-depth explore a topic of choice
  - Grafana / TIG-stack
  - Grafana Cloud
  - Ansible
  - Python
  - Other vendors (MIST/Meraki/Aruba)

Scope

- In scope
  - Getting started with various systems/languages/solutions for lab purposes
  - Get a lab environment up and running on your own 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

Topology



Passwords to common devices

Adm user: devnet-adm / ChangeMe2024!
YANG: yang / yangyang
RESTCONF (RW): restconf-adm / restconf-pass
RESTCONF (RO): restconf-read / restconf-read
NETCONF (RW): netconf-adm / netconf-pass
NETCONF (RO): netconf-read / netconf-read

IP plan

Range	Usage
192.168.10.1-10	Static adm IPs
192.168.10.11-30	Pod#X WLC
192.168.10.31-50	Pod#X Ubuntu
192.168.10.51-70	Reserved static
192.168.10.71-250	DHCP range

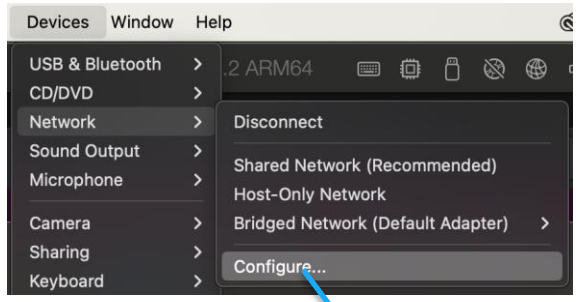
POD addresses

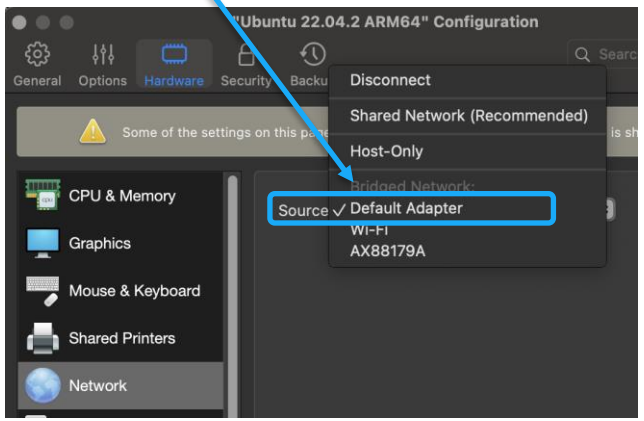
Pod#	WLC IP	Ubuntu IP
11	192.168.10.11	192.168.10.31
12	192.168.10.12	192.168.10.32
13	192.168.10.13	192.168.10.33
14	192.168.10.14	192.168.10.34

(... WLC IP = Pod# + 20 for all...)

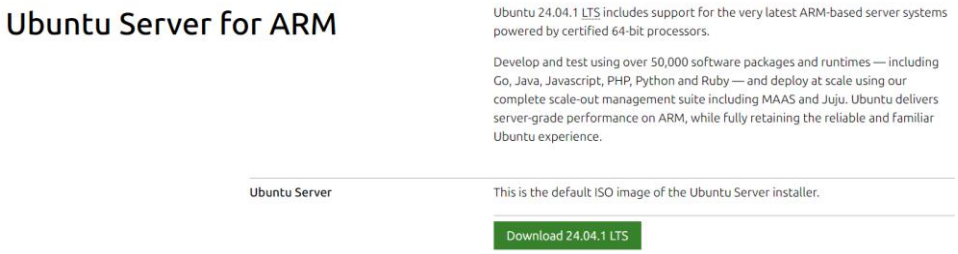
Notes for Mac users #1 - Ubuntu VM

- If you are using a newer MacOS
  - No VirtualBox version available for MacOS with Apple silicon (only for x86), as of Oct 2024
  - You can run Ubuntu in Parallels, you will need the ARM64 version of Ubuntu Server
  - Detailed instructions will not be provided, but in general it will be the same, including network adapter in bridged mode





- Since we are not Mac people, in this first version of the deep dive, no detailed instructions for Mac installation will be provided. It might be added in a future version
- Do not install the pre-selected "Ubuntu" in Parallels, it is the Desktop version with GUI and lots of preloaded programs. Good for its use, but not neccessary for this lab. It will not hurt, but the instructions will not match this guide.
- Link to Ubuntu Server for ARM64: https://ubuntu.com/download/server/arm



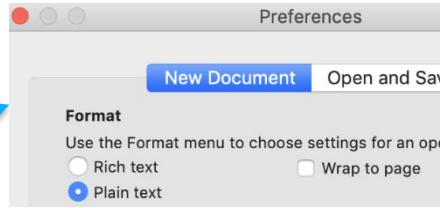
- If you would rather use an Ubuntu Server hosted on our VM, just give us a hint and we will prepare one for you ☺

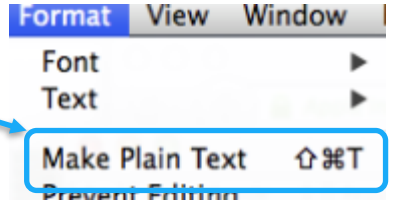
Notes for Mac users #2 - Cisco 9800 WLC

- As of now, 9800-CL will NOT run on Macs with Apple silicon. You can use the shared installation wlc9 (see topology)
- If many people need to use a hosted WLC, we can create "your own" 9800 in our virtual environment
- 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/

Notes for Mac users #3 - 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





Notes for WSL users (optional / alternative)

- You can use WSL (Windows Subsystem for Linux) instead of a Ubuntu VM running in VirtualBox for Python and Ansible stuff
- TIG stack and other docker stuff is not tested (yet)
- No instructions for using WSL or adopting the lab guide to WSL is included at this moment
- Install WSL (start Powershell as administrator)

```
PS C:\> wsl --install
```

- Check WSL version

```
PS C:\> wsl --version
WSL version: 2.3.24.0
(... if this command is not showing version, you must upgrade)
```

- Upgrade WSL

```
PS C:\> wsl --update
```

- Install Ubuntu, run "wsl --install -d Ubuntu

```
PS C:\> wsl --install -d Ubuntu
```

- List installed WSL VMs, run "wsl --list"

```
PS C:\> wsl --list
```

- Fix network connectivity (from Powershell. Restart WSL)

```
PS C:\> echo '[wsl2]' > ~\.wslconfig
PS C:\> echo 'networkingMode=mirrored' >> ~\.wslconfig
```

- Set Windows user home directory as WSL home directory
  - Do this inside the WSL Ubuntu terminal

```
koks@TNL-15143nc81:~$ sudo nano /etc/passwd
```

  - Find this line

```
koks:x:1000:1000:,,,:/home/koks:/bin/bash
```

  - Edit the home directory to your Windows home directory

```
koks:x:1000:1000:,,,:/mnt/c/Users/akoksrud:/bin/bash
```

  - Edit the wsl.conf file inside Ubuntu

```
koks@TNL-15143nc81:~$ sudo nano /etc/wsl.conf
```

  - Add the following part

```
[automount]
options = "metadata"
```

  - Copy the bash profile files to your new home directory (after re-login)

```
koks@TNL-15143nc81:~$ cp /home/koks/.bashrc ~
koks@TNL-15143nc81:~$ cp /home/koks/.profile ~
```

I only want the "automation" part, not the "building your own lab" part

- If you for some reason do NOT want to do the "building your own automation lab" part of this deep dive, only the "a taste from the automation buffet" part, you can skip some parts. We will try to note those parts at the intro slide to each part.
- In short, you can make these decisions to do less building and more automation
  - Use an Ubuntu server that we can host on a server that we bring along. This will be "your own" and we will delete it after the deep dive. The resources will be shared along with some other participants and the shared WLC
  - Use a C9800 WLC on a server that we bring along. This WLC will be shared by all students that need this. We might be able to host some more WLCs if neccessary
  - Instead of installing YANG Suite you can use it from a shared Ubuntu Server where it is already installed
  - Instead of installing the TIG stack (Grafana etc) you can use it from a shared Ubuntu Server where it is already installed
- Please note that by doing this you might have more time for automation during the deep dive, but you will be less familiar with building the items yourself when you get home or need it in some other situation.

Pre-lab task #1: VirtualBox

- In this task we will install VirtualBox
- This lab will use VirtualBox to run virtual machines (VMs) on your laptop
  - Cisco C9800-CL Wireless LAN Controller
  - Ubuntu Linux with Docker to run some different containers
- RightCtrl + F switches between full-screen and window
- Tap RightCtrl to "free" input from guest
- !!! Note !!!
  - As a rule of thumb when running VMs, you should turn OFF power save stuff on your laptop, as you might run into unexpected problems if your laptop enters power saving modes

(optional info)

The Python-based exercises can also be run directly on your host OS Python-environment. But a VM give flexibility to test stuff without messing with your production OS. Ansible can not (currently) be run on Windows so it needs a Linux VM to run. The TIG-stack could also be run on the host OS, but again for lab purposes a VM is much easier to just wipe if (when) you test some weird stuff that breaks everything.

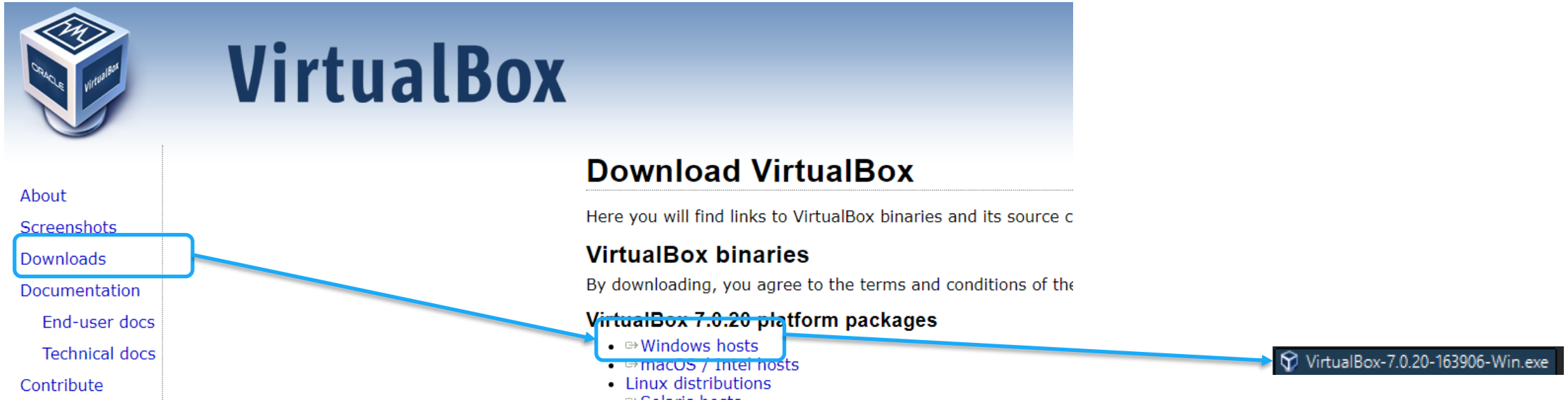
You can also use WSL (Windows Subsystem for Linux), which might be just as good or better in many situations. But this lab covers the use of a VM on VirtualBox ☺

Alternative solutions - Shared servers

- Shared servers
  - I bring with me a portable datacenter (i.e. my old work laptop...) where I have installed Proxmox Virtual Environment. On that, I have installed a C9800, and a couple of Ubuntu servers
  - If you for any reason are unable to spin up 9800 or Ubuntu on your own laptop, just notice me and you kan use the shared servers
    - If you do not have admin privileges, VirtualBox will not work
    - If your laptop is weak, you might be able to run Ubuntu server, but can use the shared WLC on proxmox (wlc9, see topology)
  - 1x 9800 and 3-4 Ubuntu servers should run fine (we might test with more if neccessary)

VirtualBox installation

https://virtualbox.org/



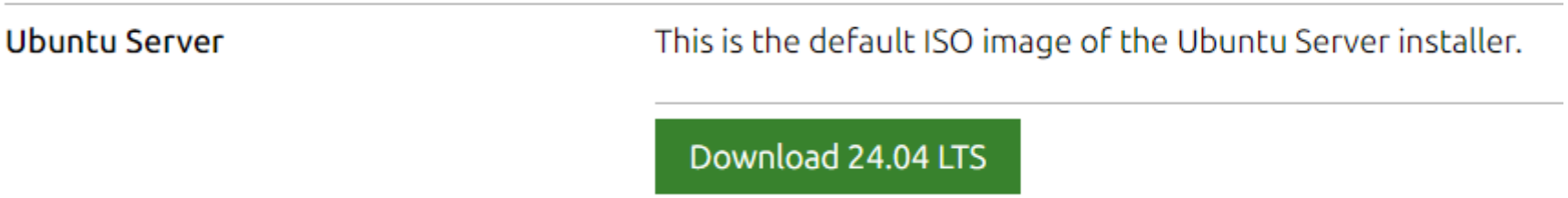
- Needs admin privileges
- Not all steps are shown here, it is mostly "Next-next-next"
- As default folder for VMs, I use "C:\Virtual Machines", you can use whatever, but screenshots here will show this path

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

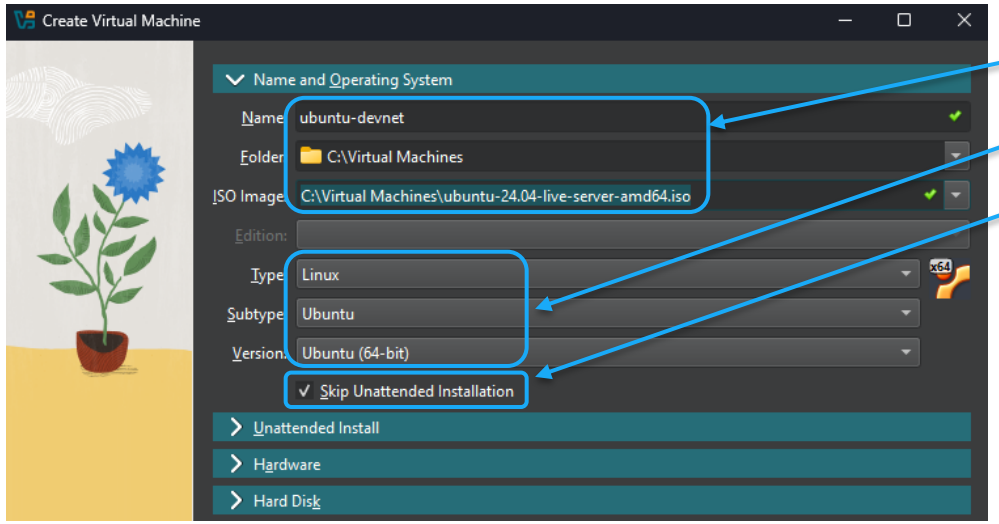
Pre-lab task #2: Ubuntu Server

- Download Ubuntu Server 24.04
  - https://ubuntu.com/download/server
- If using MacOS, download the ARM64 version, and use Parallels
  - https://ubuntu.com/download/server/arm

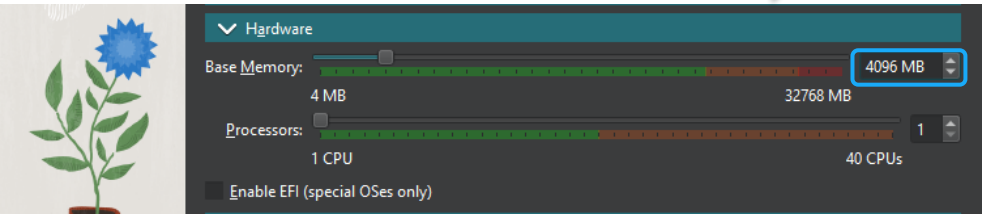


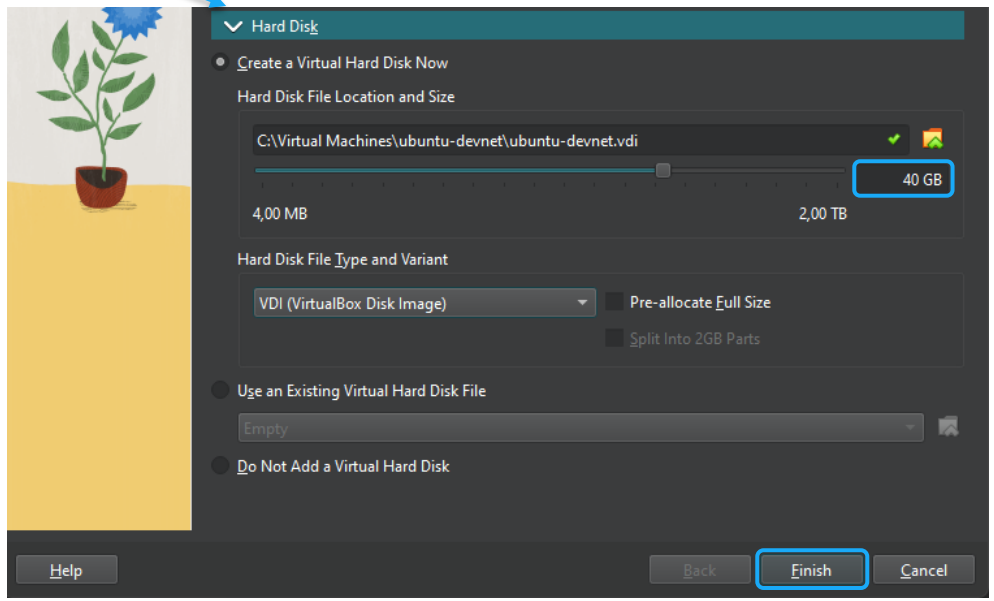
- If using MacOS, see slide 10 "Alternative solutions" for network interface config on the Parallels side

Create Ubuntu server (VirtualBox GUI screenshots)

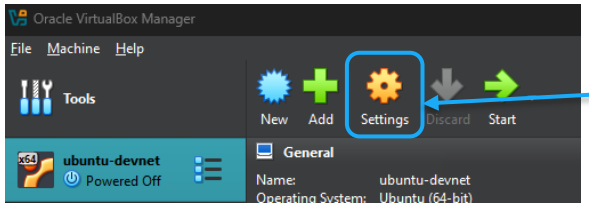


- Set the name, folder and select the ISO that you downloaded
- Select Type as shown
- Check «Skip Unattended Installation»
- Expand the «Hardware» section. Increase to 4096MB
- Expand the «Hard Disk» section. Increase to 40GB, then Finish

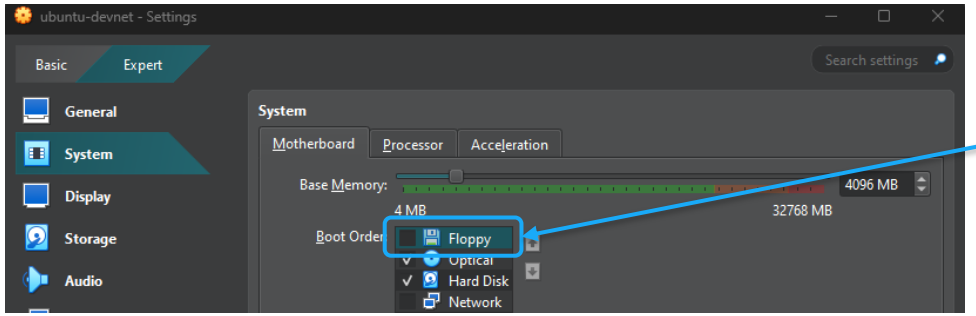




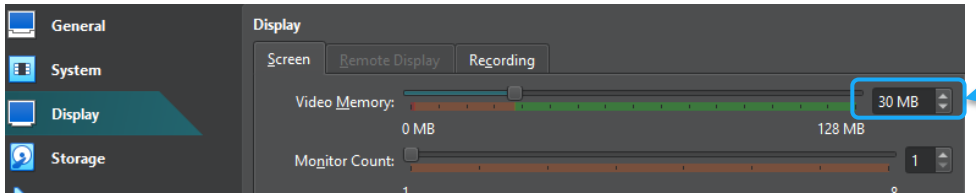
Create Ubuntu server (VirtualBox GUI screenshots)



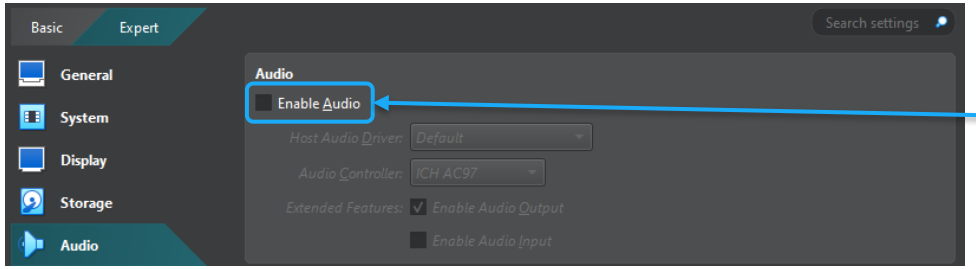
- Enter settings
- General: No change needed



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

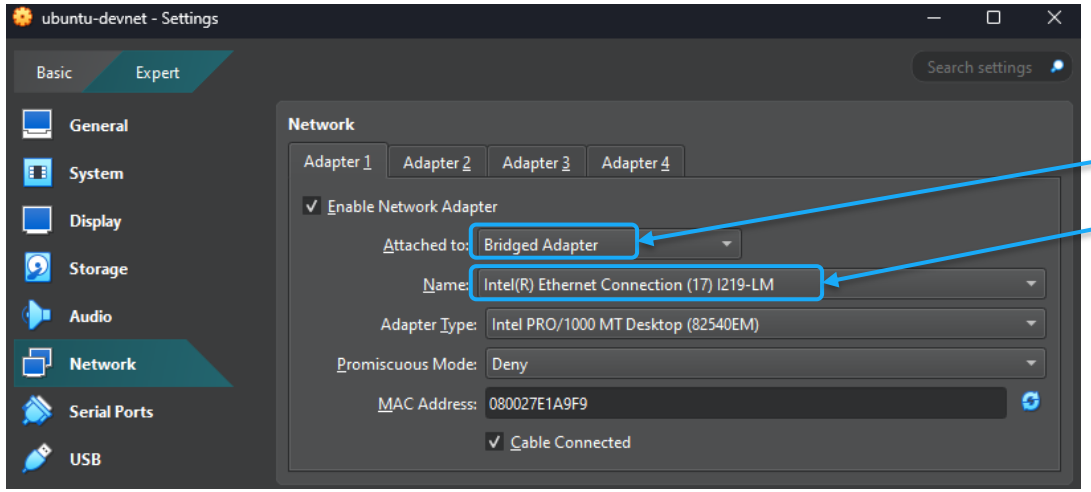


- Display: Ensure «Video Memory» is at least 30MB. Keep the rest default
- Storage: No change needed



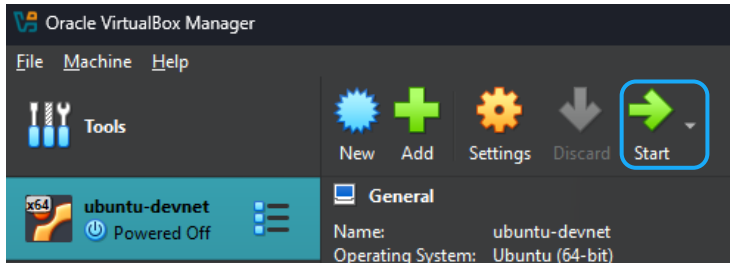
- Audio: Remove «Enable Audio» checkbox

Create Ubuntu server (VirtualBox GUI screenshots)



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

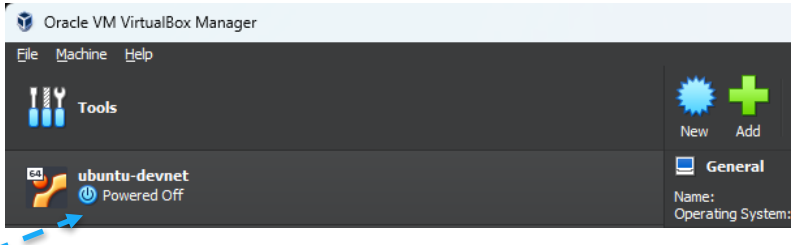
- Click OK to save the changes



- 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)

Installing Ubuntu Server (optional - CLI alternative)

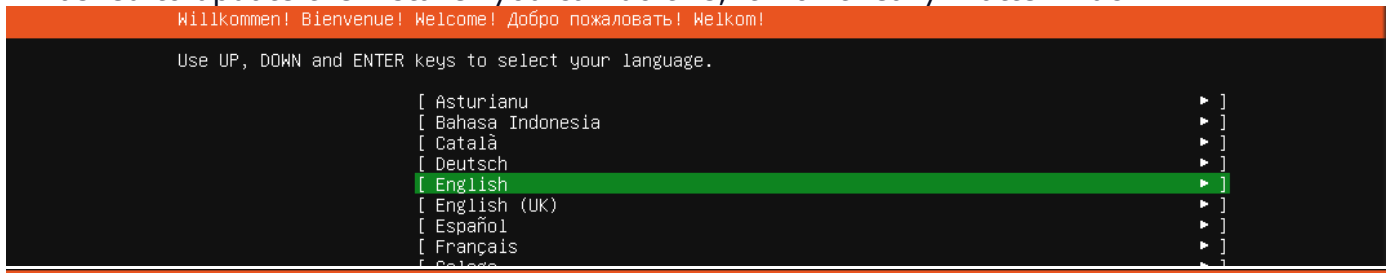
- Start PowerShell
- Go to folder where you have your VMs
  - The example uses 'c:\Virtual Machines', change to your directory accordingly
  - cd 'c:\Virtual Machines\'
  - Create and start the VM, follow these instructions
  - (copy-paste is possible from this box)

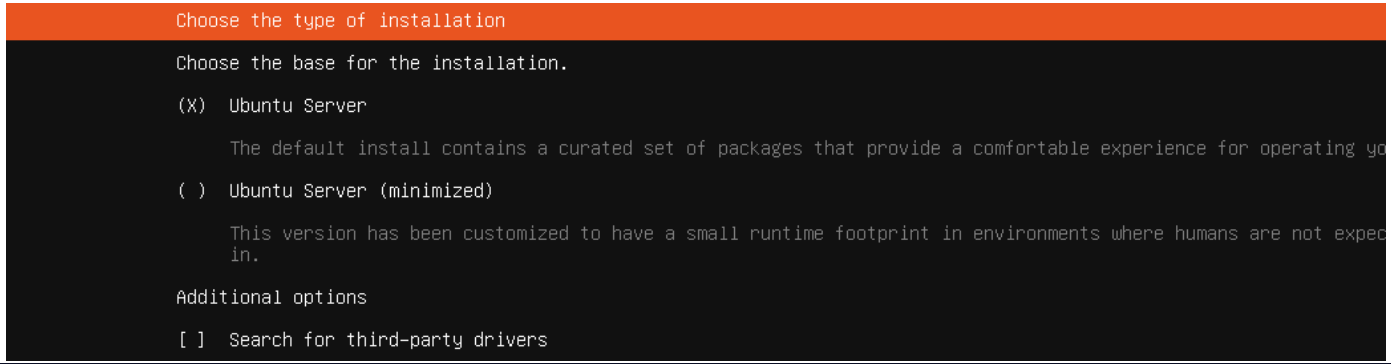



```
PS C:\> cd 'Virtual Machines'
PS C:\Virtual Machines>
PS C:\Virtual Machines> VBoxManage createvm --name "ubuntu-devnet" --ostype Ubuntu_64 --register
PS C:\Virtual Machines> VBoxManage modifyvm "ubuntu-devnet" --cpus 1 --memory 4096 --vram 30 --graphicscontroller vmsvga --usbohci on --mouse usbtablet --pae
off --boot1 disk --boot2 dvd --boot3 none --audio-enabled off
PS C:\Virtual Machines> VBoxManage createhd --filename ".\ubuntu-devnet\ubuntu-devnet.vdi" --size 40000 --variant Standard
PS C:\Virtual Machines> VBoxManage storagectl "ubuntu-devnet" --name "SATA Controller" --add sata --bootable on
PS C:\Virtual Machines> VBoxManage storageattach "ubuntu-devnet" --storagectl "SATA Controller" --port 0 --device 0 --type hdd --medium ".\ubuntu-
devnet\ubuntu-devnet.vdi"
PS C:\Virtual Machines> VBoxManage storagectl "ubuntu-devnet" --name "IDE Controller" --add ide
PS C:\Virtual Machines> VBoxManage storageattach "ubuntu-devnet" --storagectl "IDE Controller" --port 0 --device 0 --type dvddrive --medium ".\ubuntu-24.04-
live-server-amd64.iso"
PS C:\Virtual Machines> ipconfig /all
<<< Copy the "Description" for the NIC you will use. For me it is "Intel(R) Ethernet Connection (7) I219-LM" >>>
PS C:\ Virtual Machines> VBoxManage modifyvm "ubuntu-devnet" --nic1 bridged --cable-connected1 on --bridgeadapter1 "Intel(R) Ethernet Connection (7) I219-LM"
PS C:\ Virtual Machines> VBoxManage startvm "ubuntu-devnet"
```

Ubuntu server install

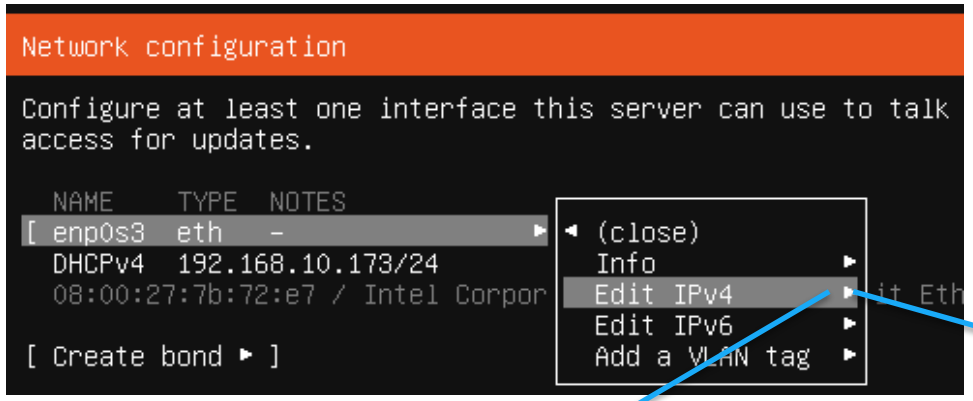
- Frequent updates to Ubuntu can make the following pages somewhat different, but the basics should be recognizable
- If asked to update the installer you can do this, it won't really matter much

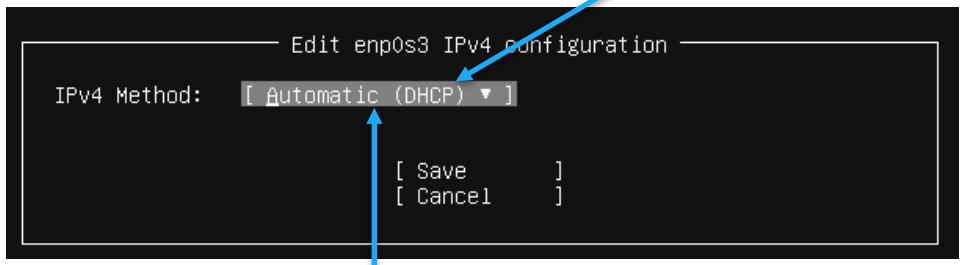


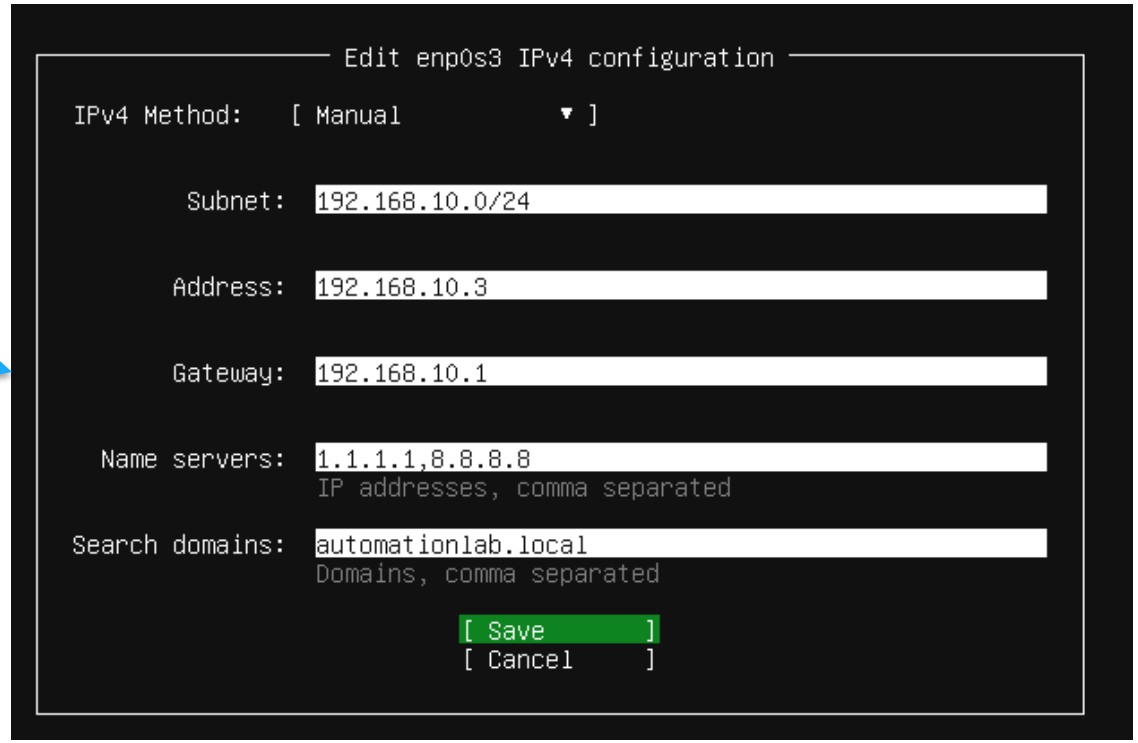


- Choose your own keyboard layout (unless you just love the norwegian æøå characters and funky placement of all the useful characters like dash and slash)

Ubuntu server install - Network configuration

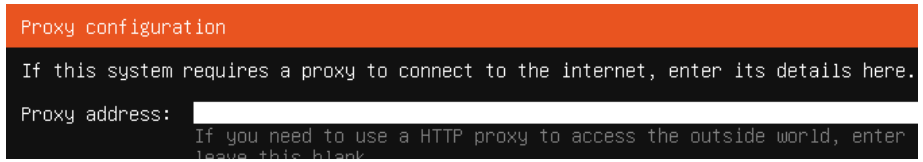






For pre-lab install you can use DHCP or static IP in your own environment. When in the lab environment, you can change to (another) static IP. See later slide on static vs DHCP.

Leave this blank



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

```
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  ▸
```

```
(close)
Edit
Delete
```

```
Editing logical volume ubuntu-lv of

Name:   ubuntu-lv

Size (max 37.996G):  37.996G

Format:  [ ext4          ▼ ]

Mount:   [ /             ▼ ]

[ Save   ]
[ Cancel ]
```

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

```
devnet-adm@ubuntu-devnet:~$ sudo lvdisplay | 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 syste
password is still needed for sudo.

            Your name:  Andreas Koksrud

    Your servers name:  ubuntu-devnet
                        The name it uses when it talks to other com

      Pick a username:  devnet-adm

    Choose a password:  **************   ChangeMe2024!

Confirm your password:  **************   ChangeMe2024!
```

```
SSH Setup

You can choose to install the OpenSSH server package to enable s

                [X]  Install OpenSSH server

Import SSH identity:  [ No              ▼ ]
                      You can import your SSH keys from GitHub o

    Import Username:

                [X]  Allow password authentication over SSH
```

```
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.

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

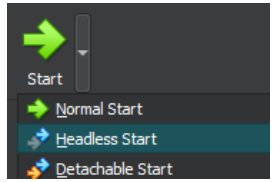
```
ubuntu-devnet login: devnet-adm
Password:
Welcome to Ubuntu 24.04 LTS (GNU/Linux 6.8.0-35-generic x86_64)

 * 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

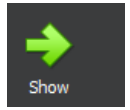
```
devnet-adm@ubuntu-devnet:~$ sudo apt update
devnet-adm@ubuntu-devnet:~$ sudo apt upgrade
```

Connecting to the server using SSH

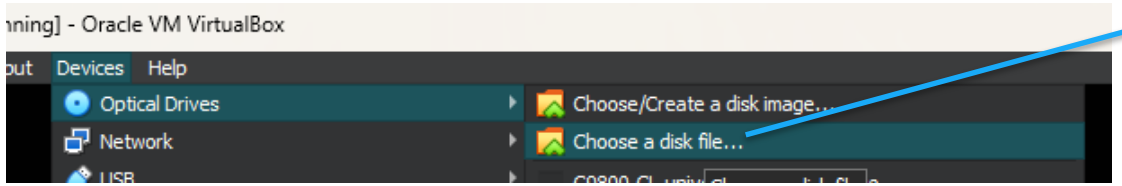
- Now would be a great time to connect to the Ubuntu server using your favourite SSH client. Like MobaXterm, SecureCRT, PuTTY, etc. You can also use the ssh CLI tool in your preferred shell
- The main reason for this is copy-paste, which you will use during these pre-lab tasks, and much more in the deep 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"☺
- 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.

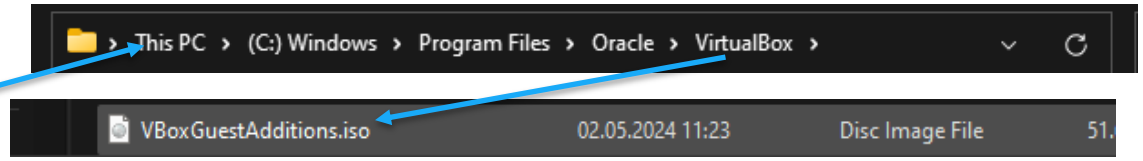


- If you need the console when started headless, just use the "Show" button found where the Start button usually is before the VM is started



VirtualBox Guest Additions

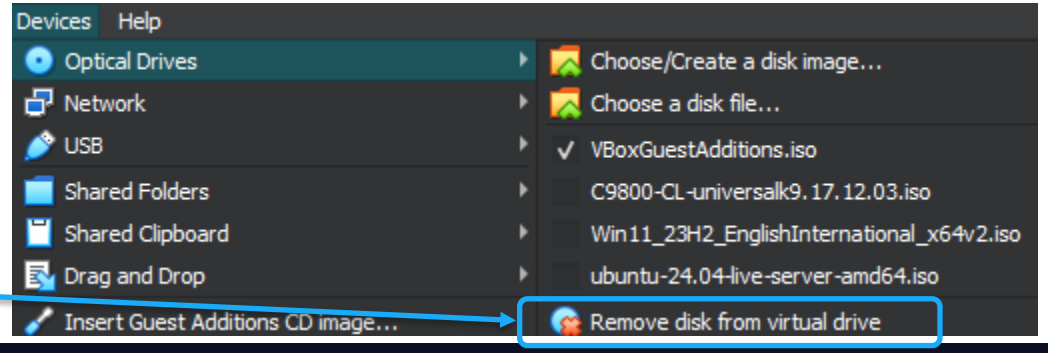




- RightCtrl + F switches between full-screen and window
- Tap RightCtrl to "free" input from guest

```
devnet-adm@ubuntu-devnet:~$ sudo apt install build-essential dkms linux-headers-$(uname -r)
devnet-adm@ubuntu-devnet:~$ sudo mkdir -p /mnt/cdrom
devnet-adm@ubuntu-devnet:~$ sudo mount /dev/cdrom /mnt/cdrom
mount: /mnt/cdrom: WARNING: source write-protected, mounted read-only.
devnet-adm@ubuntu-devnet:~$ cd /mnt/cdrom
devnet-adm@ubuntu-devnet:~$ sudo ./VBoxLinuxAdditions.run
devnet-adm@ubuntu-devnet:~$ sudo reboot
```

```
devnet-admin@ubuntu-devnet:/mnt/cdrom$ sudo ./VBoxLinuxAdditions.run
Verifying archive integrity...  100%   MD5 checksums are OK. All good.
Uncompressing VirtualBox 7.0.18 Guest Additions for Linux  100%
VirtualBox Guest Additions installer
Copying additional installer modules ...
Installing additional modules ...
VirtualBox Guest Additions: Starting.
VirtualBox Guest Additions: Setting up modules
VirtualBox Guest Additions: Building the VirtualBox Guest Additions kernel
```



Static IP vs DHCP

- One method to change between DHCP and static IP, is to edit the YAML files in /etc/netplan/
- We just create a new file called 99_config.yaml, and when we run "netplan apply" it will use that file for config. We delete the auto-created "50-cloud-init.yaml"

```
devnet-admin@ubuntu-devnet:~$ netplan status
     Online state: online
    DNS Addresses: 127.0.0.53 (stub)
       DNS Search: .

●  1: lo ethernet UNKNOWN/UP (unmanaged)
      MAC Address: 00:00:00:00:00:00
        Addresses: 127.0.0.1/8
                   ::1/128

●  2: enp0s3 ethernet UP (networkd: enp0s3)
      MAC Address: 08:00:27:31:cf:1e (Intel Corporation)
        Addresses: 192.168.10.7/24
                   fdef:c834:a92d:6a49:a00:27ff:fe31:cf1e/64
```

Change this to whatever name you have here

```
devnet-adm@ubuntu-devnet:~$ sudo nano /etc/netplan/99_config.yaml
(copy-paste from the example in the top right corner, then save and quit)
devnet-adm@ubuntu-devnet:~$ sudo chmod 600 /etc/netplan/99_config.yaml
devnet-adm@ubuntu-devnet:~$ ls -l /etc/netplan/
total 8
-rw------ 1 root root 593 Sep 04 13:35 50-cloud-init-yaml
-rw------ 1 root root 85  Sep 08 15:46 99_config.yaml
(... cut for brevity ...)
devnet-adm@ubuntu-devnet:~$ sudo rm /etc/netplan/50-cloud-init.yaml
devnet-adm@ubuntu-devnet:~$ sudo netplan apply
```

!!! In YAML, indentation is REALLY important !!!

99_config.yaml

```
network:
  version: 2
  ethernets:
    enp0s3:
      dhcp4: true
#      dhcp4: false
#      addresses:
#      - 192.168.10.{YOUR_POD_IP}/24
#      routes:
#      - to: default
#        via: 192.168.10.1
#      nameservers:
#        addresses: [1.1.1.1,8.8.8.8]
```

(example with DHCP)

```
network:
  version: 2
  ethernets:
    enp0s3:
      dhcp4: true
#      dhcp4: false
#      addresses:
#      - 192.168.10.7/24
#      routes:
#      - to: default
#        via: 192.168.10.1
#      nameservers:
#        addresses: [1.1.1.1,8.8.8.8]
```

For simplicity, I usually just comment out the parts that I don't use.

- For DHCP I comment out the lines related to static
- For static config I comment the line "dhcp4: true" and uncomment the rest

(example with static IP)

```
network:
  version: 2
  ethernets:
    enp0s3:
#      dhcp4: true
      dhcp4: false
      addresses:
      - 192.168.10.7/24
      routes:
      - to: default
        via: 192.168.10.1
      nameservers:
        addresses: [1.1.1.1,8.8.8.8]
```

Create an SSH key -> Import to Ubuntu keystore

- Start by opening Powershell (not CMD for this task)
- 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"
```

```
The authenticity of host '192.168.10.7 (192.168.10.7)' can't be established
ED25519 key fingerprint is SHA256:V+TRZ8hkGRQh+kcu1OgUp+roD4dsO0ulip5HAcoY96s.
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:\>
```

```
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
```

Change to your server's IP

- 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:~$ 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:~$
```

- 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:~$ git pull origin main
```

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
  - YANG-Suite to explore IOS-XE YANG models
  - Telegraf, InfluxDB and Grafana to visualize Telemetry data from IOS-XE
- 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:~$
```

- The hello-world container is a small container, that just helps you check if your Docker installation is working

```
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
```

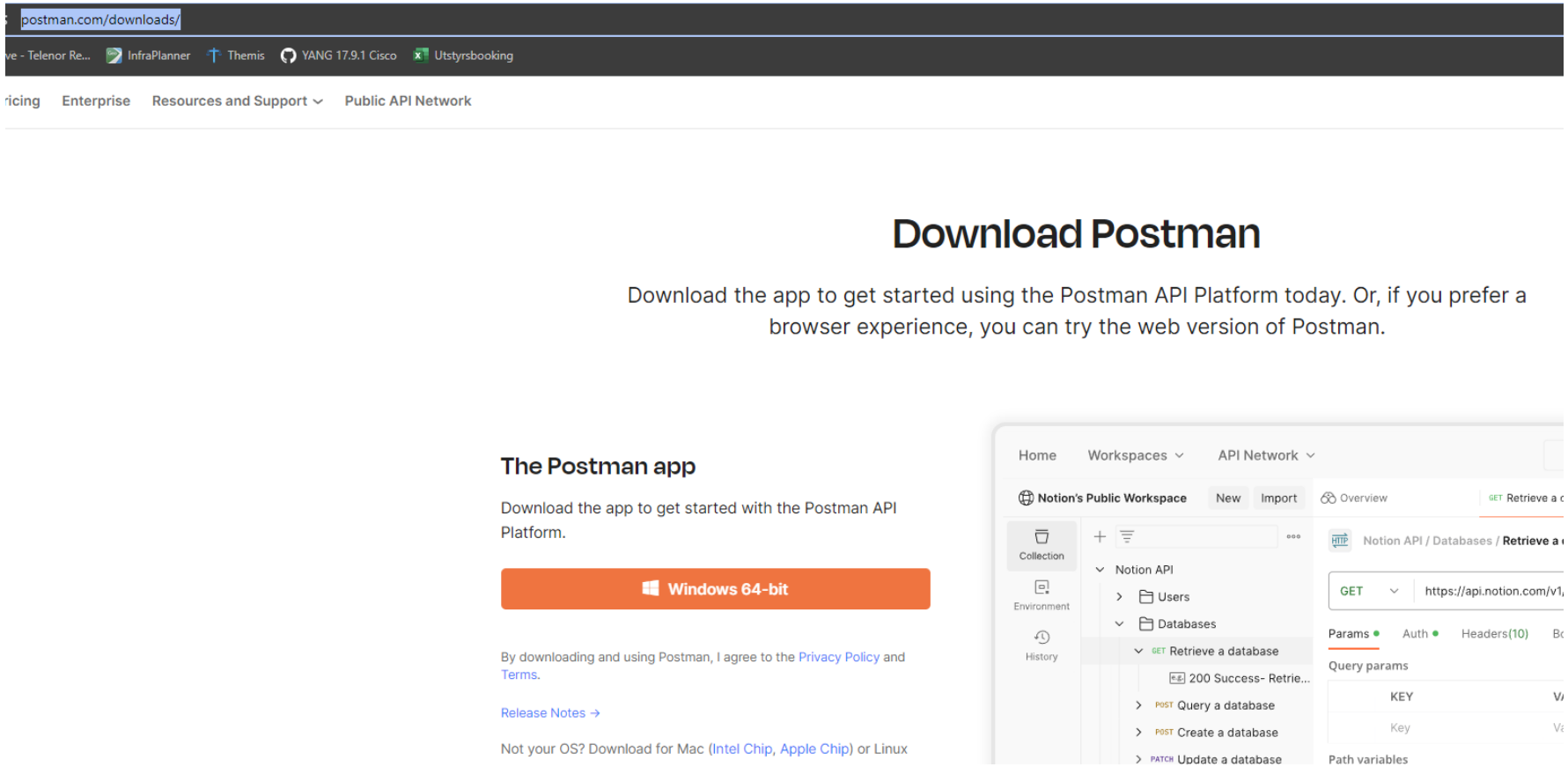
- Alpine is a very slim Linux distro. As an empty container, it requires about 8MB disk space. A lot of pre-built containers (like we will use later) are based on the alpine image.
- We will first pull Alpine, then run it and connect to its terminal using the "-it" option

```
/ # cat /etc/os-release
NAME="Alpine Linux"
ID=alpine
VERSION_ID=3.20.1
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
```

- 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)

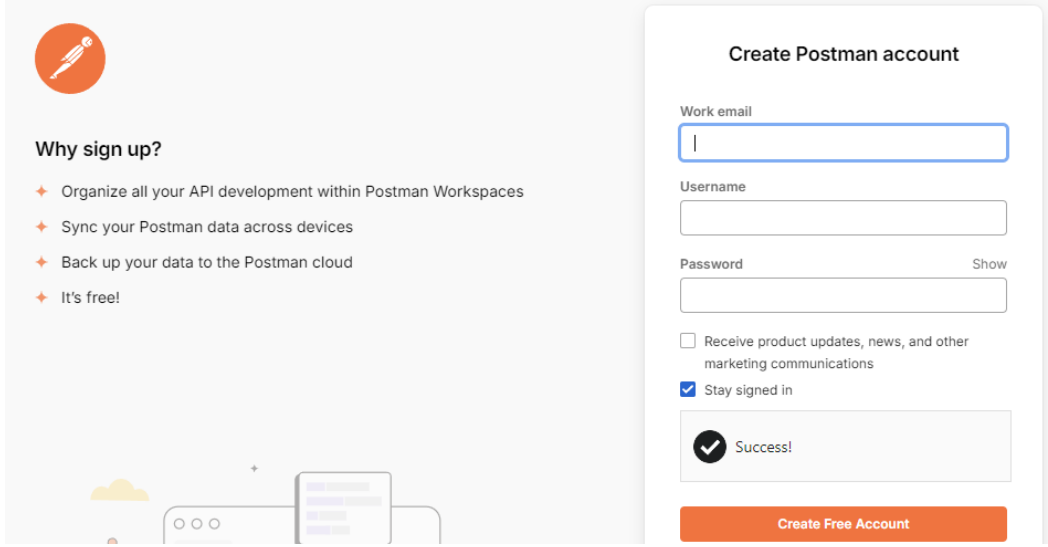
Pre-lab task #4: Install Postman on your laptop

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

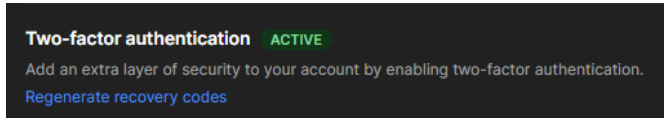


Create a Postman account

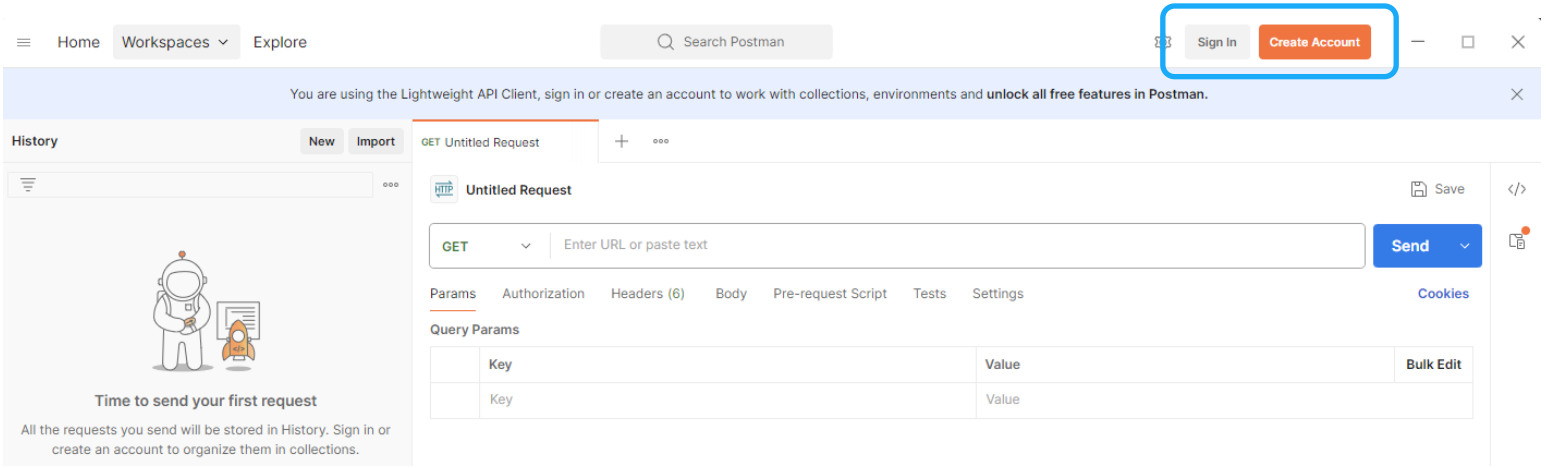
- Create a Postman account



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



- Log in to the Postman app using your account



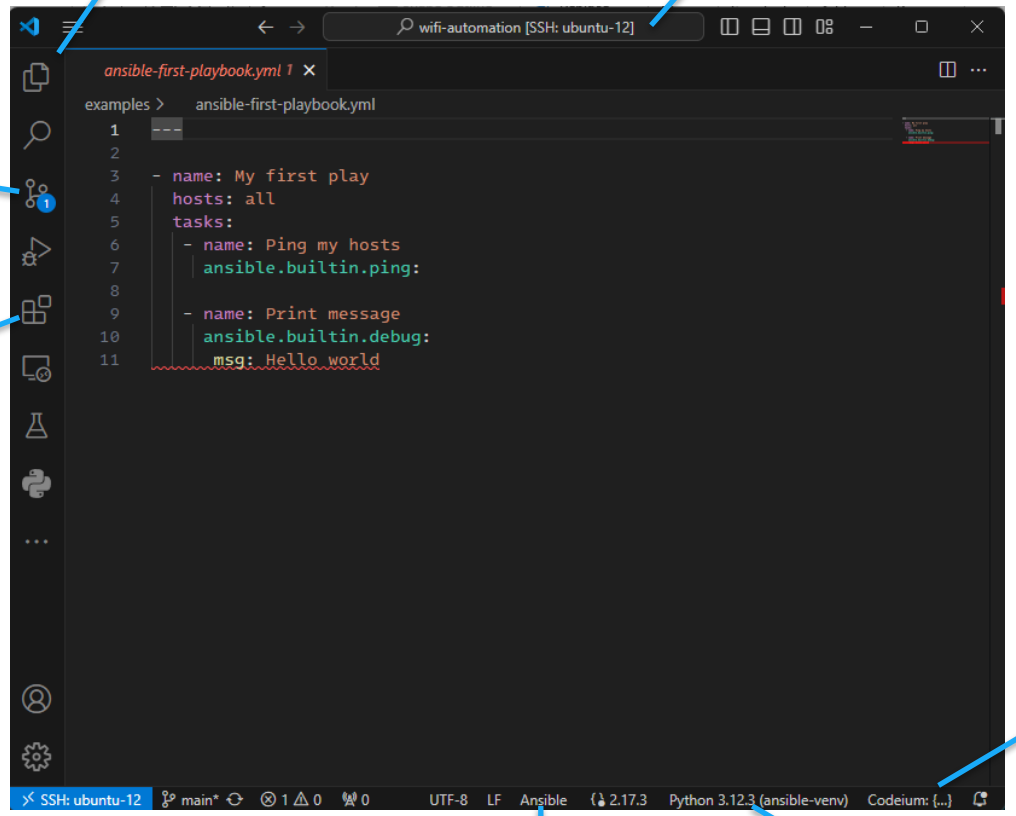
Pre-lab task #5: Install VS Code on your laptop

- 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)

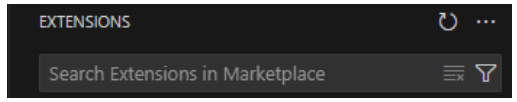
Command palette



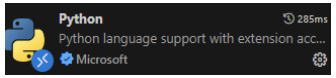
Source control (Git):

- 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.

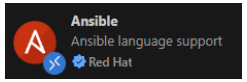
Extensions:



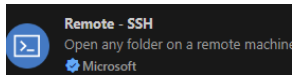
- Python



- Ansible



- Remote - SSH



Some active extension

(Codeium, GitHub Copliot, etc)

Python environment

Language (Python/Ansible/etc)

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

- In this task we will install Cisco 9800-CL on VirtualBox
- This task is marked as optional, but:
- You may do some (most) of the C9800 tasks in the deep dive using a shared WLC instead of having your own
- This might be the case if you have a weak laptop, are using MacOS, or are primarily interested in the non-Cisco stuff from the lab
- !!! 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 17.15.1 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

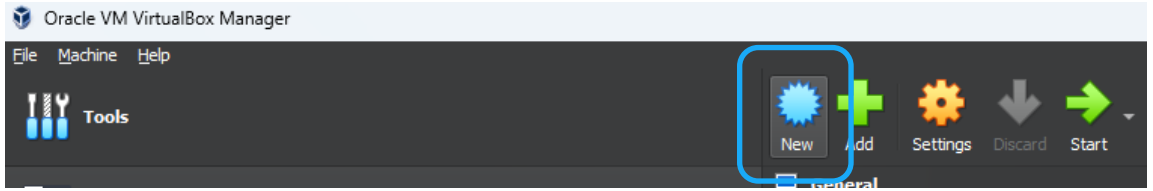
Cisco Catalyst 9800 Wireless Controller for Cloud - Hyper-V / ESXi / KVM C9800-CL-universalk9.17.15.01.iso	13-Aug-2024	1530.35 MB

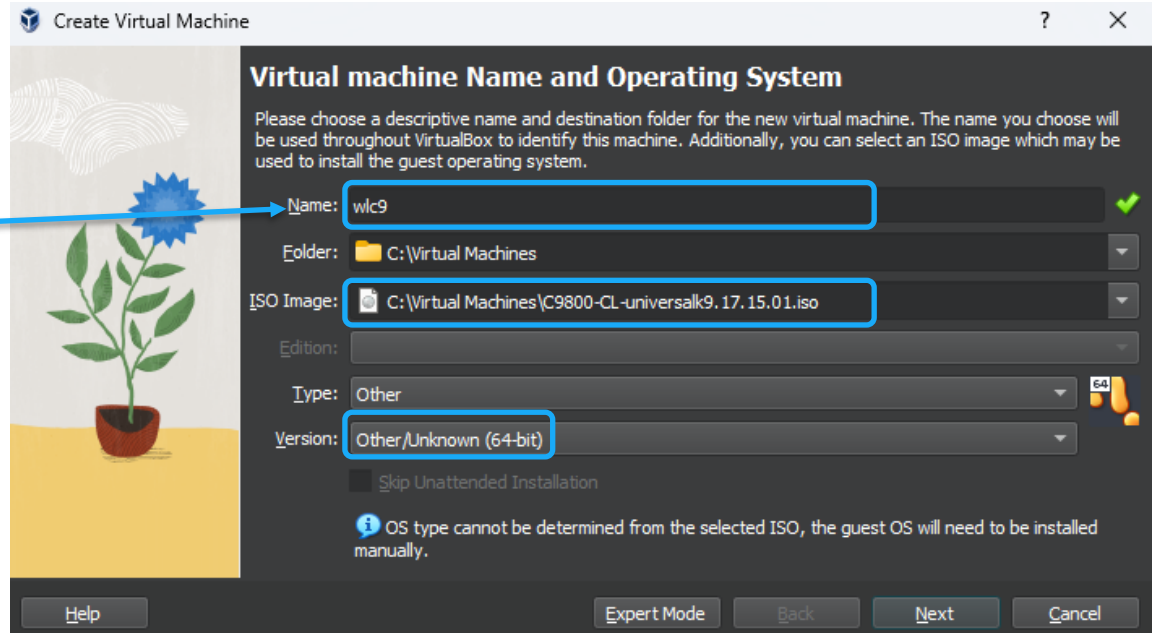
Table 1. Supported Profile Configurations

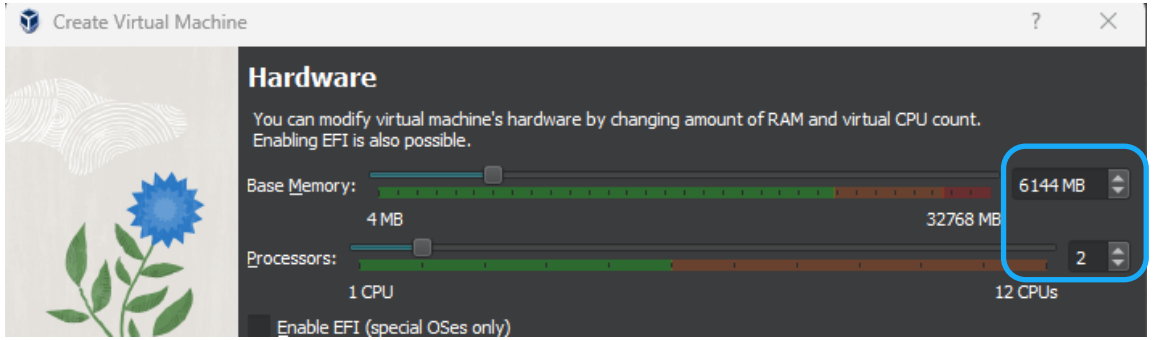
Profiles	CPUs	RAM	APs	Clients
Ultra-Low	2 vCPUs	6 GB	100	1,000
Small	4 vCPUs	8 GB	1000	10,000
Medium	6 vCPUs	16 GB	3000	32,0000
Large	10 vCPUs	32 GB	6000	64,0000

Create VM

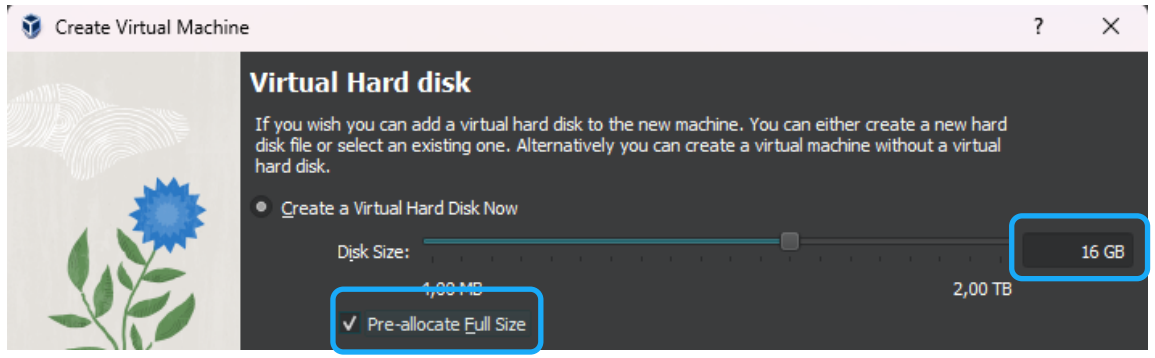
Change the name to reflect your pod number. E.g.
wlc11
wlc12
wlc13
etc

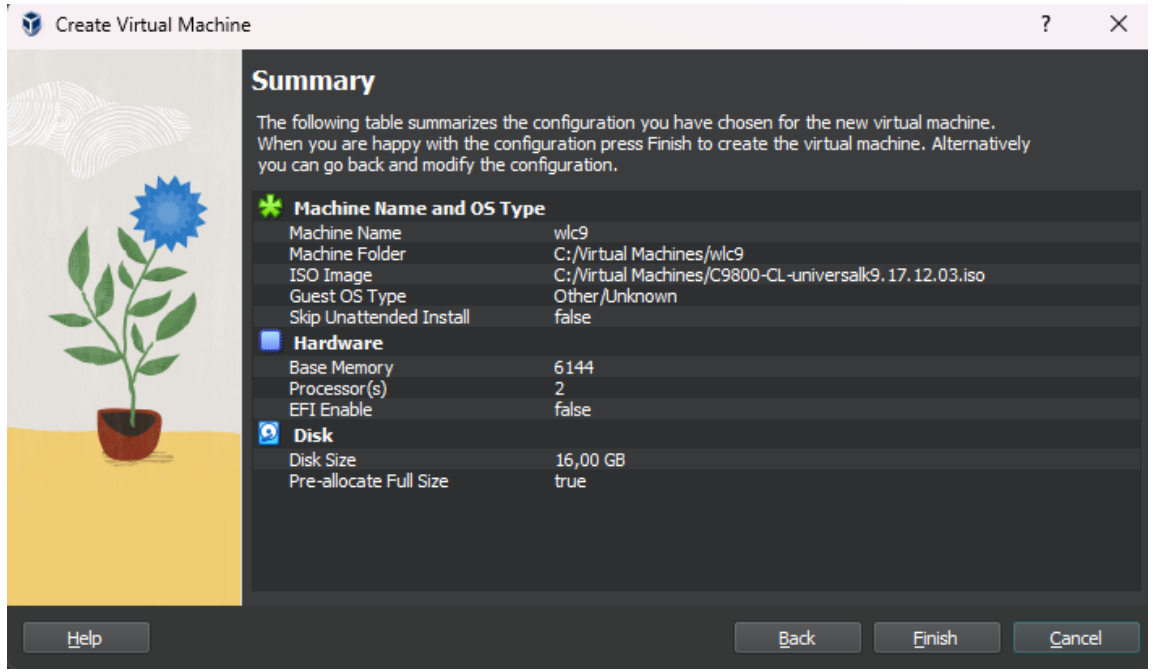






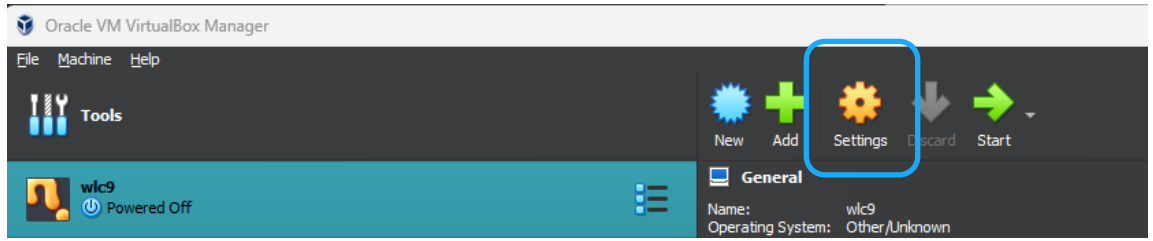
Create VM

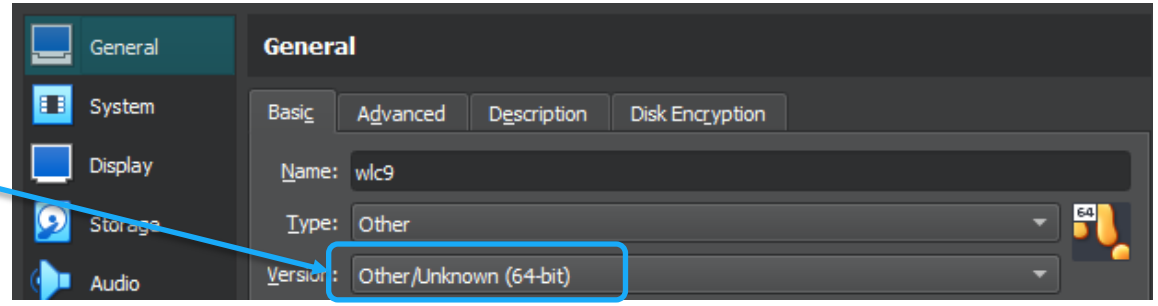

Create Virtual Machine
Virtual Hard disk
If you wish you can add a virtual hard disk to the new machine. You can either create a new hard disk file or select an existing one. Alternatively you can create a virtual machine without a virtual hard disk.
Create a Virtual Hard Disk Now
Disk Size:
16 GB
2,00 TB
Pre-allocate Full Size

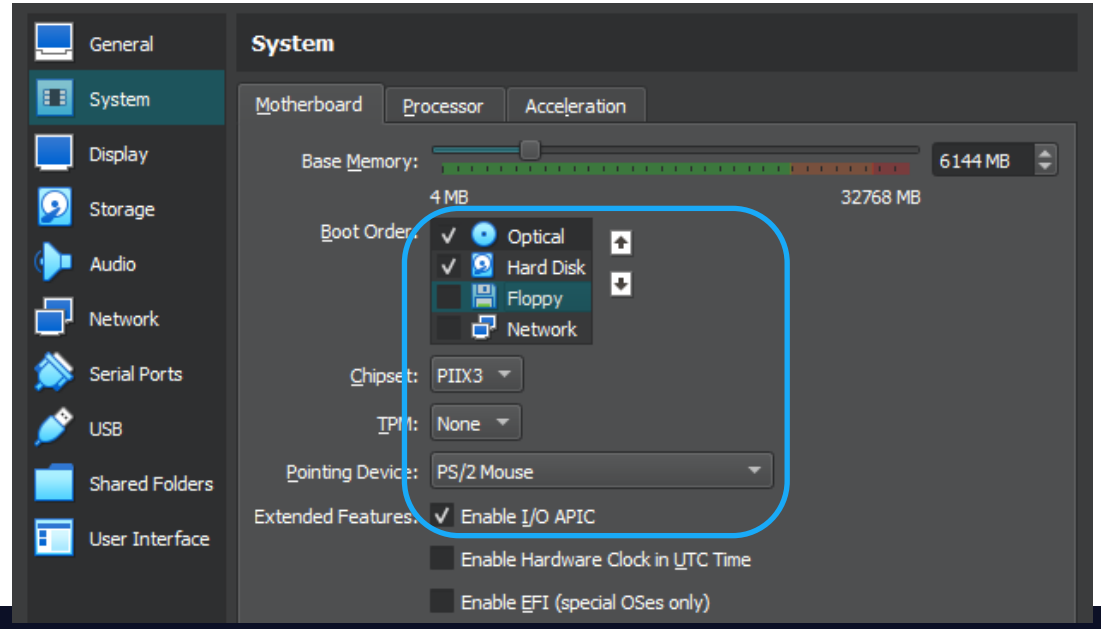

Create Virtual Machine
Summary
The following table summarizes the configuration you have chosen for the new virtual machine. When you are happy with the configuration press Finish to create the virtual machine. Alternatively you can go back and modify the configuration.
Machine Name and OS Type
Machine Name wlc9
Machine Folder C:/Virtual Machines/wlc9
ISO Image C:/Virtual Machines/C9800-CL-universalk9.17.12.03.iso
Guest OS Type Other/Unknown
Skip Unattended Install false
Hardware
Base Memory 6144
Processor(s) 2
EFI Enable false
Disk
Disk Size 16,00 GB
Pre-allocate Full Size true
Help
Back
Finish
Cancel

Modify VM

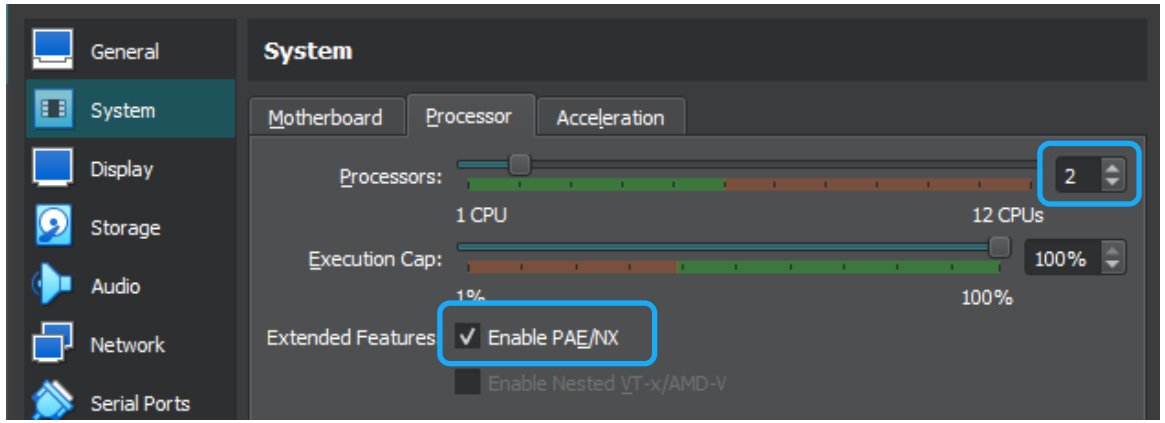
Important

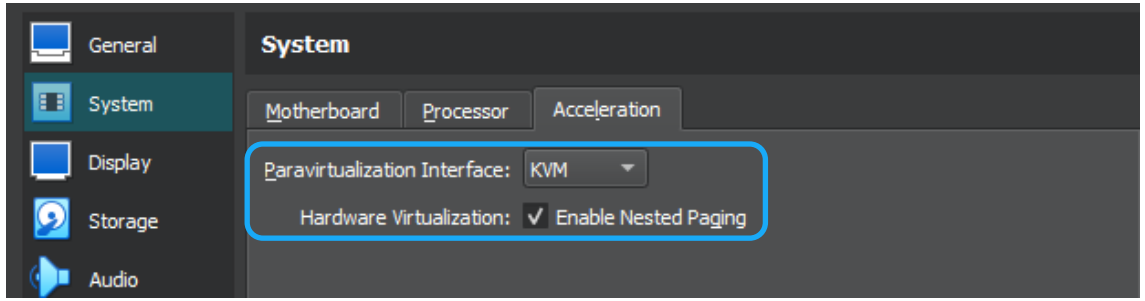




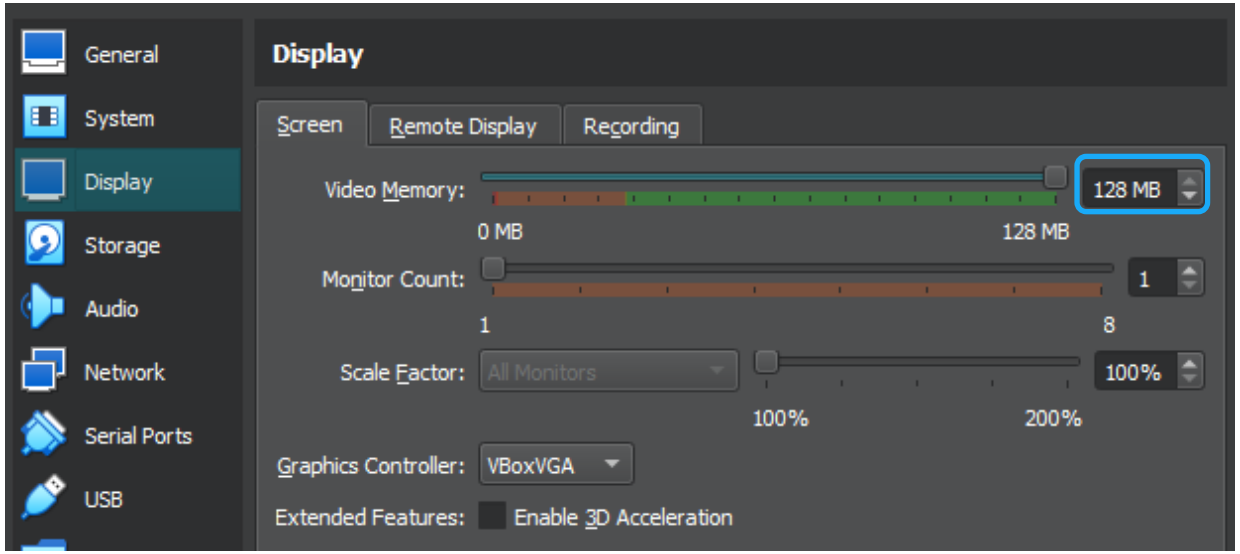


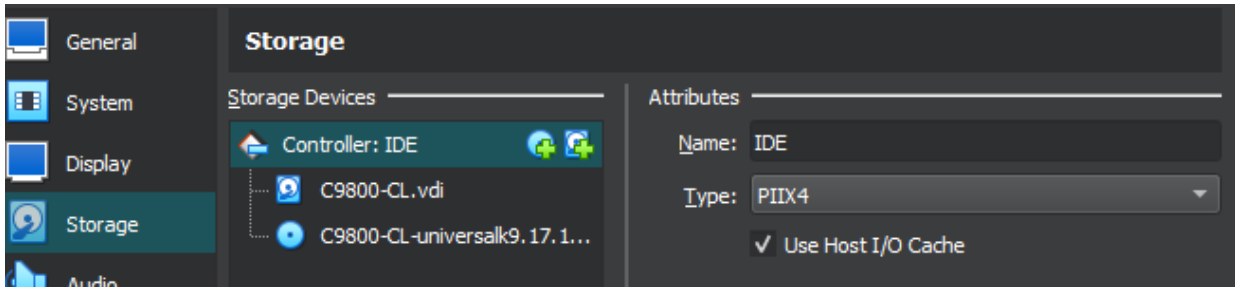
Modify VM


General
System
Display
Storage
Audio
Network
Serial Ports
System
Motherboard
Processor
Acceleration
Processors:
2
1 CPU
12 CPUs
Execution Cap:
100%
1%
100%
Extended Features
Enable PAE/NX
Enable Nested VT-x/AMD-V

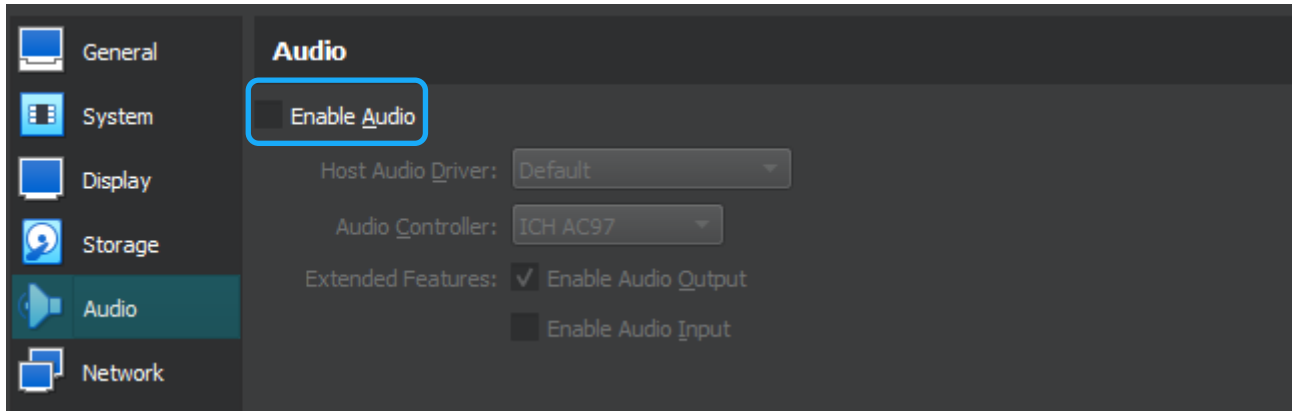

General
System
Display
Storage
Audio
System
Motherboard
Processor
Acceleration
Paravirtualization Interface:
KVM
Hardware Virtualization:
Enable Nested Paging

Modify VM

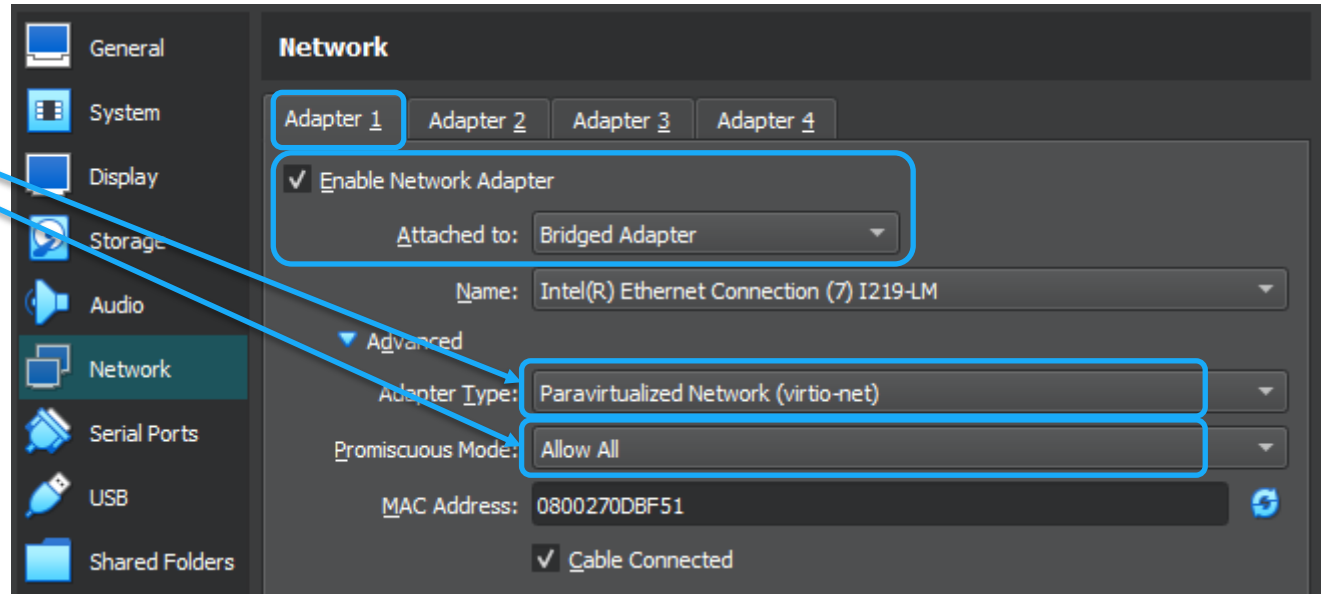

General
System
Display
Storage
Audio
Network
Serial Ports
USB
Display
Screen
Remote Display
Recording
Video Memory:
128 MB
0 MB
128 MB
Monitor Count:
1
1
8
Scale Factor:
All Monitors
100%
100%
200%
Graphics Controller:
VBoxVGA
Extended Features:
Enable 3D Acceleration


General
System
Display
Storage
Storage
Storage Devices
Controller: IDE
C9800-CL.vdi
C9800-CL-universalk9.17.1...
Attributes
Name:
IDE
Type:
PIIX4
Use Host I/O Cache

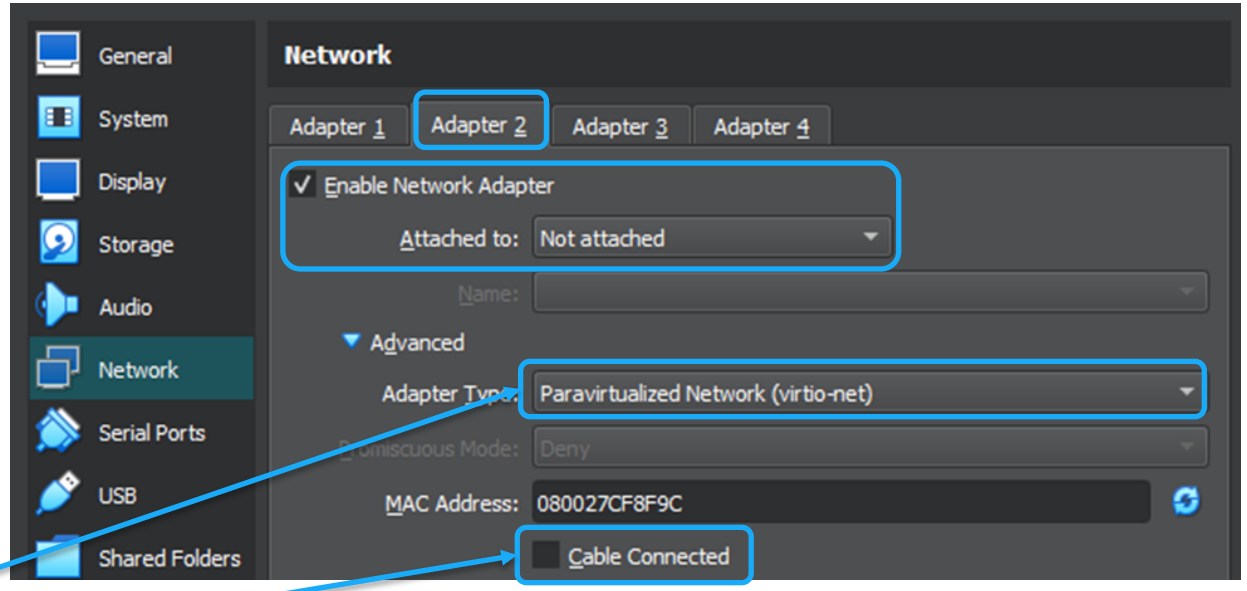
Modify VM



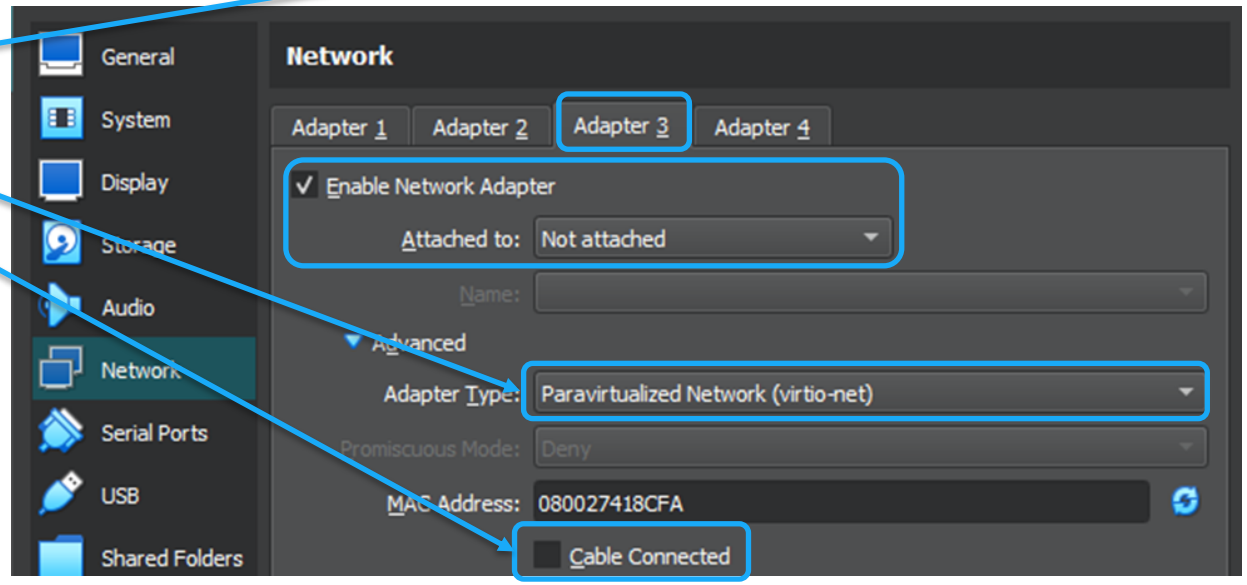
Important



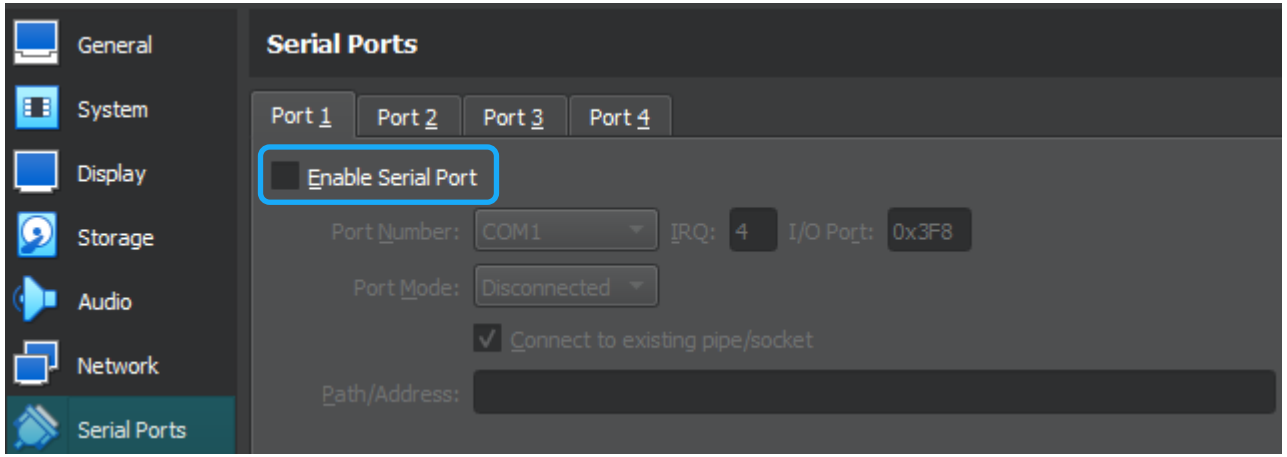
Modify VM

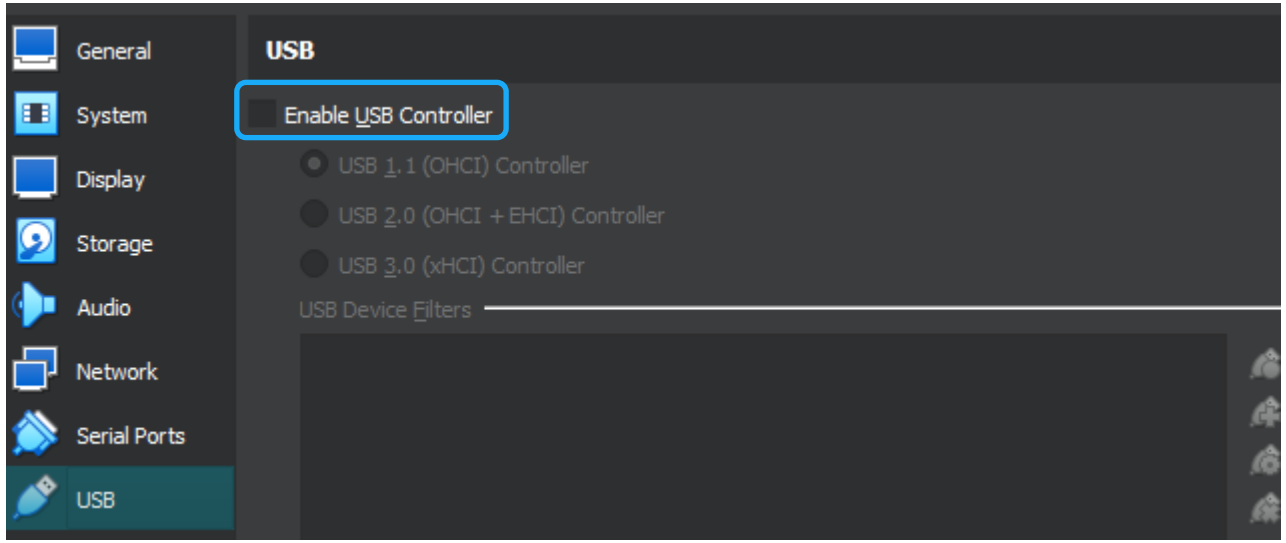


Important

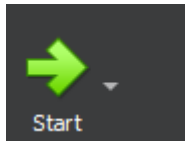


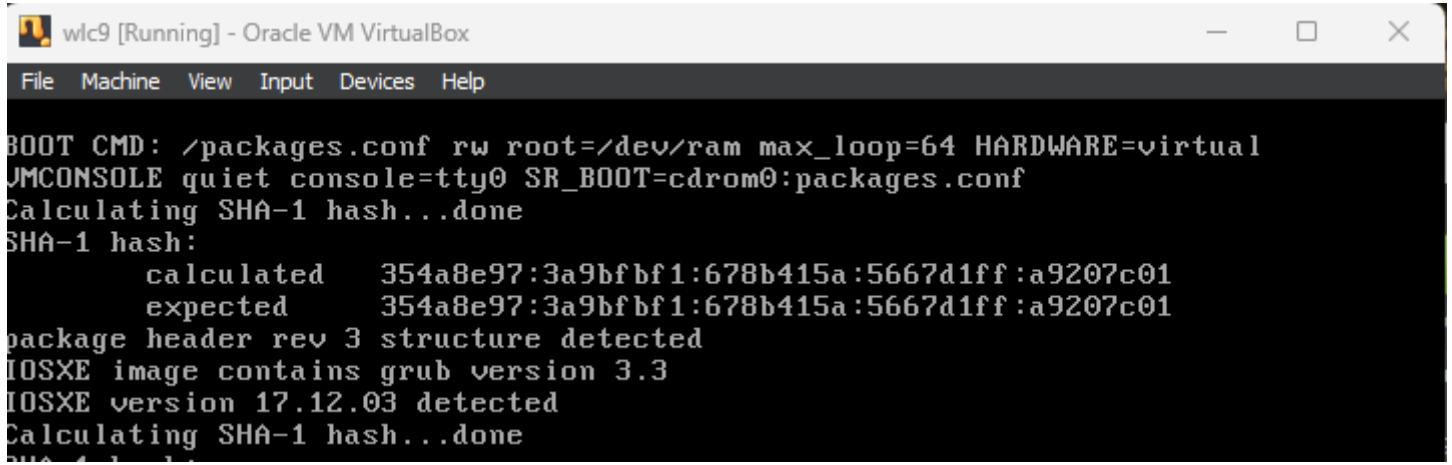
Modify VM


General
System
Display
Storage
Audio
Network
Serial Ports
Serial Ports
Port 1
Port 2
Port 3
Port 4
Enable Serial Port
Port Number:
COM1
IRQ:
4
I/O Port:
0x3F8
Port Mode:
Disconnected
Connect to existing pipe/socket
Path/Address:

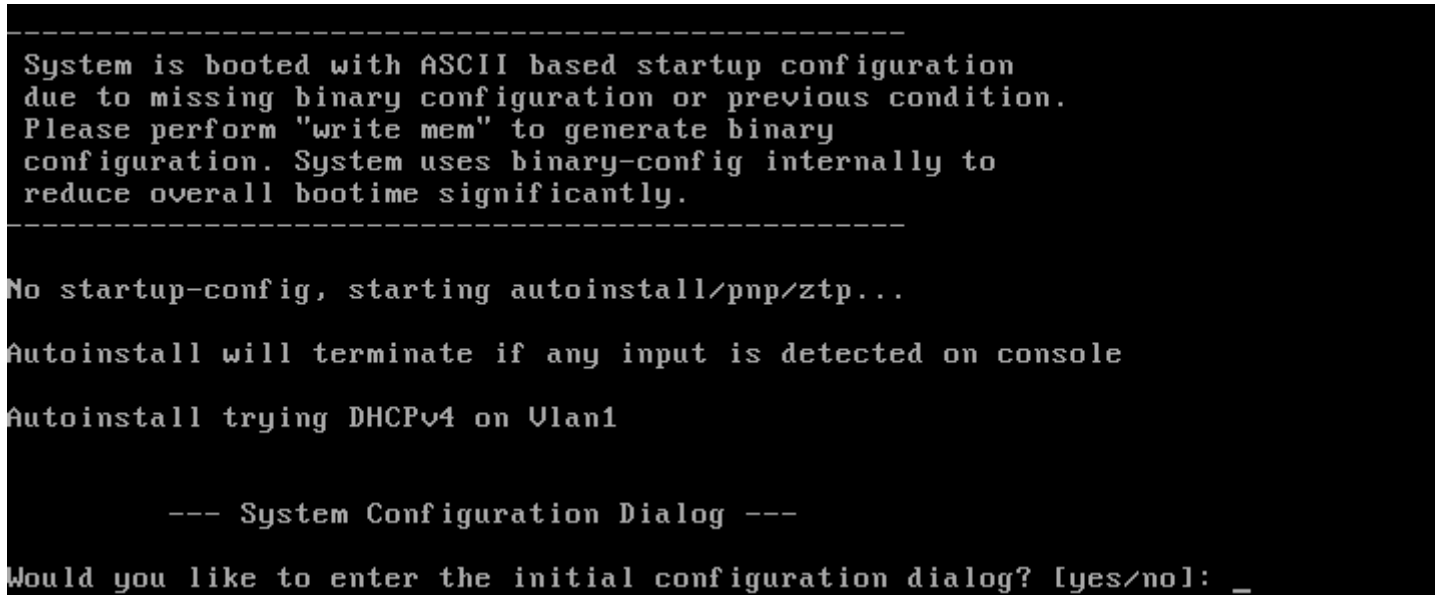

General
System
Display
Storage
Audio
Network
Serial Ports
USB
USB
Enable USB Controller
USB 1.1 (OHCI) Controller
USB 2.0 (OHCI + EHCI) Controller
USB 3.0 (xHCI) Controller
USB Device Filters

Run VM





Take some minutes to boot...
It looks like this when first boots are completed and ready for config:



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)

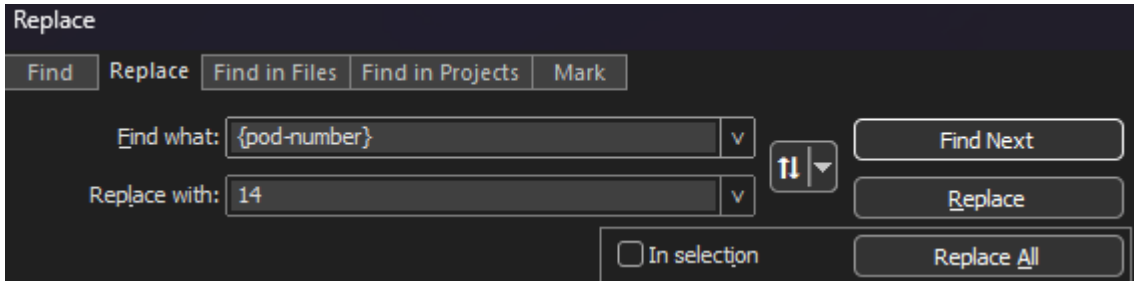
```
 The enable secret is a password used to protect
 access to privileged EXEC and configuration modes.
 This password, after entered, becomes encrypted in
 the configuration.
 ------------------------------------------------
 secret should be of minimum 10 characters and maximum 32 characters with
 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/c2u6YO992/VkxN..O4UDSQVV0HowSM1M1Qo
!
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

- Copy the config script
- Replace the text {pod-number} with your pod number



- It should be 5 occurences
  - Hostname: `hostname wlc{pod-number}`
  - IP: `ip address 192.168.10.{pod-number}`
  - SSID name: `wlan lab-network 17 lab-network{pod-number}`
  - Mobility group: `wireless mobility group name automationlab{pod-number}`
  - RF-group: `wireless rf-network automationlab{pod-number}`
- **The first section (40ish lines) must be manually entered, then connect using SSH**
- The second section can be copy-pasted after connecting to WLC with SSH
- You should change the country code from NO to your country, or to CZ for local compliance
- The last line should be pasted alone after the rest is finished

Copy this to Notepad++
Replace {pod-number} with number
Then paste in conf mode on 9800:

```
!!! Enter this manually to get SSH up and running !!!
enable
conf t
hostname wlc{pod-number}
ip domain name automationlab.local
crypto key generate rsa
4096
!
vlan 10
name management
!
interface Vlan 1
shutdown
!
interface vlan 10
description management
ip address 192.168.10.{pod-number} 255.255.255.0
no shutdown
!
interface GigabitEthernet1
shutdown
description Uplink
switchport trunk allowed vlan 30,11
switchport mode trunk
switchport trunk native vlan 10
no shutdown
!
ip default-gateway 192.168.10.1
ip route 0.0.0.0 0.0.0.0 192.168.10.1
aaa new-model
username devnet-adm privilege 15 algorithm-type sha256 secret ChangeMe2024!
enable algorithm-type sha256 secret ChangeMe2024!
aaa authentication login default local
aaa authentication enable default enable
line vty 0 10
privilege level 15
logging synchronous
transport input ssh
!

!!! Log in with SSH, then you can copy-paste the rest !!!
conf t
!
interface GigabitEthernet2
description Out-of-Band
shutdown
!
interface GigabitEthernet3
description HA-sync
shutdown
!
vlan 11
name lab-network
!
do terminal length 40
ip dhcp bootp ignore
no service pad
no pae
service tcp-keepalives-in
service tcp-keepalives-out
ntp server 40.123.168.38
service timestamps debug datetime msec localtime
service timestamps log datetime msec localtime
service password-encryption
aaa authentication login
aaa authorization exec default local
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 40960
configuration mode exclusive auto
errdisable recovery cause all
errdisable recovery interval 600
ip name-server 1.1.1.1 8.8.8.8
ip ssh time-out 15
!
line con 0
 exec-timeout 5 0
 logging synchronous
 stopbits 1
line vty 0 10
 exec-timeout 60 0
 privilege level 15
 logging synchronous
 transport input ssh

wireless mobility group name automationlab{pod-number}

wireless mobility dscp 0

wireless rf-network automationlab{pod-number}
wireless client max-user-login 3
ap dot11 5ghz shutdown

ap dot11 2 shutdown

wireless country NO

no ap dot11 2 shutdown
no ap dot11 5 shutdown
!
wireless management interface vlan10
!
wireless profile flex lab-flex-profile
 native-vlan-id 10
 vlan-name 10
  vlan-id 10
 vlan-name 11
  vlan-id 11
!
ap profile lab-ap-profile
 mgmtuser username devnet-adm password 0 ChangeMe2024! secret 0 ChangeMe2024!
 ntp
!
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{pod-number}
 radio policy dot11 5ghz
 security ft over-the-ds
 wmm require
 security wpa psk set-key ascii 0 ChangeMe2024!
 no security wpa akm dot1x
 security wpa akm psk
 no shutdown
!
wireless profile policy lab-policy
 no central authentication
 no central dhcp
 no central switching
 dhcp-tlv-caching
 http-tlv-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-regex .
 tag policy lab-policy-tag
 tag rf default-rf-tag
 tag site lab-site-tag
ap filter priority 100 filter-name lab-filter-rule
!
password encryption aes
key config-key password-encrypt
PasswordForDecryptOrAutomationLab!
PasswordForDecryptOrAutomationLab!
!
!
exit
!
!
! The last line should be pasted alone, after the rest is finished
wireless config vwlc-ssc key-size 2048 signature-algo sha256 password 0 CertPasswordAutomationLab
```

Prepare WLC for API connections

- Enable NETCONF, RESTCONF, and create the users

```
username restconf-adm privilege 15 secret restconf-pass
username netconf-adm privilege 15 secret netconf-pass
netconf-yang
restconf
ip http secure-server
aaa authorization exec default local
```

- Enable Role-Based Access Control and create some RO users

```
username restconf-read privilege 1 secret restconf-read
username netconf-read privilege 1 secret netconf-read
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". Try later 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
  - VirtualBox
    - Virtual Mahine 1: 9800-CL
    - Virtual Machine 2: Ubuntu Server w/Docker
  - Postman
  - VS Code w/some extensions