

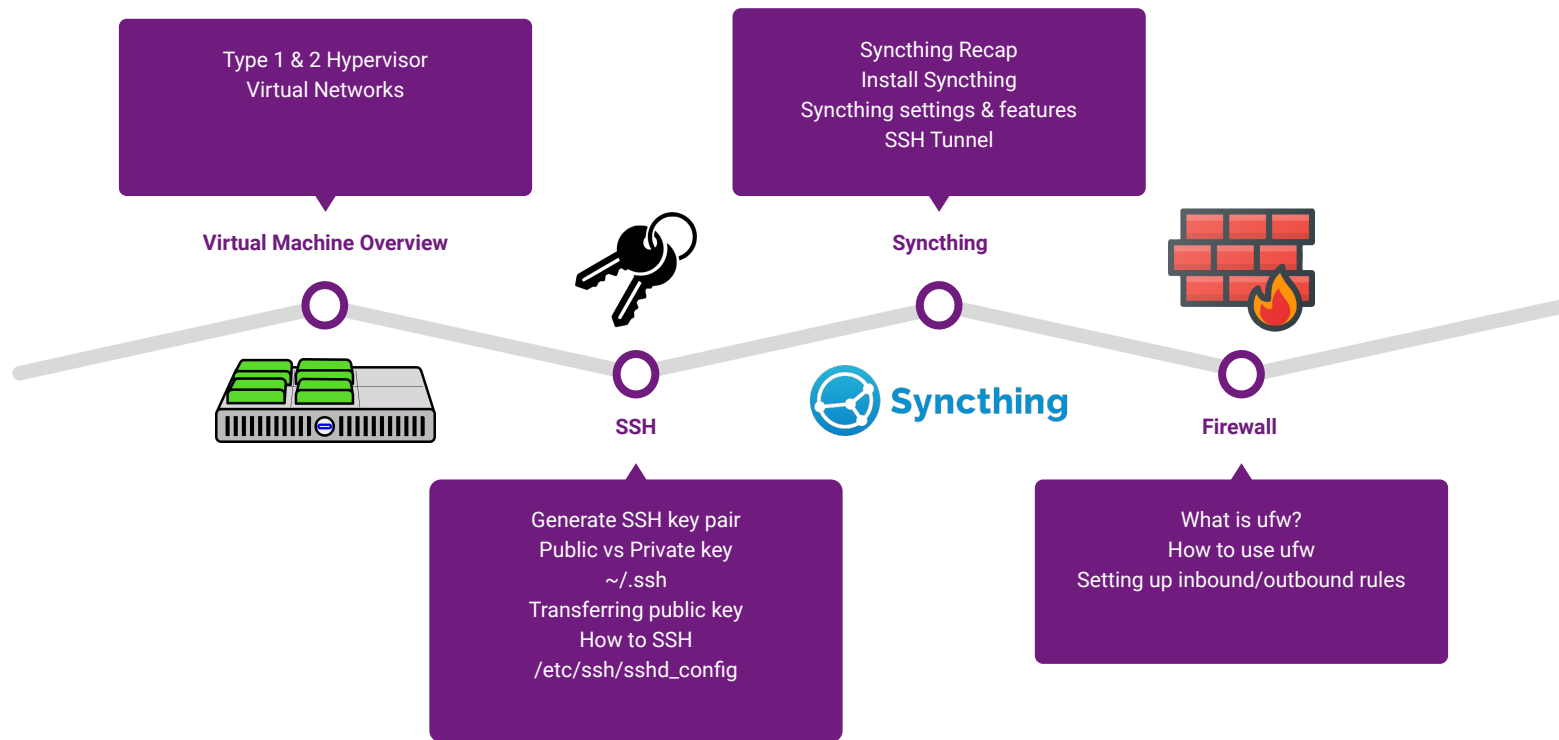


CHECK OUT: [rsecke.github.io](https://rsecke.github.io)

# Setting Up A Syncthing Server

Robinson Tran

# Overview



# Recap

**What is a server?** A server is a computer that provides a service to another computer

**What is the Cloud?** High demand, high availability of computing resources that provides users with a service

**What is Syncthing?** Software that allows users to have file synchronization using a peer to peer network (no cloud)

# Before We Get Into Syncthing...

- We need to learn a few things about how to set up, configure, and secure the server
  - Understand the infrastructure you're working with before you create a server on your network
  - Understand SSH in order to connect to the server, and use it to get Syncthing to run as your central file server
- Installing Syncthing is actually the easiest part. Making sure to set the server up correctly is just as important.



# Virtual Machine Review



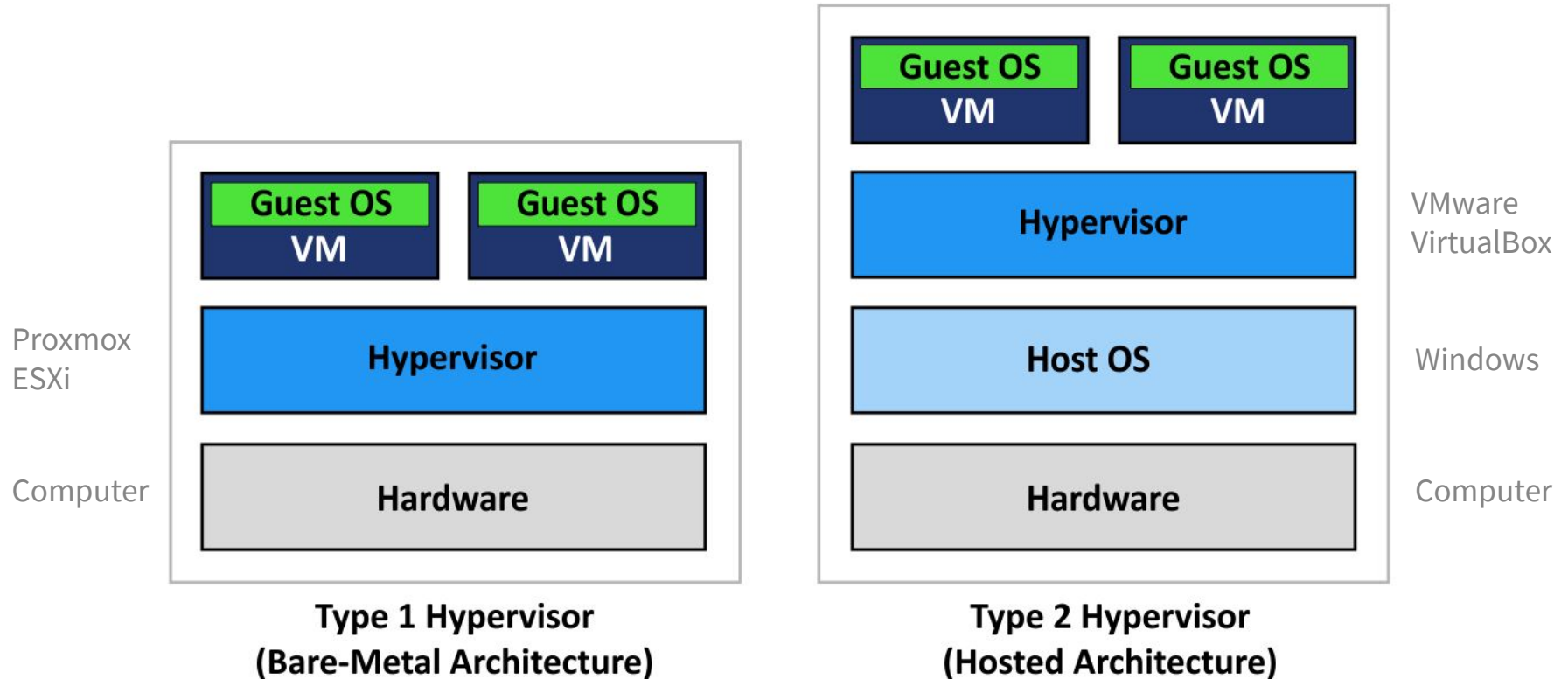
# Hypervisors

**Virtual machine:** a simulated environment (OS) running inside your computer

**Hypervisor:** software that works with your computer's resource pool to manage VMs

- **Type I:** Runs on “bare metal”, which means the only purpose of this computer is to manage virtual machines. (ESXi, Proxmox)
- **Type II:** Run on top of an operating system (VirtualBox, Workstation)

# Type 1 vs Type 2 Hypervisor





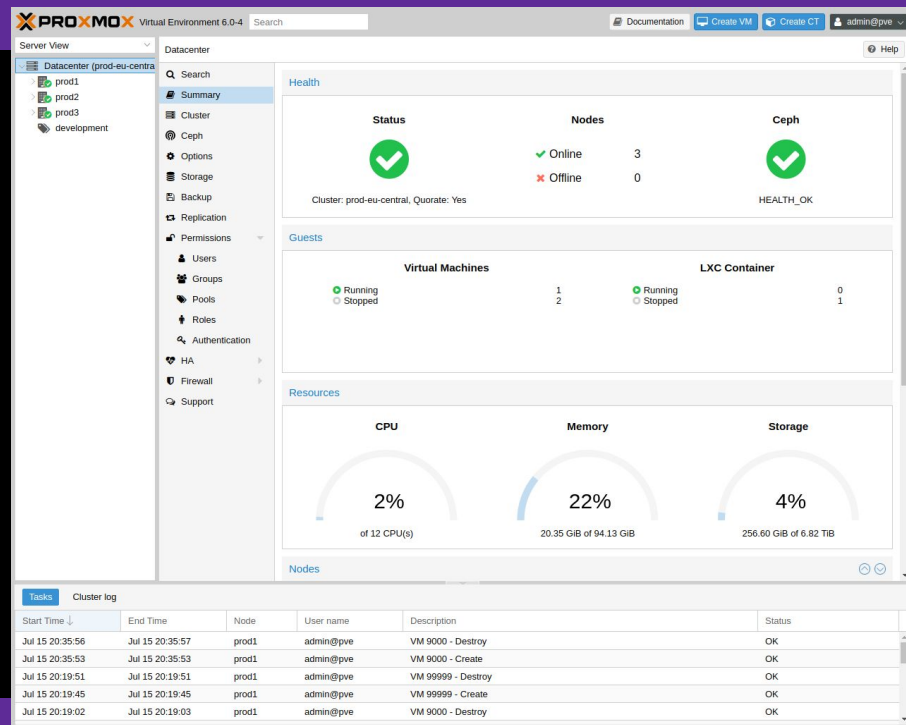
# Direct Access to Proxmox

## Proxmox Web GUI

Welcome to the Proxmox Virtual Environment. Please use your web browser to  
configure this server - connect to:

`https://192.168.200.82:8006/`

`pve login: _`



# Virtual Networks

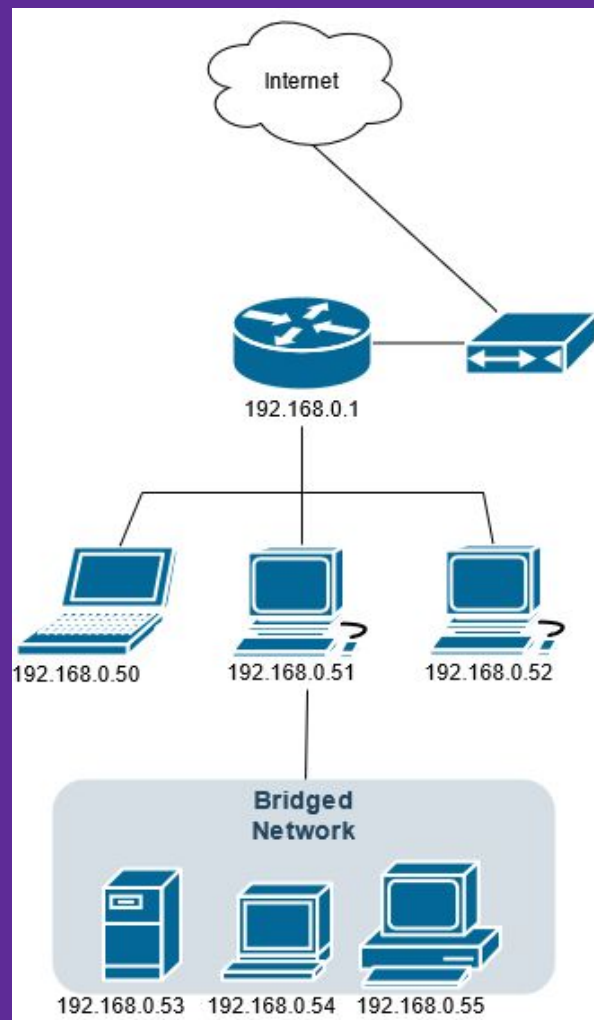
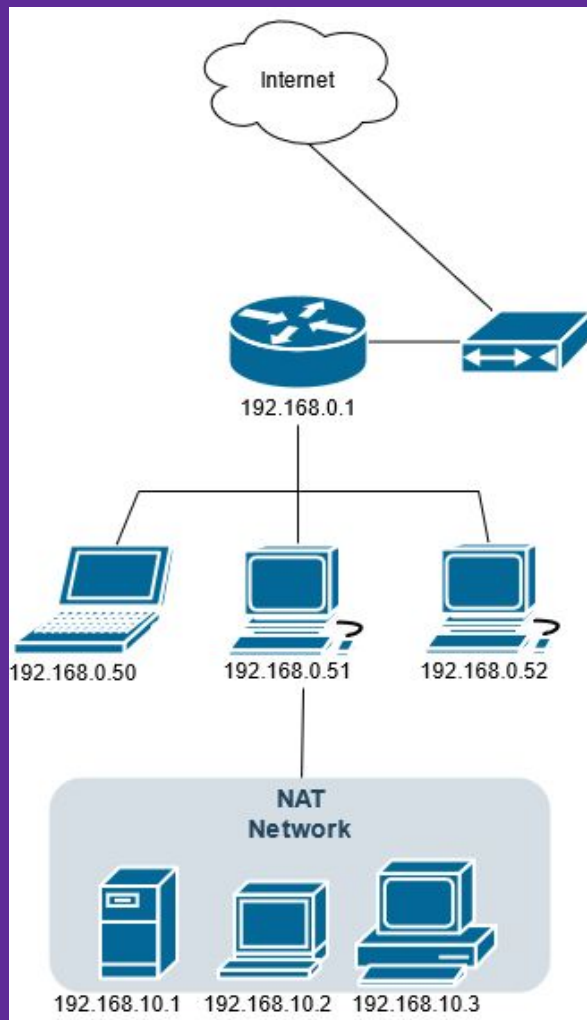
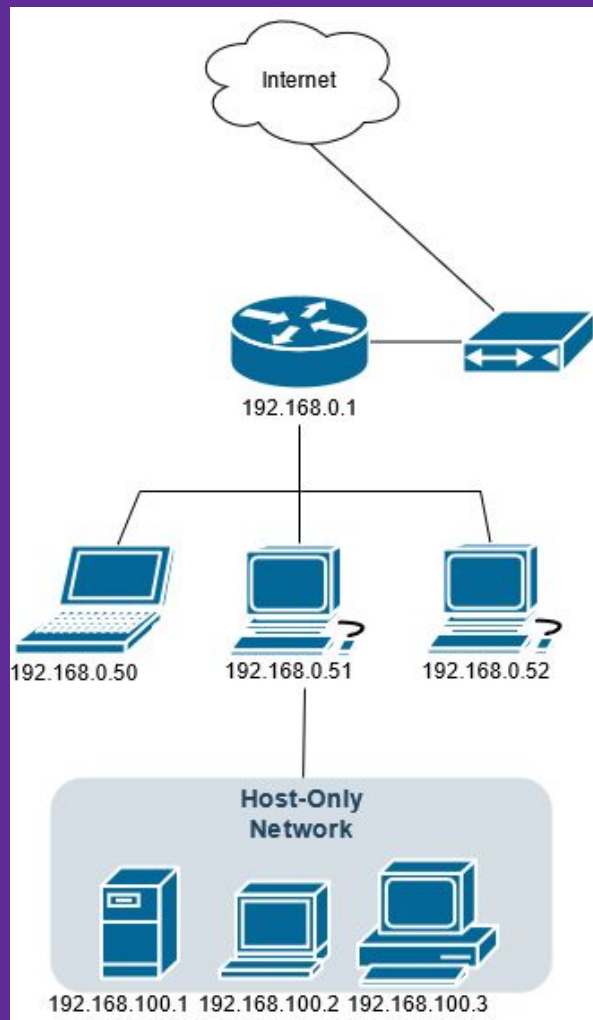
There are 3 main networking configurations on virtual machines

- **Bridged network:** The VM will act as if it is on your home network
- **NAT network:** The VM gets masked into its own private network
- **Host-only network:** The VM will be in an isolated virtual network

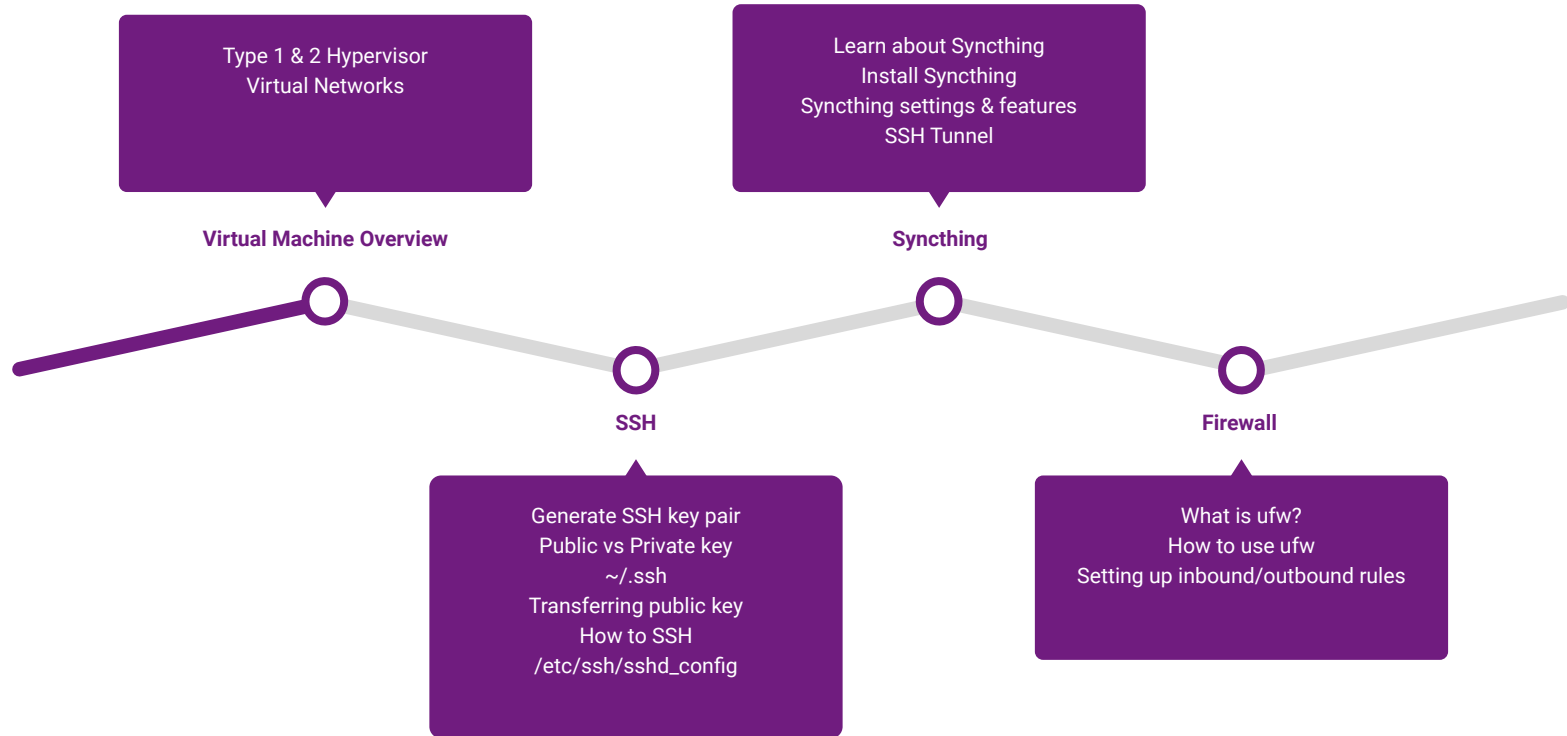
Bridged > On your LAN

NAT > Can communicate with those also on NAT (your local machine can't see this)

Host-only > Contained in an isolated environment



# SSH (Secure SHell)



# SSH (Secure SHell)

- SSH is a way for to securely connect to a remote computer
  - Encrypted connection
- Works over port 22
- Replaces the insecure protocols, telnet
  - SSH also has other communication protocols: scp, sftp
- Very important to learn because it is how you will be communicating with servers/systems
- **sudo apt install ssh** if you dont have SSH installed

# Creating an SSH keypair

**ssh-keygen** **-t** <key type>

Diagram illustrating the components of the `ssh-keygen` command:

- `ssh-keygen`: Command
- `-t`: Flag
- `<key type>`: Input

**ssh-keygen**: command to create an SSH keypair

**-t**: flag to indicate what type of key to create

**Input**: dsa, ecdsa, ed25519, rsa

**ssh-keygen -t ed25519**

# SSH Keypair Location

/home/<username>/.ssh

What is the dot?

If it's hidden, how can we see it? **ls -la**

**ls -la**

Command    Flag

**ls**: command to show content in a directory

**-l**: lists content in a list format

**-a**: shows ALL contents (including hidden folders)

# What are SSH keys?

The **ssh-keygen** -t <key type> command creates 2 keys.

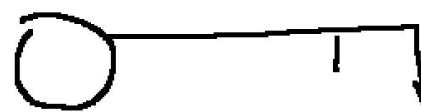
Asymmetric encryption uses two keys to encrypt and decrypt. One key is **public**, and the other is **private**.

- Public key encrypts something
- Private key decrypts something

DO NOT SHARE THE PRIVATE KEY WITH ANYONE

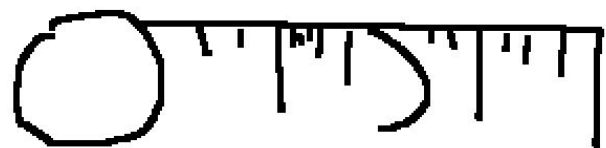


you vs the guy she told you not  
to worry about:



A hand-drawn diagram of a simple key. It consists of a circle on the left, representing the head, and a single horizontal line extending to the right, representing the blade. A small vertical tick mark is located on the blade.

you (rsa)



A hand-drawn diagram of a complex key. It consists of a circle on the left, representing the head, and a horizontal line extending to the right, representing the blade. The blade is marked with several vertical tick marks of varying lengths, indicating a more complex or multi-bit structure.

The Guy (ed 25519)

# Transferring the PUBLIC key

**ssh-copy-id** **-i** <path/to/PUBLIC key> <remote user>@<remote IP>

└──────────┬──┴──┬──────────────────────────┬────────────────────────────────────────┘  
Command Flag Input Input

**ssh-copy-id**: command to copy the PUBLIC key to a remote computer

**-i**: identifies the path to the PUBLIC key

**Input**: robinson@192.131.64.100

**ssh-copy-id -i** ~/.ssh/publickey robinson@192.131.64.100

# How to SSH

**ssh** <remote user>@<remote IP Address>

└──┬──────────────────────────────────────────────────────────────────────────┘  
Command Input

ssh: command to SSH into another computer

**-i**: specifies the public key used to SSH into another computer

**ssh** robinson@192.131.64.100

# Editing the SSH Configuration File

/etc/ssh/sshd\_config

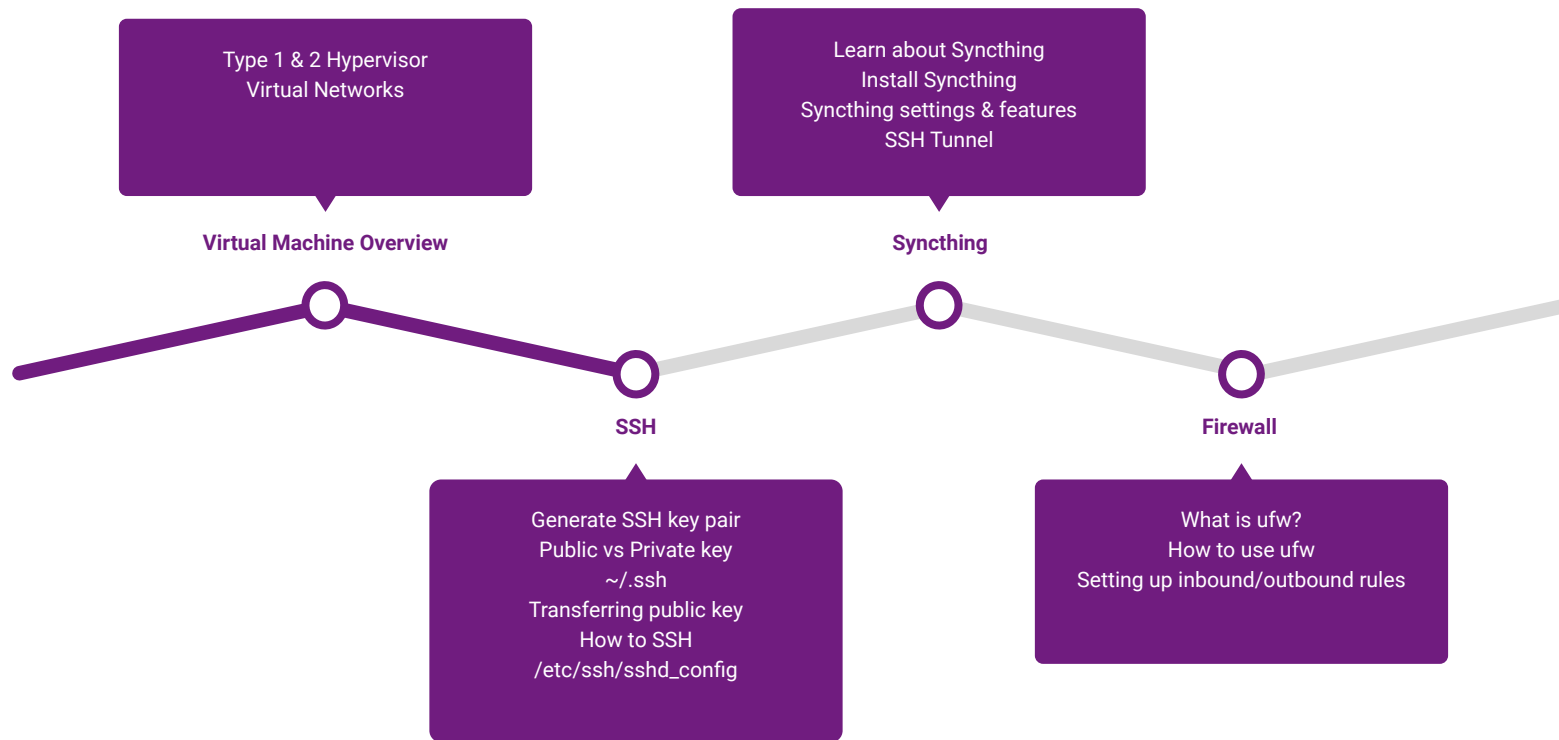
- `cp sshd_config sshd_config.bak`

What we're going to do:

- Disable password authentication
- Enable public key authentication
- Disable root login
- Limit # of times for a wrong password

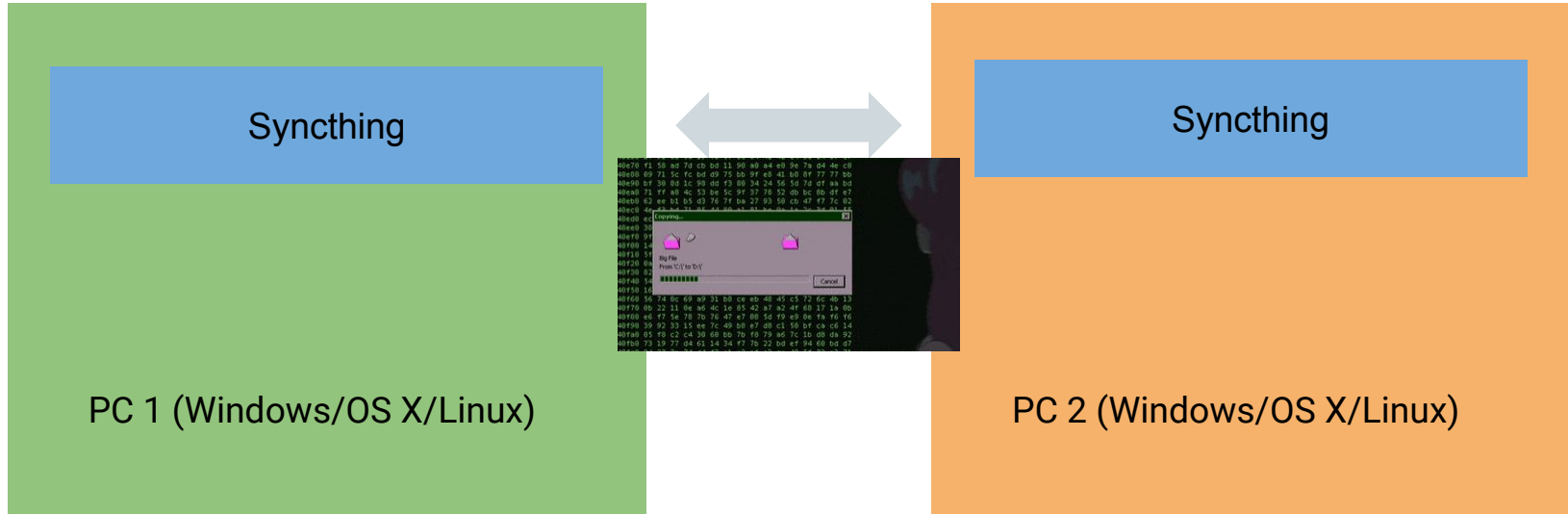
Benefits: easier to SSH, SSH provides better security, only those allowed can login

# Overview



# What is Syncthing?

Syncthing is software you can install onto your computer for file synchronization. It works by establishing a connection between 2+ computers.



# Installing/Starting Syncthing

## On Windows:

Install Syncthing Trayzor from <https://syncthing.net/downloads/>

## On Linux:

```
sudo apt update
```

```
sudo apt install syncthing
```

```
sudo adduser syncthing
```

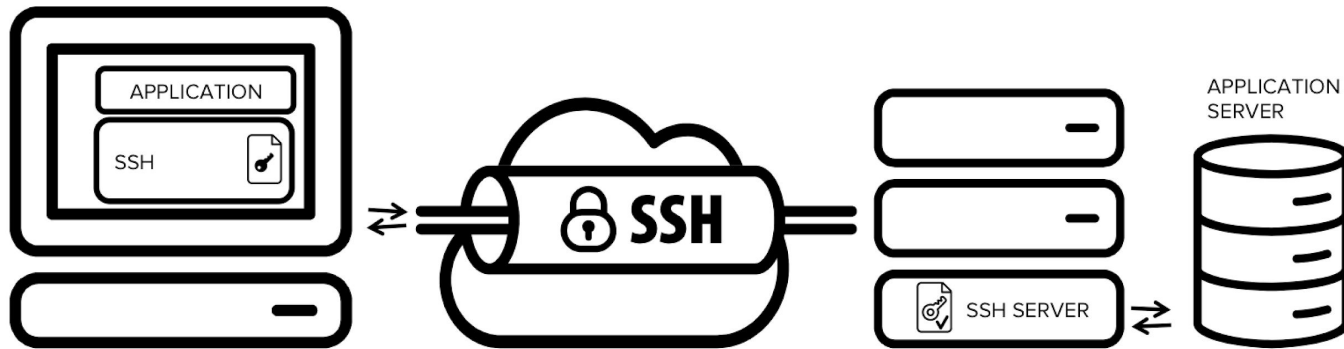
```
sudo systemctl enable syncthing@syncthing.service
```

```
sudo systemctl start syncthing@syncthing.service
```

# SSH Tunneling

SSH tunneling allows you to forward the traffic from one port to another over an SSH connection

- It is also called port forwarding
- The idea is you have a local computer listening on an open port. This port will be tunneling traffic from another remote port





# Local SSH Tunneling

**ssh** **-L** <listening port>:<remote IP address>:<remote port> <remote username>@<remote IP>

[ ] [ ] [ ] [ ]

Flag

Input

Input

Command

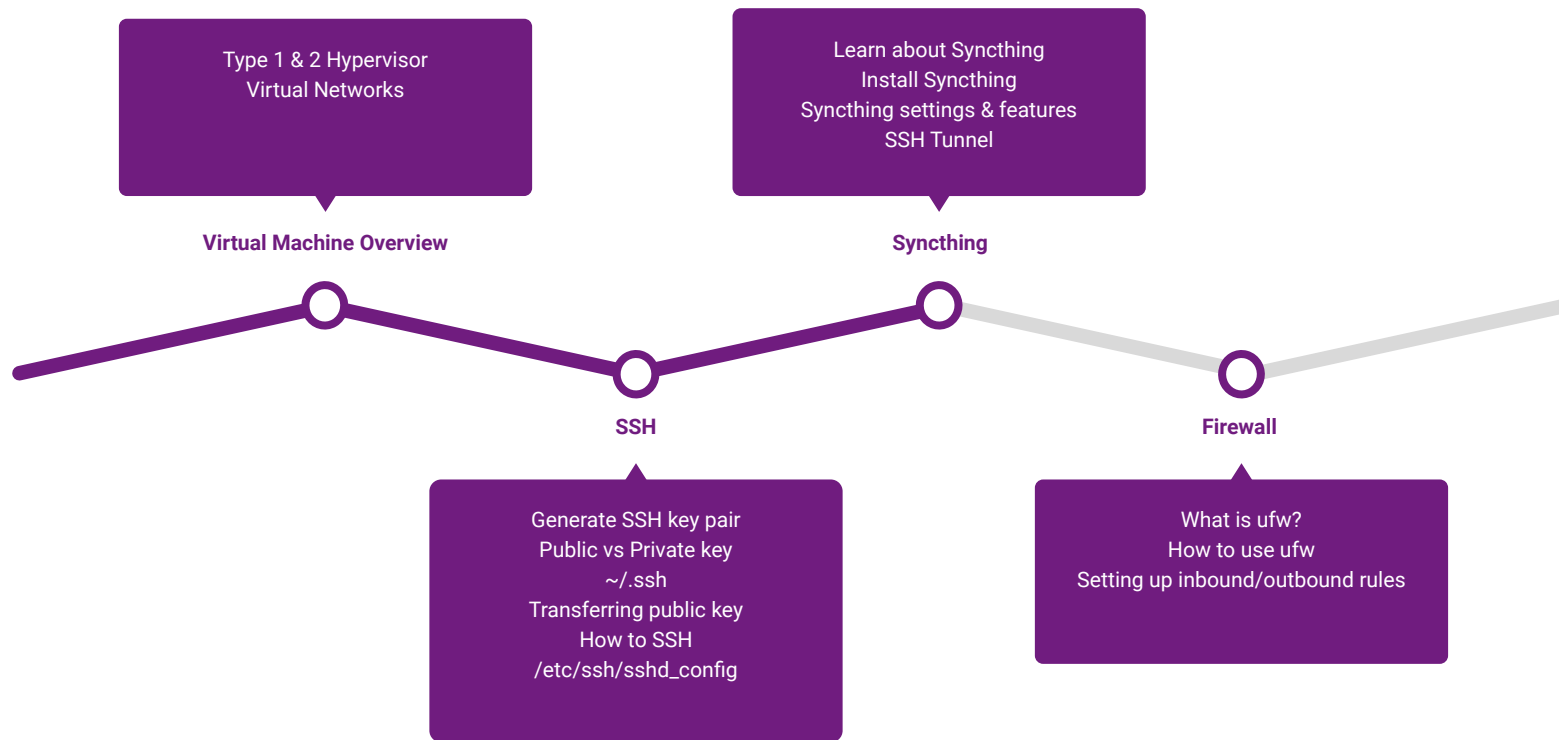
**ssh**: command to create an SSH connection

**-L**: flag to establish a local SSH tunnel (port forwarding)

**Input**: 8888:localhost:8384 rob@192.168.10.4

**ssh -L** 8888:localhost:8384 rob@192.168.10.4

# Overview



# Firewall/ufw

Firewalls monitor incoming/outgoing traffic and based on rule, it will either allow or block certain traffic

- You set the rules that define what kind of traffic the firewall will accept/deny

ufw has a 2 default rules when the firewall is activated:

- default **deny incoming**
- default **allow outgoing**

We are going to add 1 rule:

- `sudo ufw allow 22`

# What We've Accomplished!

