HOME LAB

MATEJ PAPAJ

2025

INFORMATION ABOUT THE STEPS TAKEN

Personal work

2025
KOŠICE

RIEŠITEĽ:
MATEJ PAPAJ

1. Introduction

This document provides details about the setup and configuration of my home server. It includes hardware specifications, operating system installation, services, configurations, and lessons learned during the process. It is also a replacement for the old notebook that was used before hand.

2. Hardware specifications

Device	Tabletop PC
CPU	Intel(R) Xeon(R) CPU E5-2682 v4 @ 2.50GHz
Motherboard	Intel X99-H5 (temporary)
Memory	32GB ECC DDR4
NVMe	Apacer AS2280P4 512GB
HDD	2TB
GPU	GK208B [GeForce GT 730]

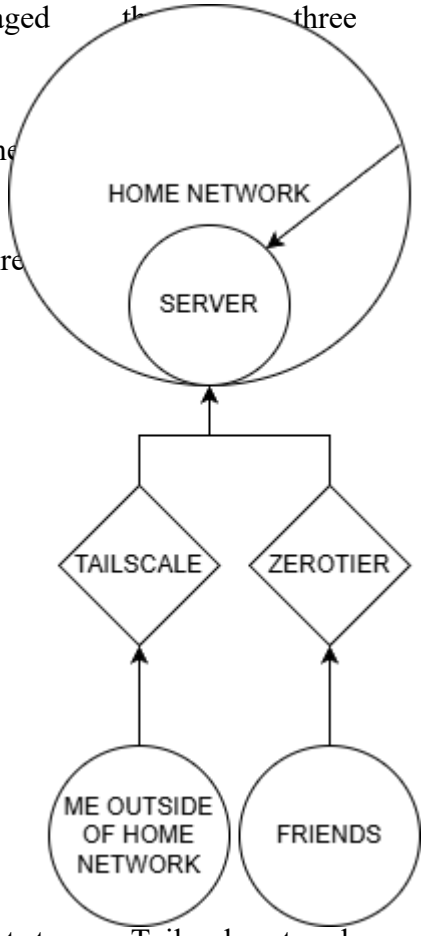
More info using lshw

OS

PRETTY_NAME	Ubuntu 24.04.3 LTS
NAME	Ubuntu
VERSION_ID	24.04
ID	ubuntu
ID_LIKE	debian
HOME_URL	https://www.ubuntu.com/
PRIVACY_POLICY_URL	https://www.ubuntu.com/legal/terms-and-policies/privacy-policy

3. Network

The access to the server is managed th three ways.



First is home network the access the then we have 2 access points with tunnel that is TailScale also the access is unre ZeroTier used for friends when hosting games.

Setup

Home network

Nothing

Tailscale

Install:

```
curl -fsSL https://tailscale.com/install.sh | sh
sudo tailscale up
```

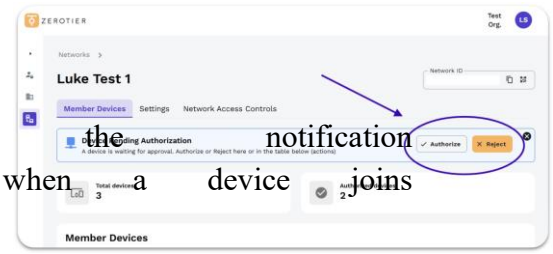
The output will display a URL that you can use to authenticate to your Tailscale network (known as a tailnet). After you authenticate, check the Machines page of the admin console to confirm the device appears in your tailnet.

ZeroTier

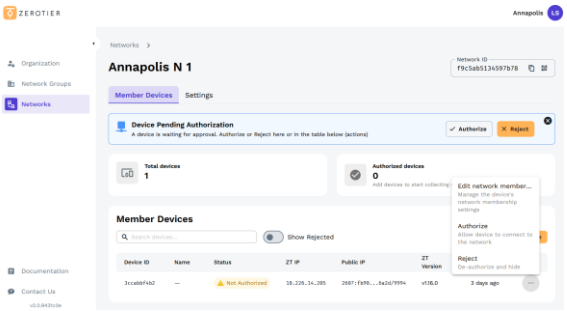
Install:

```
curl -s https://install.zerotier.com | sudo bash
sudo zerotier-cli join NETWORK_ID
```

When a device joins, you have two options:



Option 1: Respond to the notification
A notification pop-up will appear when a device joins
Click Authorize in the notification popup



Option 2: Manage devices
Go to the Member
Click on the device or use the menu to authorize

Afterwards the UFW needs to be configured to block all port except the ones for the game.

4. K3s

Install:

```
$ curl -sfL https://get.k3s.io | sh -
$ k3s server --write-kubeconfig-mode=644
```

Rancher:

install helm (to make rancher work):

```
$ curl -fsSL -o get_helm.sh https://raw.githubusercontent.com/helm/helm/main/scripts/get-helm-4
$ chmod 700 get_helm.sh
$ ./get_helm.sh


$ helm repo add rancher-latest https://releases.rancher.com/server-charts/latest
$ kubectl create namespace cattle-system
$ kubectl apply -f https://github.com/cert-manager/cert-manager/releases/download/<VERSION>/cert-manager.crds.yaml
$ helm repo add jetstack https://charts.jetstack.io
$ helm repo update
$ helm install cert-manager jetstack/cert-manager \
  --namespace cert-manager \
  --create-namespace
```

5. Games servers

Vintage story

Create /game_servers/vintage_story

```
sudo mkdir -p /game_servers/vintage_story
sudo chown -R 1000:1000 /game_servers/vintage_story
sudo chmod -R 755 /game_servers/vintage_story
```

Add to hdd lv:

```
sudo mount /dev/vg0/game-servers /game_servers
sudo blkid /dev/vg0/game-servers
sudo vim /etc/fstab
UUID=ID  /game_servers  ext4  defaults  0 2
```

Create these 3 files exactly:

1. manual-sc.yaml

```
apiVersion: storage.k8s.io/v1
kind: StorageClass
metadata:
  name: manual
provisioner: kubernetes.io/no-provisioner
volumeBindingMode: WaitForFirstConsumer
```

2. pv-pvc.yaml

```
apiVersion: v1
kind: PersistentVolume
metadata:
  name: vintage-story-pv
spec:
  capacity:
    storage: 50Gi
  accessModes:
    - ReadWriteOnce
  storageClassName: manual
  persistentVolumeReclaimPolicy: Retain
  hostPath:
    path: /game_servers/vintage_story
    type: DirectoryOrCreate
---
apiVersion: v1
kind: PersistentVolumeClaim
metadata:
  name: vintage-story-pvc
spec:
  accessModes:
```

```
    - ReadWriteOnce
  storageClassName: manual
  resources:
    requests:
      storage: 50Gi
```

3. statefulset.yaml

```
apiVersion: apps/v1
kind: StatefulSet
metadata:
  name: vintage-story
spec:
  serviceName: vintage-story
  replicas: 1
  selector:
    matchLabels:
      app: vintage-story
  template:
    metadata:
      labels:
        app: vintage-story
    spec:
      containers:
      - name: vintage-story
        image: zsuatem/vintagestory:1.21.6
        ports:
        - containerPort: 42420
        resources:
          requests:
            cpu: "4"
            memory: "8Gi"
          limits:
            cpu: "8"
            memory: "16Gi"
        volumeMounts:
        - name: world-data
          mountPath: /vintagestory/data
      volumes:
      - name: world-data
        persistentVolumeClaim:
          claimName: vintage-story-pvc
---
apiVersion: v1
kind: Service
metadata:
  name: vintage-story
spec:
  type: NodePort
  selector:    app: vintage-story
  ports:
    - port: 42420
      targetPort: 42420
      nodePort: 32420
```

Execute these commands in order:

1. Apply StorageClass, PV, PVC

```
$ sudo kubectl apply -f manual-sc.yaml
$ sudo kubectl apply -f pv-pvc.yaml
```

2. Verify PV/PVC bound

```
$ sudo kubectl get pv,pvc
```

3. Apply StatefulSet + Service

```
$ sudo kubectl apply -f statefulset.yaml
```

4. Verify pod running

```
$ sudo kubectl get pods
```

5. Test 2-way sync

```
$ sudo kubectl exec -it vintage-story-0 -- touch /vintagestory/data/pod-test
$ ls -la /game_servers/vintage_story/pod-test
$ touch /game_servers/vintage_story/host-test
$ sudo kubectl exec vintage-story-0 -- ls /vintagestory/data/host-test
```

6. Services

Jellyfin

Make these:

/services - lv_services – 100G

/movies - lv_movie – 1T

own 1000:1000 on both

jelly.yaml:

```
apiVersion: v1
kind: Namespace
metadata:
  name: jellyfin
---
apiVersion: apps/v1
kind: Deployment
metadata:
  name: jellyfin
  namespace: jellyfin
spec:
  replicas: 1
  selector:
    matchLabels:
      app: jellyfin
  template:
    metadata:
      labels:
        app: jellyfin
    spec:
      containers:
      - name: jellyfin
        image: jellyfin/jellyfin:latest
        env:
        - name: PUID
          value: "1000"
        - name: PGID
          value: "1000"
        - name: TZ
          value: "Europe/Bratislava"
        ports:
        - containerPort: 8096
          name: http
        volumeMounts:


        - name: config
          mountPath: /config
        - name: media
          mountPath: /media
      volumes:
      - name: config
        hostPath:
          path: /services/jellyfin
          type: DirectoryOrCreate
```

```
      - name: media
        hostPath:
          path: /movies
          type: Directory
---
apiVersion: v1
kind: Service
metadata:
  name: jellyfin
  namespace: jellyfin
spec:
  selector:
    app: jellyfin
  type: NodePort  # Changed from ClusterIP
  ports:
  - port: 8096
    targetPort: 8096
    protocol: TCP
    name: http
    nodePort: 30096  # Fixed high port (30000-32767 range); auto-assigned if omitted
```

Create a script (e.g., /usr/local/bin/iptables-startup.sh):

```
#!/bin/bash
iptables -t raw -I PREROUTING 1 -p tcp --dport 30096 -j DROP
iptables -t raw -I PREROUTING 1 -s 192.168.0.0/24 -p tcp --dport 30096 -j ACCEPT
iptables -t raw -I PREROUTING 1 -s 100.64.0.0/10 -p tcp --dport 30096 -j ACCEPT
```

Then:

```
$ sudo chmod +x /usr/local/bin/iptables-startup.sh
```

Create /etc/systemd/system/iptables-startup.service:

```
[Unit]
Description=Custom iptables rules
After=network.target

[Service]
Type=oneshot
ExecStart=/usr/local/bin/iptables-startup.sh
RemainAfterExit=yes

[Install]
WantedBy=multi-user.target
```

Enable and start the service:

```
$ sudo systemctl daemon-reload
$ sudo systemctl enable iptables-startup.service
$ sudo systemctl start iptables-startup.service
```

Pomodoro

mv html to /services/pomodoro/

index.html
style.css
script.js

there do:

```
$ sudo kubectl create configmap pomodoro-web --from-file=index.html --from-file=style.css --from-file=script.js
```

make a deployment.yaml:

```
apiVersion: apps/v1
kind: Deployment
metadata:
  name: pomodoro
spec:
  replicas: 1
  selector:
    matchLabels:
      app: pomodoro
  template:
    metadata:
      labels:
        app: pomodoro
    spec:
      containers:
      - name: nginx
        image: nginx:alpine
        ports:
        - containerPort: 80
        volumeMounts:
        - name: website
          mountPath: /usr/share/nginx/html
      volumes:
      - name: website
        configMap:
          name: pomodoro-web
---
apiVersion: v1
kind: Service
metadata:
  name: pomodoro
spec:
  type: NodePort
  selector:
    app: pomodoro
  ports:
    - port: 80
      targetPort: 80
      nodePort: 30808
```

and:

```
$ sudo kubectl apply -f pomodoro.yaml
```

Create or adjust a script (e.g., /usr/local/bin/iptables-startup.sh):

```
#!/bin/bash
iptables -t raw -I PREROUTING 1 -p tcp --dport 30808 -j DROP
iptables -t raw -I PREROUTING 1 -s 192.168.0.0/24 -p tcp --dport 30808 -j ACCEPT
iptables -t raw -I PREROUTING 1 -s 100.64.0.0/10 -p tcp --dport 30808 -j ACCEPT
```

Then:

```
$ sudo chmod +x /usr/local/bin/iptables-startup.sh
```

Create /etc/systemd/system/iptables-startup.service:

```
[Unit]
Description=Custom iptables rules
After=network.target

[Service]
Type=oneshot
ExecStart=/usr/local/bin/iptables-startup.sh
RemainAfterExit=yes

[Install]
WantedBy=multi-user.target
```

Enable and start the service:

```
$ sudo systemctl daemon-reload
$ sudo systemctl enable iptables-startup.service
$ sudo systemctl start iptables-startup.service
```

7. More information to all

Tailscale - https://tailscale.com/kb/1031/install-linux

Zerotier - https://www.zerotier.com/download/
https://docs.zerotier.com/quickstart/

K3s - https://docs.k3s.io/quick-start

Rancher - https://ranchermanager.docs.rancher.com/getting-started/quick-start-guides/deploy-rancher-manager/helm-cli