Server Installation

Pre-requisites

- A machine running Ubuntu Server 20.04 LTS
- Machine must have a public facing IP address

Initial configuration of server

Updates software on the machine

sudo apt update -y && sudo apt upgrade -y

Installation of docker

Installs docker onto the machine3, you can either use snap or apt to install docker

sudo apt install docker.io -y

OR

sudo snap install docker

Configure FreeTAKServer (FTS) with docker

Creates a docker volume with the name 'fts_data'

sudo docker volume create fts_data

Creates a docker container running FreeTAKServer:

- As the docker image isn't on the machine, docker will download it from the docker hub
- The red ports are the ports for the ubuntu machine, you can change these to whatever you want but make sure to update firewall rules appropriately.
- Docker will utilise the 'fts_data' volume created earlier

sudo docker run -d -p xxxxx:8080/tcp -p xxxxx:8087/tcp -e FTS_CONNECTION_MESSAGE="Server Connection Message" -e FTS_COT_TO_DB="True" -v fts_data:/data --name fts --restart unless-stopped freetakteam/freetakserver:1.1.2

NOTE:

- Data packages that are uploaded to the server are done via the **"xxxxx:8080"** port configuration, this will need to be changed on the client device under:
  "Settings" - "Show All Preferences" - "Network Preferences" – "Network Connection Preferences" – "Unsecure Server API Port"
- The '**xxxxx:8087'** port will be the port used to configure network connections on client devices.

ZeroTier Installation

Install ZeroTier

curl -s https://install.zerotier.com | sudo bash

Download ztncui

curl -O https://s3-us-west-1.amazonaws.com/key-networks/deb/ztncui/1/x86_64/ztncui_0.8.6_amd64.deb

Install ztncui

sudo apt-get install ./ztncui_0.8.6_amd64.deb

sudo sh -c "echo 'HTTPS_PORT=3443' > /opt/key-networks/ztncui/.env"

sudo sh -c "echo 'NODE_ENV=production' >> /opt/key-networks/ztncui/.env"

sudo systemctl restart ztncui

sudo systemctl enable ztncui

NOTE:

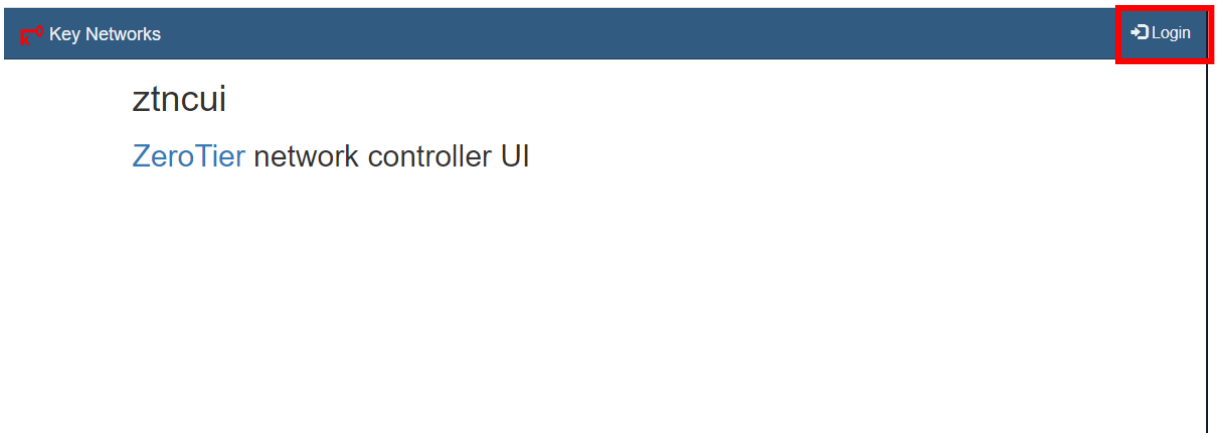
- The server will listen on port 3443 for https traffic
- The web interface is using TLS but is using a self-signed certificate

ZeroTier Server Configuration

Navigate to the web page for your zerotier server

https://xxx.xxx.xxx.xxx:3443/

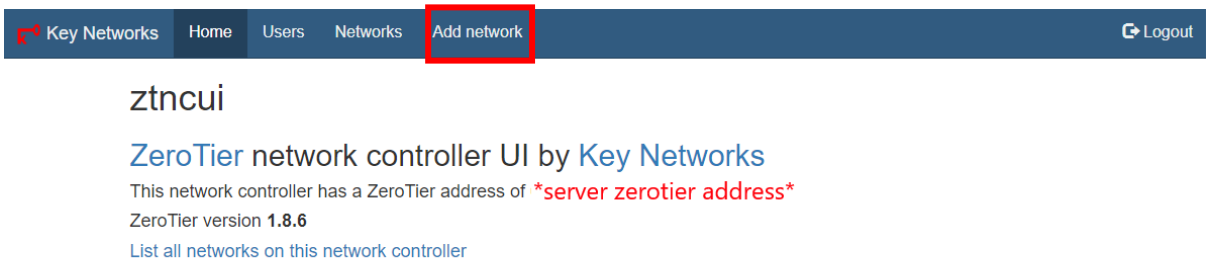
Login with the default credentials



Username – admin

Password – password

You will be prompted to set a new password, after changing the password navigate to the 'Add network' page



Enter a network name and select ‘Create Network’

From here you will want to configure the following settings:

NOTE: For this example, a 10.10.10.0/24 network will be used

Routes:

Add your internal subnet for the VPN to allow ZT to route internally:

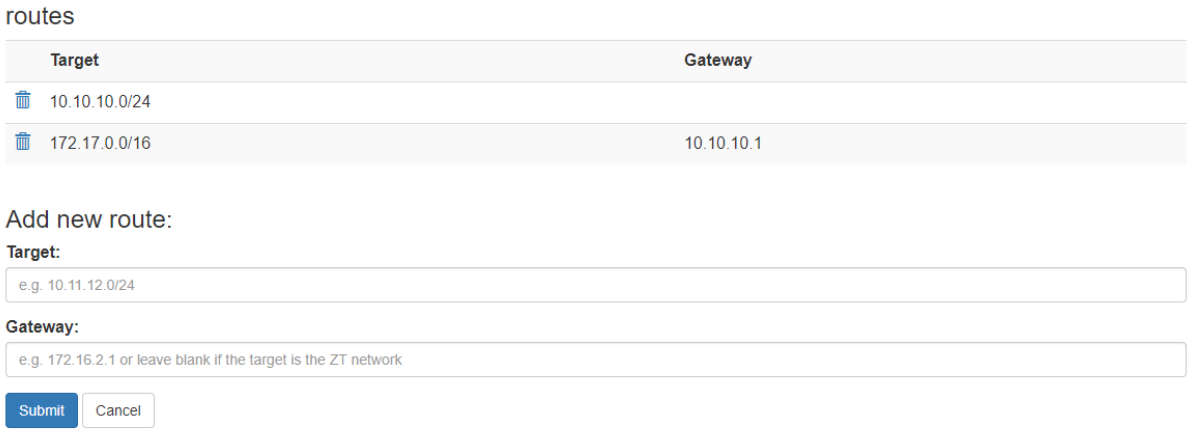
Target: 10.10.10.0/24

Gateway: Leave blank

Add the docker network to route through the ZT server:

Target: 172.17.0.0/16 (default subnet for docker network)

Gateway: 10.10.10.1



Assignment Pools:

Assign your start and end IP for assignment:

IP range start: 10.10.10.2

IP range end: 10.10.10.10

ipAssignmentPools

	IP range start	IP range end
🗑	10.10.10.2	10.10.10.10

Add new IP Assignment Pool:

IP range start:

IP range start

IP range end:

IP range end

Submit Cancel

IPv4 Assign Mode:

Select the checkbox marked “Auto-assign from IP Assignment Pool”

v4AssignMode

☑ Auto-assign from IP Assignment Pool

DNS:

Assign DNS servers in the “Servers” text field, one server per line

dns

Domain:
Servers: 1.1.1.1
1.0.0.1

Change DNS configuration:

Domain:

Servers:

1.1.1.1
1.0.0.1

Submit Cancel

Adding ZeroTier Server to network

This will join the ZT Controller to the network in order to allow traffic to be routed from end devices to the FreeTakServer within the docker container.

Join the server to the network

sudo zerotier-cli join *network id*

On the ZT web interface authorise the server and select the “Active Bridge” checkbox

	Member name	Member ID	Authorized	Active bridge	IP assignment	Peer status	Peer address / latency
🗑	server	2adbf2fd28	☑	☑	10.10.10.1	CONTROLLER	

Select the link under “IP Assignment” and manually set the IP address to 10.10.10.1

ipAssignments

Member name: **server** ZeroTier address: **2adbf2fd28**

	IP address
🗑	10.10.10.1
+	IP address

Managed routes

	Target	Gateway
🗑	10.10.10.0/24	
🗑	172.17.0.0/16	10.10.10.1

Identifying Docker IP address to connect to from within ZT network

List IP addressing information

ip addr

OR

ifconfig

You are looking for the docker interface, example below:

```
docker0: flags=4163<UP,BROADCAST,RUNNING,MULTICAST>  mtu 1500
        inet 172.17.0.1  netmask 255.255.0.0  broadcast 172.17.255.255
        inet6 fe80::42:8cff:feca:b636  prefixlen 64  scopeid 0x20<link>
        ether 02:42:8c:ca:b6:36  txqueuelen 0  (Ethernet)
        RX packets 549  bytes 118491 (118.4 KB)
        RX errors 0  dropped 0  overruns 0  frame 0
        TX packets 7647  bytes 382936 (382.9 KB)
        TX errors 0  dropped 0 overruns 0  carrier 0  collisions 0
```

This is the IP address used to connect client devices to the FTS server when connected via ZeroTier,

NOTE: Use the same port assignment as if connected from outside the ZeroTier network, only the IP address changes

Iptables Configuration (optional)

READ THIS FIRST: If using a cloud provider to host the server, it is not necessarily needed to implement iptables rules to restrict ports on the machine itself as the cloud provider's services can be configured to filter incoming traffic as it is processed before it reaches the machine e.g. security groups on AWS. It is recommended for simplicities sake to use the cloud provider's integrated means to filter traffic. If you are hosting services locally and have no other firewall, use the rules below to implement traffic filtering.

Install iptables persistent package (allows iptables to keep rules after reboot)

sudo apt-get install iptables-persistent -y

NOTE: You will get prompted with 2 yes/no prompts for saving current rules, answer yes on both prompts

Apply base level iptables rules to allow ssh and the ports for zerotier and fts, insert the port numbers previously specified here in place of "xxxxx"

sudo iptables -A INPUT -p tcp --dport xxxxx -m tcp -j ACCEPT

sudo iptables -A INPUT -p tcp --dport xxxxx -m tcp -j ACCEPT

sudo iptables -A INPUT -p tcp --dport 3443 -m tcp -j ACCEPT

sudo iptables -A INPUT -p tcp --dport 22 -m tcp -j ACCEPT

sudo iptables -A INPUT -i lo -j ACCEPT

sudo iptables -A OUTPUT -o lo -j ACCEPT

sudo iptables -A INPUT -m conntrack --ctstate ESTABLISHED,RELATED -j ACCEPT

sudo iptables -A INPUT -m conntrack --ctstate INVALID -j DROP

sudo iptables -A INPUT -p tcp --dport 22 -m conntrack --ctstate NEW,ESTABLISHED -j ACCEPT

sudo iptables -A INPUT -j DROP

Overwrite the old rules with your new iptables rules so it will apply on start-up

sudo su

iptables-save > /etc/iptables/rules.v4

exit