

# System

How to administer core system services such as networking, storage, monitoring, etc.

- [audit](#)
- [Checklist of Things Done](#)
- [crypttab](#)
- [Docker Firewall Configuration](#)
- [FiOS Router](#)
- [Intel NIC Configuration](#)
- [lm-sensors](#)
- [LUKS](#)
- [Minecraft](#)
- [PiKVM](#)
- [PiKVM Tailscale Certificate Update Service](#)
- [Promtail](#)
- [traefik](#)
- [Users/Groups](#)

# audit

Kernel Parameters:

```
audit=1 audit_backlog_limit=8192
```

This prevents the message

```
kauditd: hold queue overflow
```

# Checklist of Things Done

## Network

## Storage

hdparm -S 0 for all drives

hdparm -B 164 for all drives

## System

suricata

# crypttab

This configuration allows us to automatically unlock but not mount external drives. For example:

```
/etc/crypttab
diskn          UUID=<path to disk by /dev/disk/by-uuid>    /etc/keyfiles/<keyfile name>
luks,nofail
```

This configuration will use the keyfile `/etc/keyfiles/keyfile` to `/dev/disk/by-uuid/id` and create a device node for that disk at `/dev/mapper/diskn` for mounting in `fstab`.

Do not directly mount the disk

# Docker Firewall Configuration

Source: [Firewalld Strict Docker Filtering](#)

## Preparation

Required parts:

Install firewalld and activate service:

```
pacman -Syu firewalld
systemctl enable --now firewalld.service
```

Disable any other firewall services.

Disable iptables for docker by adding or changing `/etc/docker/daemon.json` by adding the following config options:

```
{
  "iptables": false
}
```

After changing this config file, restart the Docker daemon to apply the previous change:

```
systemctl restart docker.service
```

As a result of the previous steps, only allowed ports on firewalld are accessible from the outside. However containers are now unable to connect outbound to the internet.

## firewalld Configuration

### Allow internet access for Docker containers

We need to allow masquerading to allow traffic from the Docker zone to the internet:

```
# Running this command allows your containers to reach out to the internet
firewall-cmd --permanent --zone=home --add-masquerade
```

```
# Since we used the --permanent flag, we need to reload the firewall for the changes to take effect  
firewall-cmd --reload
```

Docker creates a zone in firewalld specifically for its bridge network interfaces for each container network along with the docker0 interface.

To fix networking for containers that are not connected to a docker network, add your network interface connected to the network to the masqueraded zone.

# FiOS Router

## Set Router to Bridge Mode

Login to router administration interface

Select "My Network" on the top bar

<https://kb.its-et.me/uploads/images/gallery/2024-07/NjfmO28jwrLeg9rS-image.png>

Select "Network Connections" > "Advanced"

[image.png](#)

Select edit icon for "Network (Home/Office)", then click "Settings" on the bottom right

[image.png](#)

Check the box for bridge mode under the "Bridge" section

[image.png](#)

If you set up another router and it detects that the ISP modem/router is still active, it will use the 10.0.0.0/16 network rather than the 192.168.0.0/24 network

## Configure IP Allocation

Login to router administration interface

Select "Advanced" on the top bar, then select "Yes"

[image.png](#)

Under "Routing", select "IP Address Distribution"

[image.png](#)

Select "Connection List" on the bottom

[image.png](#)

Edit a specific host's IP allocation



# Intel NIC Configuration

## Wireless Configuration

```
# iwlwifi.conf  
# Enable antenna aggregation  
options iwlwifi 11n_disable=8
```

# Im-sensors

Label	Value
CPUTIN	Motherboard's CPU temp sensor
SYSTIN	Motherboard temp sensor
AUXTIN	Aux temp sensors, usually for PSU

# LUKS

[https://wiki.archlinux.org/title/Dm-crypt/Specialties#Disable\\_workqueue\\_for\\_increased\\_solid\\_state\\_drive\\_\(SSD\)\\_performance](https://wiki.archlinux.org/title/Dm-crypt/Specialties#Disable_workqueue_for_increased_solid_state_drive_(SSD)_performance)

# Minecraft

## Client

### Launcher for running the mods

I use the Fabric launcher, get its installer from the AUR with the following command:

```
$ aurget fabric-installer
```

Since this is a graphical installer, launch and select the Minecraft version that it should install a launcher for. It will appear as an entry in the Mojang launcher.

### Installer for the mods themselves

Install either `fabric` or `fabric-bin`. I typically use the latter since I get build issues with the former.

Follow instructions and we are good.

List of mods that I run on the client:

- fabric-api
- sodium
- lithium
- phosphor
- effective
- visuality
- iris
- ferrite-core
- coordinates-display
- lambdynamicalights
- betterf3
- enhancedblockentities (ebe)
- AppleSkin
- spark
- immediatelyfast
- sound-physics-remastered
- exordium
- better-animations-collection

# Server

## Architecture

### Firewalld Forwarding Setup

I run the actual server on `krypton`, but use `oxygen` to proxy the traffic, masking the actual IP address. Use port forwarding through `firewalld` to achieve this:

```
# firewall-cmd --permanent --zone=public --add-forward-  
port=port=25565:proto=tcp:toport=25565:toaddr=<addr of target server>  
# firewall-cmd --permanent --zone=public --add-forward-  
port=port=25565:proto=udp:toport=25565:toaddr=<addr of target server>  
# firewall-cmd --permanent --zone=internal --add-masquerade  
# firewall-cmd --reload
```

Add a masquerade to the zone where the target server is located at. This will allow traffic to pass from the public zone to that target zone, which in this case is `internal`.

## Traefik

I run the actual server on `krypton`, but use `oxygen` to proxy the traffic, masking the actual IP address. Use a TCP router through `traefik` with `HostSNI(`*`)` to allow access to the server.

Here is an example configuration using the file provider on traefik:

```
tcp:  
  routers:  
    minecraft:  
      entrypoints:  
        - "minecraft"  
      rule: "HostSNI(`*`)"  
      service: minecraft  
  
  services:  
    minecraft:  
      loadBalancer:
```

```
servers:  
  - address: "/path/to/server/host:port"
```

## Mods

- simply-optimized modpack
- no-chat-reports
- no-telemetry
- krypton
- spark
- ledger

## Whitelisting

This server is exposed to the public internet, whitelisting is required to prevent bots and unauthorized players from joining the server

The whitelist file is a JSON-formatted file that contains player usernames and UUIDs. That in combination with online mode prevents attackers from spoofing accounts.

Here is the file structure for the player whitelist:

```
[  
  {  
    "name": "<mojang username 1>",  
    "uuid": "<uuid 1>"  
  },  
  {  
    "name": "<mojang username 2>",  
    "uuid": "<uuid 2>"  
  },  
]
```

To retrieve the UUID of a user, run the following command:

```
curl --silent https://api.mojang.com/users/profiles/minecraft/<mojang username>
```

This command will send a JSON response with "name" and "id" attributes, don't forget to change "id" to "uuid" for whitelisting to work properly.

The UUID will be returned in the HTTP response.

To enable parsing and managing of this whitelist:

```
ENABLE_WHITELIST="true"
ENFORCE_WHITELIST="true"
WHITELIST_FILE="/path/to/whitelist/file.json"
EXISTING_WHITELIST_FILE="SYNC_FILE_MERGE_LIST"
```

## Server Ops

Similar to the whitelist file, the ops file is in JSON format as seen below:

```
[
  {
    "name": "<mojang username 1>",
    "uuid": "<uuid 1>",
    "level": 4,
    "bypassesPlayerLimit": true
  },
  {
    "name": "<mojang username 2>",
    "uuid": "<uuid 2>",
    "level": 3,
    "bypassesPlayerLimit": false
  },
]
```

```
OPS_FILE="/path/to/ops/file.json"
EXISTING_OPS_FILE="SYNC_FILE_MERGE_LIST"
```

# PiKVM

## Important Lines for /boot/config.txt

Used to fix kvmd-otg and kvmd-tc358743 not starting at boot

```
dtoverlay=tc358743  
dtoverlay=disable-bt  
dtoverlay=dwc2,dr_mode=peripheral
```



# PiKVM Tailscale Certificate Update Service

These systemd services allow me to update the Tailscale certificates for PiKVM every 80 days without manual intervention.

## cert-update.timer

```
[Unit]
Description=Update tailscale certificates for nginx

[Timer]
OnBootSec=1min
OnUnitActiveSec=80d
AccuracySec=1h
Persistent=true

[Install]
WantedBy=timers.target
```

## tailscale-cert-update.service

```
[Unit]
Description=Update tailscale certificates for nginx
After=network-online.target tailscaled.service

[Service]
Type=oneshot

# Service isolation
ProtectHome=true
ReadWritePaths=/etc/kvmd/nginx/ssl
PrivateNetwork=false
ProtectClock=true
ProtectHostname=true
ProtectKernelTunables=true
ProtectKernelModules=true
```

```
ProtectControlGroups=true
```

```
LockPersonality=true
```

```
# Execution steps
```

```
ExecStartPre=/usr/bin/curl --silent --max-time 10 --retry 5 https://hc.its-et.me/ping/PlGPBqq-0rLI4N4ya3jYmg/pve-01k-certificate-update/start
```

```
ExecStartPre=/usr/bin/rw
```

```
ExecStart=tailscale cert --cert-file=/etc/kvmd/nginx/ssl/server.crt --key-file=/etc/kvmd/nginx/ssl/server.key pve-01k.tail755c5.ts.net
```

```
ExecStartPost=/usr/bin/curl --silent --max-time 10 --retry 5 https://hc.its-et.me/ping/PlGPBqq-0rLI4N4ya3jYmg/pve-01k-certificate-update
```

```
ExecStartPost=/usr/bin/systemctl restart kvmd-nginx.service
```

```
ExecStartPost=/usr/bin/ro
```

```
[Install]
```

```
WantedBy=default.target
```

Don't use `PrivateDevices=` in `[Service]`, this disallows `/usr/bin/ro` and `/usr/bin/rw` from executing properly

# Promtail

## My Default Promtail Configuration

```
# Global promtail configuration
server:
  http_listen_port: 9080
  grpc_listen_port: 0

clients:
  - url: http://<log-host>:3100/loki/api/v1/push

limits_config:
  max_line_size: 384kb

scrape_configs:
  - job_name: web
    file_sd_configs:
      - files:
          - /etc/loki/web.yaml

  - job_name: flog_scrape
    docker_sd_configs:
      - host: unix:///var/run/docker.sock
        refresh_interval: 5s
        #filters:
        #  - name: label
        #    # Add logging=promtail label to enable log capture
        #    values: ["logging=promtail"]
    relabel_configs:
      - source_labels: ["__meta_docker_container_name"]
        regex: "/(.*)"
        target_label: "container"
      - source_labels: ["__meta_docker_container_log_stream"]
        target_label: "logstream"
      - source_labels: ["__meta_docker_container_label_logging_jobname"]
```

```
    target_label: "job"

- job_name: systemd-journal
  journal:
    labels:
      cluster: ops-tools1
      job: default/systemd-journal
    path: /var/log/journal
  relabel_configs:
    - source_labels:
        - __journal__systemd_unit
      target_label: systemd_unit
    - source_labels:
        - __journal__hostname
      target_label: hostname
    - source_labels:
        - __journal_syslog_identifier
      target_label: syslog_identifier
```

Don't forget to verify if there is a separate promtail user that needs to be added to the docker group for container logs

# traefik

## Docker Label Configuration

### Base Labels

This is the minimum set of labels you need to expose a container to traefik:

```
labels:
  traefik.enable: true
  traefik.http.routers.<service_name>.entrypoints: <ep1>, <ep2>
  traefik.http.routers.<service_name>.rule: Host(`host1`, `host2`)
  traefik.http.routers.<service_name>.tls: true
  traefik.http.routers.<service_name>.tls.certresolver: <cert_resolver>
```

### Middleware configuration

To configure a middleware for a particular service, add the following label:

```
traefik.http.routers.<service_name>.middlewares: middleware@provider
```

### Accessing on a non-default port

If a container exposes multiple ports or a non-default port:

```
traefik.http.services.<service_name>.loadbalancer.server.port: <port_num>
```

## Networking

To expose only containers on a certain network to traefik, you must specify the `providers.docker.network` option as so:

```
providers:
  docker:
    endpoint:
      exposedByDefault: false # Require label in docker-compose file for each container
      network: <net_name>
```

```
watch: true
```

If traefik itself is running in a docker container, you must place it on the same network as the containers you want to expose.

# TLS

Basic TLS configuration that enables resolvers for both single-domain and wildcard Let's Encrypt certificates, as well as staging certificates:

```
# ===== TLS Configuration =====
tls:
  # Disable TLS version 1.0 and 1.1
  options:
    default:
      minVersion: VersionTLS12
      sniStrict: true

  certificatesResolvers:
    staging:
      acme:
        email: "email@email.com"
        storage: /etc/traefik/certs/acme.json
        caServer: "https://acme-staging-v02.api.letsencrypt.org/directory"
        tlsChallenge: {}

    production:
      acme:
        email: "email@email.com"
        storage: /etc/traefik/certs/acme.json
        caServer: "https://acme-v02.api.letsencrypt.org/directory"
        tlsChallenge: {}
```

Wildcard certificates can only be obtained with the DNS-01 challenge. Therefore a resolver that uses these must have dnsChallenge configured accordingly.

# Tailscale

When running traefik in a docker container, ensure that it has access to the tailscale socket to be able to issue TLS certificates through tailscale

# Users/Groups

## krypton

User	Group	Type (login/system)	Purpose
restic	backup	system	Run the restic-rest-server
www-srv	www	system	Run web-accessible services
-	timemachine		
traefik	traefik	system	Run the t
	syncthing		
	paperless		
	docker		
	users		
	wheel		

## oxygen
