05. Creating the Docker Compose file

Now, that the NFS share and folder structure is created and added to the Virtual machine with /etc/fstab we can start creating the docker compose file and its environment ".env" file. It must be said that this docker compose file is basically a copy paste from the compose file created by home lab teacher Techhut. You can find his compose file here. In this docker page I will first explain all the different components and what they do. However, you can also directly copy and paste docker compose and environment file if you want of course.

Docker Compose File

```
networks:
  servarrnetwork:
    name: servarrnetwork
    ipam:
      config:
        - subnet: 172.39.0.0/24
services:
  gluetun:
    image: qmcgaw/gluetun
    container_name: gluetun
    cap_add:
      - NET_ADMIN
    devices:
      - /dev/net/tun:/dev/net/tun # If running on an LXC see readme for more info.
    networks:
      servarrnetwork:
        ipv4_address: ${SET_IP_GLUETUN}
    ports:
      - 8080:8080 # qbittorrent web interface
      - 6881:6881 # qbittorrent torrent port
      - 6789:6789 # nzbget
      - 9696:9696 # prowlarr
      - 8191:8191 # flaresolverr
```

```
    volumes:
      - ./gluetun:/gluetun
    # Make a '.env' file in the same directory.
    env_file:
      - .env
    healthcheck:
      test: ping -c 1 www.google.com || exit 1
      interval: 20s
      timeout: 10s
      retries: 5
    restart: unless-stopped
  qbittorrent:
    image: lscr.io/linuxserver/qbittorrent:latest
    container_name: qbittorrent
    restart: unless-stopped
    labels:
      - deunhealth.restart.on.unhealthy=true
    environment:
      - PUID=${PUID}
      - PGID=${PGID}
      - TZ=${TZ}
      - WEBUI_PORT=8080 # must match "qbittorrent web interface" port number in gluetun's
service above
      - TORRENTING_PORT=${FIREWALL_VPN_INPUT_PORTS} # airvpn forwarded port, pulled from
.env
    volumes:
      - ./qbittorrent:/config
      - ${ROOT_DIR}:/data
    depends_on:
      gluetun:
        condition: service_healthy
        restart: true
    network_mode: service:gluetun
    healthcheck:
      test: ping -c 1 www.google.com || exit 1
      interval: 60s
      retries: 3
      start_period: 20s
      timeout: 10s
```

```
  # See the 'qBittorrent Stalls with VPN Timeout' section for more information.
  deunhealth:
    image: qmcgaw/deunhealth
    container_name: deunhealth
    network_mode: none
    environment:
      - LOG_LEVEL=info
      - HEALTH_SERVER_ADDRESS=127.0.0.1:9999
      - TZ=${TZ}
    restart: always
    volumes:
      - /var/run/docker.sock:/var/run/docker.sock
  nzbget:
    image: lscr.io/linuxserver/nzbget:latest
    container_name: nzbget
    environment:
      - PUID=${PUID}
      - PGID=${PGID}
      - TZ=${TZ}
    volumes:
      - /etc/localtime:/etc/localtime:ro
      - ./nzbget:/config
      - ${ROOT_DIR}:/data
    depends_on:
      gluetun:
        condition: service_healthy
        restart: true
    restart: unless-stopped
    network_mode: service:gluetun
  prowlarr:
    image: lscr.io/linuxserver/prowlarr:latest
    container_name: prowlarr
    environment:
      - PUID=${PUID}
      - PGID=${PGID}
      - TZ=${TZ}
    volumes:
      - /etc/localtime:/etc/localtime:ro
      - ./prowlarr:/config
```

```
    restart: unless-stopped
    depends_on:
      gluetun:
        condition: service_healthy
        restart: true
    network_mode: service:gluetun
  flaresolverr:
    image: ghcr.io/flaresolverr/flaresolverr:latest
    container_name: flaresolverr
    environment:
      - LOG_LEVEL=${LOG_LEVEL:-info}
      - LOG_HTML=${LOG_HTML:-false}
      - CAPTCHA_SOLVER=${CAPTCHA_SOLVER:-none}
      - TZ=${TZ}
    depends_on:
      gluetun:
        condition: service_healthy
        restart: true
    network_mode: service:gluetun
    restart: unless-stopped
  sonarr:
    image: lscr.io/linuxserver/sonarr:latest
    container_name: sonarr
    restart: unless-stopped
    environment:
      - PUID=${PUID}
      - PGID=${PGID}
      - TZ=${TZ}
    volumes:
      - /etc/localtime:/etc/localtime:ro
      - ./sonarr:/config
      - ${ROOT_DIR}:/data
    ports:
      - 8989:8989
    networks:
      servarrnetwork:
        ipv4_address: ${SET_IP_SONARR}
  radarr:
    image: lscr.io/linuxserver/radarr:latest
```

```
    container_name: radarr
    restart: unless-stopped
    environment:
      - PUID=${PUID}
      - PGID=${PGID}
      - TZ=${TZ}
    volumes:
      - /etc/localtime:/etc/localtime:ro
      - ./radarr:/config
      - ${ROOT_DIR}:/data
    ports:
      - 7878:7878
    networks:
      servarrnetwork:
        ipv4_address: ${SET_IP_RADARR}
  lidarr:
    container_name: lidarr
    image: lscr.io/linuxserver/lidarr:latest
    restart: unless-stopped
    volumes:
      - /etc/localtime:/etc/localtime:ro
      - ./lidarr:/config
      - ${ROOT_DIR}:/data
    environment:
      - PUID=${PUID}
      - PGID=${PGID}
      - TZ=${TZ}
    ports:
      - 8686:8686
    networks:
      servarrnetwork:
        ipv4_address: ${SET_IP_LIDARR}
  bazarr:
    image: lscr.io/linuxserver/bazarr:latest
    container_name: bazarr
    restart: unless-stopped
    environment:
      - PUID=${PUID}
      - PGID=${PGID}
```

```
    - TZ=${TZ}
  volumes:
    - /etc/localtime:/etc/localtime:ro
    - ./bazarr:/config
    - ${ROOT_DIR}:/data
  ports:
    - 6767:6767
  networks:
    servarrnetwork:
      ipv4_address: ${SET_IP_BAZARR}
seerr:
  container_name: seerr
  image: ghcr.io/seerr-team/seerr:latest
  environment:
    - LOG_LEVEL=debug
    - PUID=${PUID}
    - PGID=${PGID}
    - TZ=${TZ}
  healthcheck:
    test: wget --no-verbose --tries=1 --spider http://localhost:5055/api/v1/status
      || exit 1
    start_period: 20s
    timeout: 3s
    interval: 15s
    retries: 3
  restart: unless-stopped
  volumes:
    - ./seerr:/app/config
  ports:
    - 5055:5055
  networks:
    servarrnetwork:
      ipv4_address: ${SET_IP_SEERR}
# jellystat-db:
#   image: postgres:18.1
#   container_name: jellystat-db
#   restart: unless-stopped
#   shm_size: '1gb'
#   environment:
```

```
  #     - POSTGRES_USER=${JELLYSTAT_POSTGRES_USER}
  #     - POSTGRES_PASSWORD=${JELLYSTAT_POSTGRES_PASSWORD}
  #   volumes:
  #     - ./jellystat/postgres-data:/var/lib/postgresql/data
  #   healthcheck:
  #     test: ["CMD-SHELL", "pg_isready --dbname=postgres --
username=${JELLYSTAT_POSTGRES_USER}"]
  #     interval: 10s
  #     timeout: 5s
  #     retries: 5
  #   networks:
  #     servarrnetwork:
  #       ipv4_address: ${SET_IP_JELLYSTAT_DB}
  #
  # jellystat:
  #   image: cyfershepard/jellystat:latest
  #   container_name: jellystat
  #   restart: unless-stopped
  #   environment:
  #     - POSTGRES_USER=${JELLYSTAT_POSTGRES_USER}
  #     - POSTGRES_PASSWORD=${JELLYSTAT_POSTGRES_PASSWORD}
  #     - POSTGRES_IP=jellystat-db
  #     - POSTGRES_PORT=5432
  #     - JWT_SECRET=${JELLYSTAT_JWT_SECRET}
  #     - TZ=${TZ}
  #   volumes:
  #     - ./jellystat/backup-data:/app/backend/backup-data
  #   ports:
  #     - 3000:3000
  #   depends_on:
  #     jellystat-db:
  #       condition: service_healthy
  #   healthcheck:
  #     test: wget --no-verbose --tries=1 --spider http://localhost:3000/auth/isConfigured
|| exit 1
  #     interval: 60s
  #     timeout: 30s
  #     retries: 5
  #     start_period: 30s
```

```
  #   networks:
  #     servarrnetwork:
  #       ipv4_address: ${SET_IP_JELLYSTAT}
```

Environment Variables

```
# General UID/GID and Timezone
TZ=America/Los_Angeles
PUID=1000
PGID=1000

# Input your VPN provider and type here
VPN_SERVICE_PROVIDER=airvpn
VPN_TYPE=wireguard

# Mandatory, airvpn forwarded port
FIREWALL_VPN_INPUT_PORTS=port

# Copy all these variables from your generated configuration file
WIREGUARD_PUBLIC_KEY=key
WIREGUARD_PRIVATE_KEY=key
WIREGUARD_PRESHARED_KEY=key
WIREGUARD_ADDRESSES=ip

# Optional location variables, comma separated list, no spaces after commas, make sure it matches the config you created
# NOTE: These can cause connection failures with some providers. Remove or comment out if Gluetun won't connect.
#SERVER_COUNTRIES=country
#SERVER_CITIES=city

# Health check duration
HEALTH_VPN_DURATION_INITIAL=120s

# Static IPs for services on servarrnetwork
SET_IP_GLUETUN=172.39.0.2
```

```
SET_IP_SONARR=172.39.0.3
SET_IP_RADARR=172.39.0.4
SET_IP_LIDARR=172.39.0.5
SET_IP_BAZARR=172.39.0.6
SET_IP_SEERR=172.39.0.7
SET_IP_JELLYSTAT_DB=172.39.0.8
SET_IP_JELLYSTAT=172.39.0.9

# Jellystat (uncomment service block in compose.yaml to enable)
JELLYSTAT_POSTGRES_USER=postgres
JELLYSTAT_POSTGRES_PASSWORD=changeme
JELLYSTAT_JWT_SECRET=changeme
```

Network

1. First of all an internal docker network is created called "servarrnetwork"
2. Then we give this network as well a subnet in a /24. This can be any private subnet you please since it is only internal

Gluetun

So what does this docker compose file represent ? What is the workflow here ?

Revision #1
Created 2026-07-05 21:56:16 UTC by dpasn
Updated 2026-07-05 21:56:17 UTC by dpasn