Homelab Workshop

Session 2: Hands-On Implementation

Terraform, Docker, Tailscale — let's build

Connect to Workshop WiFi

Before we start, connect to the workshop network:

SSID	`homelab-workshop`
Password	`DevOpsJogja123`

Once connected, verify you can reach your group's Proxmox server:

Group	Proxmox UI
Group 1	`https://192.168.10.101:8006`
Group 2	`https://192.168.10.102:8006`
Group 3	`https://192.168.10.103:8006`

Login: Username: `root` · Password: `Homelab123`

Quick Recap & Today's Agenda

Session 1 covered: Homelabs, hardware, VLANs, Proxmox

Today we build:

Time	What We Do
15 min	**Terraform Setup** — install, API token, provider config
15 min	**Deploy LXC** — docker-host container via Terraform
10 min	**Tailscale** — VPN for remote access
25 min	**Docker Compose** — AdGuard, Cloudflare Tunnel, and more
5 min	**Additional Services** — Immich, Nextcloud, Plex
5 min	**Q&A**

Prerequisite check: Can you access your group's Proxmox UI?

Terraform + Proxmox

~15 minutes

What is Terraform?

Infrastructure as Code — define your infrastructure in files, not by clicking UIs.

```
Manual:  Click "Create CT" → fill form → repeat for every container
```

```
Terraform:  Write code once → terraform apply → done in 30 seconds
```

Why it matters:

- **Reproducible** — rebuild everything from code if something breaks
- **Version controlled** — track changes in Git
- **Fast** — deploy entire stacks in minutes
- **Documentation** — your code IS your documentation

Step 1: Install Terraform

```
# Linux/WSL
wget -O- https://apt.releases.hashicorp.com/gpg | \
  sudo gpg --dearmor -o /usr/share/keyrings/hashicorp-archive-keyring.gpg
echo "deb [signed-by=/usr/share/keyrings/hashicorp-archive-keyring.gpg] \
  https://apt.releases.hashicorp.com $(lsb_release -cs) main" | \
  sudo tee /etc/apt/sources.list.d/hashicorp.list
sudo apt update && sudo apt install terraform

# macOS
brew install terraform

# Verify
terraform version
```

Step 2: Create Proxmox API Token

In Proxmox Web UI (`https://192.168.10.10X:8006` — use your group's IP):

1. **Datacenter** → **Permissions** → **API Tokens** → **Add**
2. User: `root@pam` · Token ID: `terraform`
3. **Uncheck** "Privilege Separation"
4. **Copy the secret immediately** (shown only once!)

Or via CLI:

```
ssh root@192.168.10.10X  # Replace X with your group number (1/2/3)
pveum user token add root@pam terraform --privsep=0
```

Step 3: Provider Configuration

Create a project directory and these files:

providers.tf

```
terraform {
  required_version = ">= 1.0"
  required_providers {
    proxmox = {
      source  = "bpg/proxmox"
      version = "~> 0.71"
    }
  }
}

provider "proxmox" {
  endpoint = var.proxmox_api_url
  insecure = var.proxmox_tls_insecure
  api_token = "${var.proxmox_api_token_id}=${var.proxmox_api_token_secret}"
}
```

We use the **bpg/proxmox** provider (community-maintained, actively developed).

Step 4: Variables & Init

`terraform.tfvars` (your values — never commit this!)

```
proxmox_api_url          = "https://192.168.10.10X:8006"  # Your group's IP
proxmox_tls_insecure     = true
proxmox_api_token_id     = "root@pam!terraform"
proxmox_api_token_secret = "xxxxxxxx-xxxx-xxxx-xxxx-xxxxxxxxxxxx"
ssh_public_key           = "ssh-ed25519 AAAAC3... you@example.com"
```

Initialize Terraform:

```
terraform init
```

```
Initializing provider plugins...
- Finding bpg/proxmox versions matching "~> 0.71"...
- Installing bpg/proxmox v0.71.0...

Terraform has been successfully initialized!
```

Deploy LXC Container

~15 minutes

The LXC Module

`modules/lxc/main.tf` (reusable container module):

```
resource "proxmox_virtual_environment_container" "container" {
  node_name    = var.target_node
  vm_id        = var.vmid
  description  = var.description
  unprivileged = var.unprivileged
  started      = true

  operating_system {
    template_file_id = var.ostemplate
    type             = "ubuntu"
  }

  cpu    { cores     = var.cores }
  memory { dedicated = var.memory }
  disk   { datastore_id = var.rootfs_storage
           size         = parseint(replace(var.rootfs_size, "G", ""), 10) }

  network_interface { name = "eth0"; bridge = "vmbr0" }
  initialization {
    hostname  = var.hostname
    ip_config { ipv4 { address = var.network_ip; gateway = var.network_gateway } }
  }
  features { nesting = var.features_nesting; keyctl = var.features_keyctl }
}
```

Deploy the Docker Host

`main.tf` — using our LXC module:

```
module "docker_host" {
  source = "./modules/lxc"

  target_node  = "prx01"
  hostname     = "docker-host"
  description  = "Docker host for compose services"
  vmid         = 100
  ostemplate   = "local:vztmpl/ubuntu-24.04-standard_24.04-2_amd64.tar.zst"
  unprivileged = false    # Privileged for Docker

  cores        = 4
  memory       = 8192     # 8GB RAM
  rootfs_size  = "64G"

  network_ip      = "192.168.10.50/24"
  network_gateway = "192.168.10.1"

  features_nesting = true  # Required for Docker
  features_keyctl  = true  # Required for Docker
}
```

Plan & Apply

```
# Preview what Terraform will create
terraform plan
```

```
  + module.docker_host.proxmox_virtual_environment_container.container
      node_name:    "prx01"
      vm_id:        100
      hostname:     "docker-host"
      cores:        4
      memory:       8192

Plan: 1 to add, 0 to change, 0 to destroy.
```

```
# Deploy!
terraform apply
# Type "yes" to confirm
```

Check Proxmox UI → container appears in ~30 seconds

```
# SSH into your new container
ssh root@192.168.10.50
```

Tailscale VPN

~10 minutes

What is Tailscale?

Mesh VPN built on WireGuard — access your homelab from anywhere.

```
Without Tailscale:
  Home only → can access 192.168.10.50 ✓
  Coffee shop → can't reach homelab ✗

With Tailscale:
  Anywhere → access via 100.x.x.x ✓
  Encrypted, peer-to-peer, no port forwarding needed
```

Install on Docker Host:

```
ssh root@192.168.10.50

# Install
curl -fsSL https://tailscale.com/install.sh | sh

# Connect (opens browser to authenticate)
tailscale up

# Verify
tailscale status
```

Subnet Routing

Expose your entire homelab network through Tailscale:

```
# Advertise your management network
tailscale up --advertise-routes=192.168.10.0/24

# Enable IP forwarding
echo 'net.ipv4.ip_forward = 1' | tee -a /etc/sysctl.conf
sysctl -p
```

In Tailscale Admin Console (`login.tailscale.com/admin`):

1. Find your docker-host machine
2. Click **"..."** → **Edit route settings**
3. **Approve** the 192.168.10.0/24 route

Now from anywhere: access Proxmox UI, SSH to any node, reach any service on VLAN 10

Docker Compose Services

~25 minutes — the main event

Install Docker

```
ssh root@192.168.10.50

# Install Docker
curl -fsSL https://get.docker.com -o get-docker.sh
sh get-docker.sh

# Install Compose plugin
apt install -y docker-compose-plugin

# Verify
docker --version
docker compose version

# Create working directory
mkdir -p /docker && cd /docker
```

Our Service Stack

Service	What it Does	Port
AdGuard Home	Network-wide DNS ad blocking	`:53` `:8080`
Nginx Proxy Manager	Reverse proxy with SSL certs	`:80` `:443` `:81`
Portainer	Docker management web UI	`:9000`
Vaultwarden	Self-hosted Bitwarden passwords	`:8090`
Cloudflare Tunnel	Expose services to internet securely	—
WordPress + MySQL	Self-hosted website/blog	`:8888`

All defined in a single `docker-compose.yml` file.
All running on one LXC container.

docker-compose.yml (1/2)

```yaml
services:
  adguard:
    image: adguard/adguardhome:latest
    container_name: adguardhome
    restart: unless-stopped
    ports:
      - "53:53/tcp"
      - "53:53/udp"
      - "3000:3000/tcp"
      - "8080:80/tcp"
    volumes:
      - ./adguard/work:/opt/adguardhome/work
      - ./adguard/conf:/opt/adguardhome/conf

  nginx-proxy-manager:
    image: jc21/nginx-proxy-manager:latest
    container_name: nginx-proxy-manager
    restart: unless-stopped
    ports: ["80:80", "443:443", "81:81"]
    volumes:
      - ./proxy/data:/data
      - ./proxy/letsencrypt:/etc/letsencrypt

  portainer:
    image: portainer/portainer-ce:latest
    container_name: portainer
    restart: unless-stopped
    ports: ["9000:9000", "9443:9443"]
    volumes:
      - /var/run/docker.sock:/var/run/docker.sock
      - ./portainer/data:/data
```

docker-compose.yml (2/2)

```
vaultwarden:
  image: vaultwarden/server:latest
  container_name: vaultwarden
  restart: unless-stopped
  ports: ["8090:80"]
  volumes:
    - ./vaultwarden/data:/data
  environment:
    - SIGNUPS_ALLOWED=false
    - ADMIN_TOKEN=change-this-to-a-secure-token

cloudflared:
  image: cloudflare/cloudflared:latest
  container_name: cloudflared
  restart: unless-stopped
  command: tunnel --no-autoupdate run
  environment:
    - TUNNEL_TOKEN=your-cloudflare-tunnel-token

wordpress-db:
  image: mysql:8.0
  container_name: wordpress-db
  restart: unless-stopped
  volumes:
    - ./wordpress/db:/var/lib/mysql
  environment:
    - MYSQL_ROOT_PASSWORD=rootpass
    - MYSQL_DATABASE=wordpress
    - MYSQL_USER=wp
    - MYSQL_PASSWORD=wppass123

wordpress:
  image: wordpress:latest
  container_name: wordpress
  restart: unless-stopped
  depends_on: [wordpress-db]
  ports: ["8888:80"]
  volumes:
    - ./wordpress/html:/var/www/html
  environment:
    - WORDPRESS_DB_HOST=wordpress-db
    - WORDPRESS_DB_USER=wp
    - WORDPRESS_DB_PASSWORD=wppass123
    - WORDPRESS_DB_NAME=wordpress
    - WORDPRESS_CONFIG_EXTRA=
        define('WP_HOME','http://192.168.10.50:8888');
        define('WP_SITEURL','http://192.168.10.50:8888');
```

Deploy & Verify

```
cd /docker

# Deploy all services
docker compose up -d

# Check status
docker compose ps
```

```
NAME                 STATUS
adguardhome          Up
nginx-proxy-manager  Up
portainer            Up
vaultwarden          Up
cloudflared          Up
wordpress-db         Up
wordpress            Up
```

Useful commands:

```
docker compose logs -f          # Follow all logs
docker compose logs adguard     # Specific service
docker compose restart          # Restart all
docker compose pull && docker compose up -d  # Update all
```

Configure AdGuard Home

1. Open `http://192.168.10.50:3000` (first-time setup wizard)
2. Set admin interface to port **8080** (or keep 3000)
3. Set DNS port to **53**
4. Create admin username & password
5. Set upstream DNS: `1.1.1.1` and `8.8.8.8`
6. Enable default filter lists

After setup, access at: `http://192.168.10.50:8080`

```
DNS flow:  All devices → AdGuard (192.168.10.50:53) → 1.1.1.1
                                    ↓
                            Ads & trackers blocked
```

Service Access

Service	URL	Default Credentials
AdGuard Home	`http://192.168.10.50:8080`	Set during setup
Nginx Proxy Mgr	`http://192.168.10.50:81`	`admin@example.com` / `changeme`
Portainer	`http://192.168.10.50:9000`	Set on first visit
Vaultwarden	`http://192.168.10.50:8090`	Create account
WordPress	`http://192.168.10.50:8888`	Set during setup

Via Tailscale (remote): Replace `192.168.10.50` with Tailscale IP (`100.x.x.x`)

Additional Services & Wrap-up

~5 minutes

What Else Can You Deploy?

Same pattern — add to `docker-compose.yml`, run `docker compose up -d`

Service	What	Effort
Immich	Google Photos replacement (AI-powered)	Medium
Nextcloud	Cloud storage (Dropbox/Drive alternative)	Medium
Plex / Jellyfin	Media server for movies & TV	Easy
Uptime Kuma	Service monitoring dashboard	Easy
Gitea	Self-hosted GitHub	Easy
Home Assistant	Smart home automation	Medium
Grafana + Prometheus	Metrics & monitoring	Advanced

Full setup guides in `docs/session-2/05-additional-services.md`

What You Built Today

- **Terraform-managed** LXC container on Proxmox
- **7-service** Docker Compose stack
- **Tailscale VPN** for remote access
- **AdGuard Home** — network-wide ad blocking
- **Vaultwarden** — self-hosted password manager
- **Nginx Proxy Manager** — reverse proxy with SSL
- **Portainer, Cloudflare Tunnel, WordPress**

Next Steps & Resources

Keep building:

1. Back up your configs (`docker compose` files + Terraform state)
2. Add more services (Immich, Nextcloud, Plex)
3. Set up monitoring (Uptime Kuma → Grafana)
4. Explore Kubernetes (K3s on Proxmox)

Community:

- **r/homelab** — 500k+ members, hardware & setup discussions
- **r/selfhosted** — service recommendations & guides
- **Proxmox Forums** — official support
- **YouTube** — TechnoTim, Jeff Geerling, DB Tech

Q&A

5 minutes

Questions?

Common questions:

- **"How do I back this up?"**
  Proxmox snapshots + `tar` your Docker volumes + Git your configs
- **"Can I expose services to the internet?"**
  Yes — Cloudflare Tunnel (secure) or Nginx Proxy Manager + port forwarding
- **"What if something breaks?"**
  `terraform apply` recreates containers, `docker compose up -d` restarts services
- **"Can I run this on a single machine?"**
  Absolutely — everything works the same on a single node

Thank you! Happy homelabbing!