HBTECHSOLUTIONS

Ubuntu Server & Infrastructure Deployment Toolkit

From a fresh Ubuntu installation to a validated Docker, Portainer, Pi-hole, and WireGuard lab deployment.

VERSION 2.0.0 | Revised November 25, 2026

Validated platform: Ubuntu Server 26.04.1 LTS, AMD64 (Proxmox VM)

Release: https://github.com/iireddrag0nii/scripts/releases/tag/v2.0.0

Scope: Guided homelab / authorized infrastructure deployment; not an unattended production installer.

> Read before installation. Confirm your selected target disk and ensure backups exist. This new edition supersedes the original manual Docker/Portainer steps. Do not combine Docker Snap, Ubuntu docker.io, and the toolkit-managed Docker installation on the same fresh host.

How to use this guide

This edition preserves the Ubuntu Server setup sequence from the October 2026 guide and replaces the older manual Docker and Portainer workflow with the tested HBTECHSOLUTIONS Infrastructure Deployment Toolkit v2.0.0. It also adds Pi-hole, WireGuard, lifecycle validation, and troubleshooting.

Part	What you will do
1	Install Ubuntu Server and enable OpenSSH
2	Update the operating system and clone the toolkit
3	Run preflight and install Docker safely
4	Deploy Portainer, Pi-hole, and WireGuard
5	Verify persistence, DNS, and VPN security
6	Troubleshoot and maintain your installation

What you need

- A dedicated test computer or VM, a supported Ubuntu Server installation ISO, and network connectivity.
- At least 2 vCPUs and 2 GB RAM for a small lab; allocate additional memory, CPU, and storage for actual workloads.
- Enough free space for Ubuntu, container images, and persistent volumes. The tested VM had a 15 GB root filesystem; larger disks are advisable.
- A terminal with sudo access, Git, and a machine from which you can SSH to the server.
- A trusted Windows or Linux WireGuard client if you plan to validate the optional VPN module.

> Platform scope. Ubuntu 26.04 AMD64 received full clean-host functional testing. Ubuntu 22.04 was used in earlier individual-module testing. The toolkit has architecture/OS eligibility checks, but equivalent clean-host deployment on ARM64 was not demonstrated in this release.

1. Install Ubuntu Server

Prepare and boot the installer

- For a VM: create a new virtual machine, attach the Ubuntu Server ISO, and boot it.
- For physical hardware: download Ubuntu Server from the official Ubuntu website, use a tool such as balenaEtcher to write it to a USB drive, and boot from that drive.

- Choose Try or Install Ubuntu Server, then select language, keyboard layout, and the normal Ubuntu Server profile.
- Configure the NIC (DHCP is appropriate for initial testing). Use a stable address or DHCP reservation if clients need a consistent endpoint.
- Review proxy and package mirror screens. For storage, carefully verify the destination disk before accepting any erase/use-entire-disk option.
- Create a named local administrator account and set a strong unique password. Ubuntu Pro can be skipped for a homelab.
- Select Install OpenSSH server if remote administration is required. Do not select the Docker Snap in Featured Server Snaps.
- Complete installation, reboot, remove installation media if necessary, and sign in.

> Historical images: Screenshots in the appendix were retained from the earlier Ubuntu installation walkthrough. Installer appearance and navigation may differ in Ubuntu 26.04; use the current on-screen prompts as the source of truth.

Update the operating system

```
sudo apt update
sudo apt upgrade -y
sudo reboot
```

Reconnect after the reboot. Verify the distribution, address, and root filesystem before continuing:

```
cat /etc/os-release
ip -4 addr
df -h /
```

2. Obtain the toolkit and assess the host

Install Git and clone the public repository. Run commands from the repository root:

```
sudo apt install -y git
git clone https://github.com/iireddrag0nii/scripts.git
cd scripts
git checkout v2.0.0
```

The release tag provides a reproducible starting point. Check the branch/tag and run the read-only host assessment:

```
git describe --tags --exact-match
./scripts/preflight.sh
```

The preflight identifies Ubuntu and CPU architecture, checks Docker/Compose presence, disk capacity, and publishes a read-only port inventory. It does not install packages or deploy containers.

Default toolkit bindings

Service	Published host binding	Purpose
Portainer	127.0.0.1:9443/TCP	HTTPS dashboard
Pi-hole	127.0.0.1:5353/TCP + UDP	DNS queries
Pi-hole	127.0.0.1:8088/TCP	Web management
WireGuard	Selected LAN IP:51821/UDP	VPN endpoint

A port warning requires investigation. Port ownership and existing Docker projects are inspected by deployment-specific safeguards before installation.

> Never run the toolkit blindly on a server with important existing containers. The original tests intentionally used isolated lab machines. The Toolkit protects against many conflicts, but an administrator must review the host and backup plan.

3. Install Docker Engine with safeguards

Toolkit v2.0 uses the guarded Docker installation workflow rather than the old manual `apt install docker.io` / Docker Snap combination. First review the plan and dry run:

```
./scripts/install-docker.sh --plan
./scripts/install-docker.sh --dry-run
```

For a fresh, eligible host, the decision should be FRESH_CANDIDATE. Existing or ambiguous Docker, Podman, rootless, custom-endpoint, Snap, or containerd installations can produce MANUAL_REVIEW, and a compatible existing installation may produce SKIP_INSTALL. Do not force the install past a manual-review result.

Interactive installation

```
sudo ./toolkit.sh
```

Choose 2 - Install Docker Engine. Review the detected host state and enter the exact requested confirmation phrase (`INSTALL DOCKER ON LAB VM` in the validated release). This changes system packages; use only on the approved test host.

Verify Docker after installation

```
sudo systemctl is-active docker
sudo docker version
sudo docker compose version
sudo docker buildx version
sudo docker run --rm hello-world
sudo ./scripts/install-docker.sh --plan
```

The final elevated planning command should report SKIP_INSTALL on an already compatible system. The toolkit intentionally does not add users to the Docker group automatically; root-equivalent Docker access requires explicit administrative consideration.

4. Deploy Portainer

Launch the menu with `sudo ./toolkit.sh`, then choose 3 - Deploy Portainer. Confirm the deployment after reviewing the security notes. Portainer is deployed with a pinned image digest and persistent named data volume. Management HTTPS is bound to localhost only.

Verify the management endpoint

```
sudo docker compose -f compose/portainer/compose.yaml ps
curl -k -I https://127.0.0.1:9443
```

The clean-host test received HTTP 200 over HTTPS. The first-run UI may use a self-signed TLS certificate; investigate the certificate before relying on the endpoint outside a trusted lab.

Access Portainer from a Windows administration PC

Establish an SSH tunnel from PowerShell, substituting your own server IP and login:

```
ssh -N -L 9443:127.0.0.1:9443 USER@SERVER_IP
```

Open https://localhost:9443 on the administration PC. Complete the administrator setup. If Portainer requires its setup token, retrieve it on the Ubuntu server and keep the token private:

```
sudo docker logs hbtech-portainer-portainer-1 2>&1 | grep 'setup_token='
```

Do not paste setup tokens or passwords into GitHub issues, screenshots, or public logs. The Portainer deployment grants the management container access to the Docker socket, which is highly privileged; keep the management interface restricted.

Persistence and repeat-run verification

```
sudo docker compose -f compose/portainer/compose.yaml \
  up -d --force-recreate
```

Confirm the administrator configuration persists after recreation. Selecting menu option 3 again should return [SKIP] instead of overwriting the project.

5. Deploy Pi-hole

Run `sudo ./toolkit.sh` and select 4 - Deploy Pi-hole. The toolkit publishes DNS on localhost TCP/UDP 5353 and the web interface on localhost TCP 8088. It does not automatically take over the Ubuntu host DNS resolver or router DNS.

Check health and test DNS

```
sudo docker inspect -f '{{.State.Health.Status}}' \
  hbtech-pihole-pihole-1
dig @127.0.0.1 -p 5353 example.com
dig +tcp @127.0.0.1 -p 5353 example.com
```

Expect healthy from the container and a successful DNS response over both UDP and TCP. To test blocking with a disposable hostname:

```
sudo docker exec hbtech-pihole-pihole-1 \
  pihole deny hbtech-block-test.example.com
dig @127.0.0.1 -p 5353 hbtech-block-test.example.com
```

The tested response was `0.0.0.0`, with DNS blocking metadata. Remove the test entry when finished:

```
sudo docker exec hbtech-pihole-pihole-1 \
  pihole deny remove hbtech-block-test.example.com
```

Open the Pi-hole dashboard securely

```
ssh -N -L 8088:127.0.0.1:8088 USER@SERVER_IP
```

From the administration PC, visit http://localhost:8088/admin/. Use the authentication instructions supplied by the current Pi-hole release; do not publish administrative passwords.

Confirm persistence

```
sudo docker compose -f compose/pihole/compose.yaml \
  up -d --force-recreate
```

Verify DNS and configured list entries remain available, then confirm a repeat run of option 4 returns [SKIP]. A test host does not become a network-wide DNS server unless you intentionally design and approve that change.

6. Deploy restricted WireGuard

Toolkit v2.0 uses a LinuxServer.io WireGuard image rather than the earlier wg-easy UI. The default VPN subnet is 10.77.77.0/24, with client routing limited to 10.77.77.1/32. LAN forwarding and unrestricted client access are intentionally disabled.

Review prerequisites

```
sudo modprobe wireguard
ip -4 addr
ip -4 route
sudo ss -lun | grep 51821 || true
```

Confirm that the planned VPN subnet does not overlap any existing local, VPN, or Docker network. Ensure the selected server LAN IP belongs to the host and that UDP 51821 is not already in use.

Deploy using menu option 5

```
sudo ./toolkit.sh
```

Choose 5 - Deploy WireGuard; supply the approved host LAN IPv4 address and confirm with the exact phrase `DEPLOY WIREGUARD`. The installer creates `compose/wireguard/.env` with permission mode 600 and never replaces an existing file or symlink.

Inspect the running service

```
sudo docker compose \
  --env-file compose/wireguard/.env \
  -f compose/wireguard/compose.yaml ps
sudo docker exec hbtech-wireguard-wireguard-1 wg show wg0 interfaces
sudo docker exec hbtech-wireguard-wireguard-1 iptables -S FORWARD
```

Expected firewall behavior includes DROP rules for forwarded traffic entering or leaving `wg0`. The service may need several seconds after container startup before the tunnel interface appears.

7. WireGuard client and isolation verification

Safely obtain a client profile

WireGuard generates peer files in its persistent Docker volume. Copy a profile to a private temporary location on the lab server:

```
sudo docker cp \
  hbtech-wireguard-wireguard-1:/config/peer1/peer1.conf \
  "$HOME/wg-v2testing.conf"
sudo chown "$(id -un):$(id -gn)" "$HOME/wg-v2testing.conf"
chmod 600 "$HOME/wg-v2testing.conf"
```

Substitute your own non-sensitive local filename. From Windows PowerShell, securely copy the profile:

```
scp USER@SERVER_IP:~/wg-v2testing.conf
"$env:USERPROFILE\Downloads\wg-v2testing.conf"
```

Import it into the official WireGuard client. Confirm the endpoint matches the approved server LAN IP on UDP 51821 and that peer AllowedIPs remain `10.77.77.1/32`. Delete the temporary export from the Ubuntu user home directory after successful import; retain the Docker volume.

```
rm -f "$HOME/wg-v2testing.conf"
```

Validate connectivity

Activate the client and test the VPN server interface from Windows:

```
ping 10.77.77.1
```

Check server-side handshake metadata without printing private keys:

```
sudo docker exec hbtech-wireguard-wireguard-1 \
  wg show wg0 latest-handshakes
sudo docker exec hbtech-wireguard-wireguard-1 \
```

```
wg show wg0 transfer
```

Validate isolation safely

In a disposable lab only, briefly include a single confirmed Docker-network destination in the client AllowedIPs, try reaching that address, and compare the forwarding DROP packet counters before and after. The clean-host test saw eight packets dropped while direct tunnel ping remained successful. Restore `10.77.77.1/32` immediately after testing.

```
sudo docker exec hbtech-wireguard-wireguard-1 \
  iptables -vnL FORWARD
```

> Do not generalize the test address. The original lab tested 172.18.0.1/32, but Docker network addresses can change. A failed ping alone is not proof of isolation; confirm that the appropriate DROP counter increases.

8. Validate persistence and safe repeat runs

Recreating a container with its existing named volume should preserve service state. This procedure is for the authorized lab, not for arbitrary third-party production workloads.

WireGuard persistence

Record the existing server public key, deactivate the Windows client, then recreate the container:

```
sudo docker exec hbtech-wireguard-wireguard-1 \
  wg show wg0 public-key
sudo docker compose \
  --env-file compose/wireguard/.env \
  -f compose/wireguard/compose.yaml \
  up -d --force-recreate
```

After initialization, check that the same public key is returned, the INPUT/FORWARD restrictions exist, and the original client profile can reconnect and ping 10.77.77.1. Repeat menu option 5: it should return [SKIP].

Final status snapshot

```
sudo docker ps --format 'table {{.Names}}\t{{.Status}}\t{{.Ports}}'
df -h /
git status --short
```

In the clean-host acceptance test, all three service containers were running, Pi-hole was healthy, and the 15 GB test VM had approximately 7.7 GB free after the installations.

9. Recovery, security and maintenance

Interrupted initial WireGuard installation

If deployment creates `compose/wireguard/.env` but Compose fails, the toolkit intentionally blocks a blind overwrite on the next attempt. Do not delete the .env file or remove volumes as a first response.

```
sudo stat -c '%a %U:%G %n' compose/wireguard/.env
sudo docker info >/dev/null
sudo docker compose --env-file compose/wireguard/.env \
  -f compose/wireguard/compose.yaml config --quiet
sudo docker compose --env-file compose/wireguard/.env \
  -f compose/wireguard/compose.yaml ps -a
```

Review service logs and network conflicts, then retry only after understanding the cause:

```
sudo docker compose --env-file compose/wireguard/.env \
  -f compose/wireguard/compose.yaml logs --tail=40
sudo docker compose --env-file compose/wireguard/.env \
  -f compose/wireguard/compose.yaml up -d
```

Do not paste logs if they expose secrets. The procedure is documented, but deliberately interrupting a fresh install and proving recovery was not part of the final live acceptance test.

Security principles

- Protect Docker and Portainer administrator access: the Docker socket is privileged.
- Use SSH tunneling or a trusted approved management path for localhost-bound web interfaces.
- Keep .env files, WireGuard peer configurations, keys, token output, and passwords out of Git and published screenshots.
- Back up persistent volumes before maintenance or image changes. Do not casually delete named Docker volumes.
- Review the pinned image digest, release notes, and service guides before updating.
- Do not enable router forwarding, full-tunnel routes, or LAN access unless intentionally designed and secured.

Read the canonical repository documentation

README.md - project overview; docs/INSTALLATION.md - current deployment path; docs/PORTAINER.md, docs/PIHOLE.md, and docs/WIREGUARD.md - component operations; SECURITY.md - boundaries; docs/RELEASE-VALIDATION.md - acceptance evidence and limitations.

The toolkit is a release artifact, not a managed service. Support status, image pinning, and user-interface behavior may change over time; consult the tagged repository documentation for the installed version.

10. Quick acceptance checklist

- [ ] Ubuntu Server installed on the intended disk; SSH access confirmed.
- [ ] Operating system updates completed; no conflicting Docker installation method used.
- [ ] Repository checked out at v2.0.0; preflight warnings reviewed.
- [ ] Docker Engine, Compose, and hello-world verified.
- [ ] Portainer dashboard reachable only through approved access; settings persist.
- [ ] Pi-hole healthy; UDP/TCP DNS and test blocking verified.
- [ ] WireGuard handshake and 10.77.77.1 tunnel ping verified.
- [ ] WireGuard DROP rules confirmed; any temporary client routes reverted.
- [ ] Docker volumes retained after recreations; repeat deployments return SKIP.
- [ ] Secrets removed from temporary locations; logs and screenshots sanitized.

Official references

Ubuntu Server: https://documentation.ubuntu.com/server/

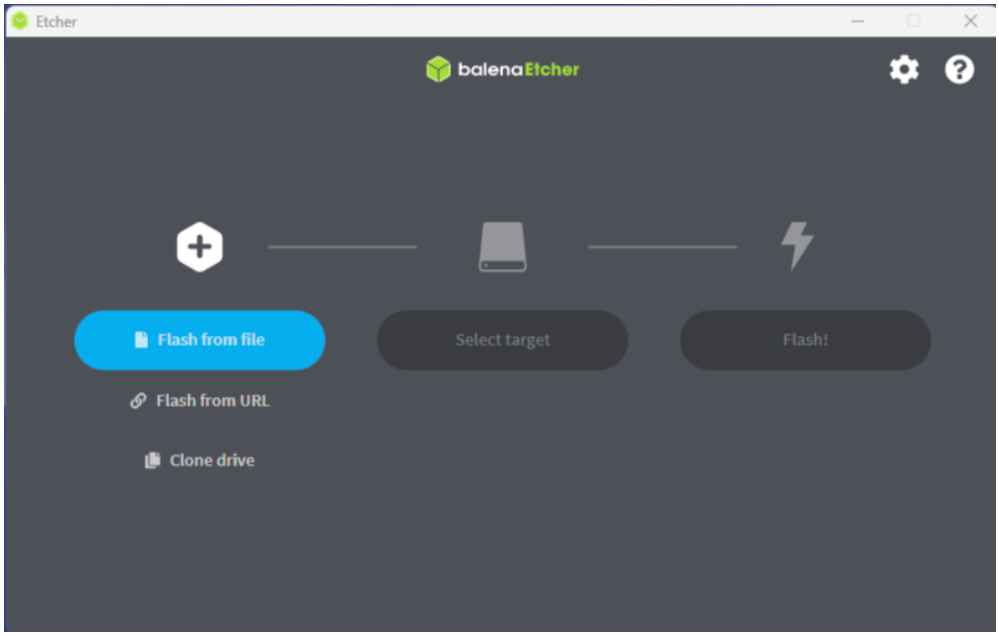
Docker Engine for Ubuntu: https://docs.docker.com/engine/install/ubuntu/

Portainer: https://docs.portainer.io/start/install-ce/server/docker/linux

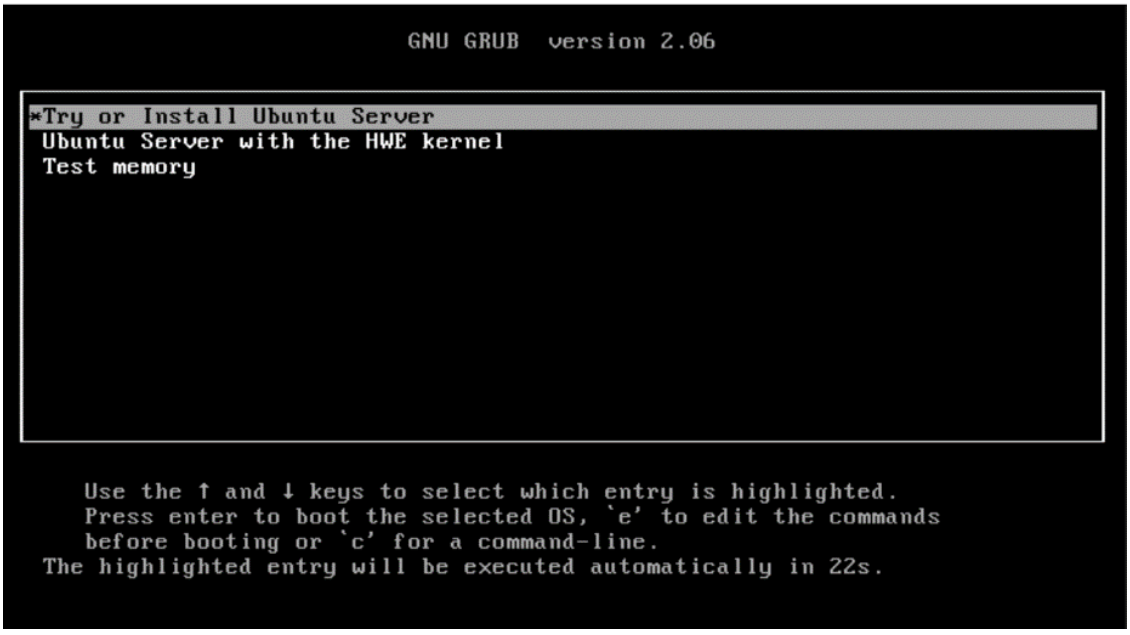
Toolkit release: https://github.com/iireddrag0nii/scripts/releases/tag/v2.0.0

Appendix - Original installer screenshot references

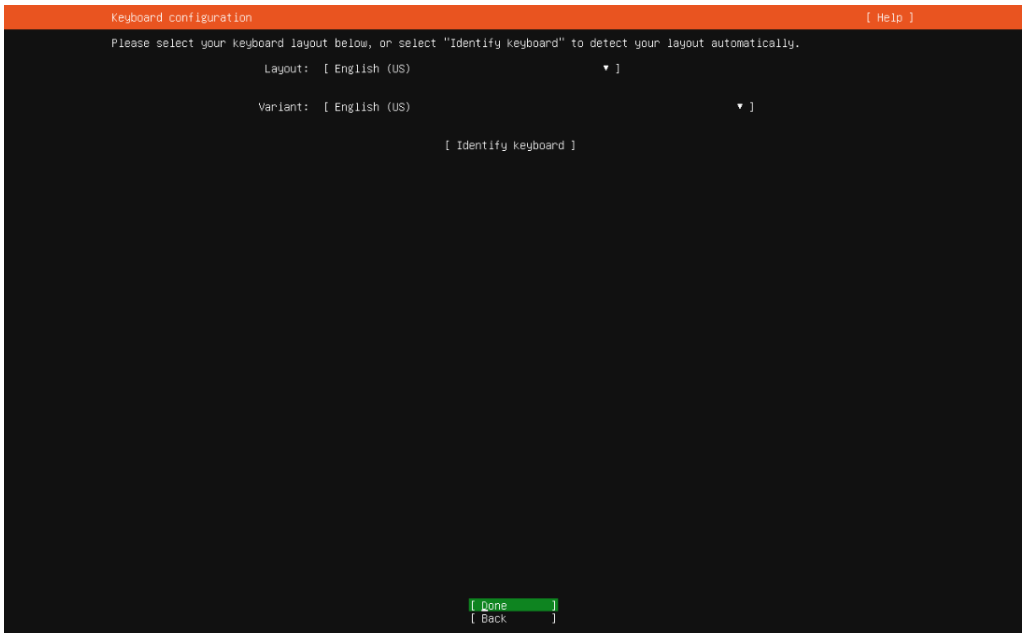
Selected original annotated Ubuntu Server installer screenshots are retained here for learners following the earlier edition. These illustrate the general workflow but are not a substitute for current Ubuntu 26.04 prompts. Outdated Docker and Portainer screenshots are intentionally not repeated.



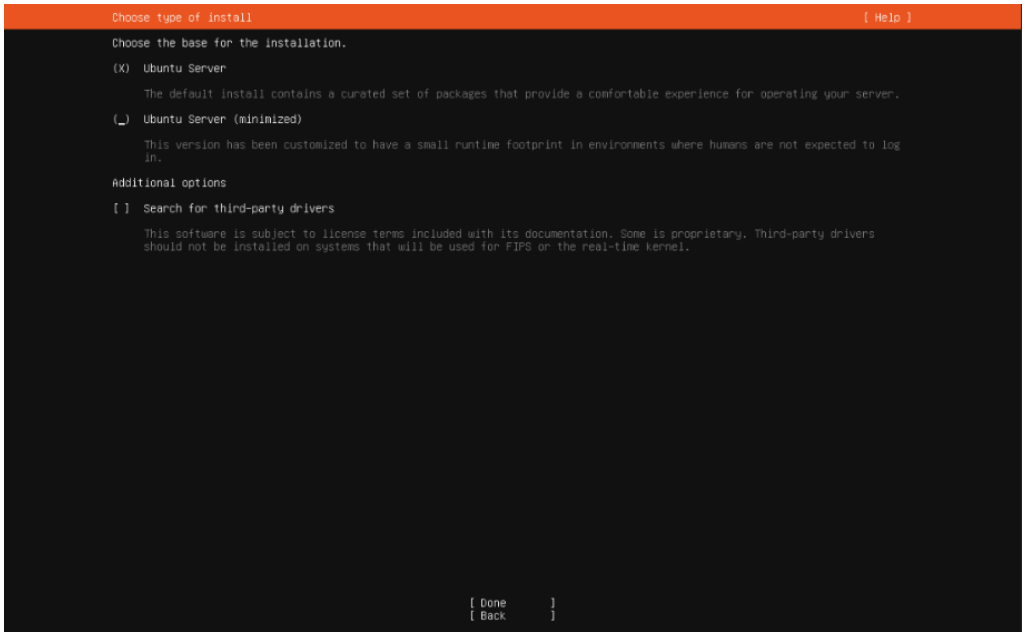
Preparing Ubuntu installation media | Original guide page 2



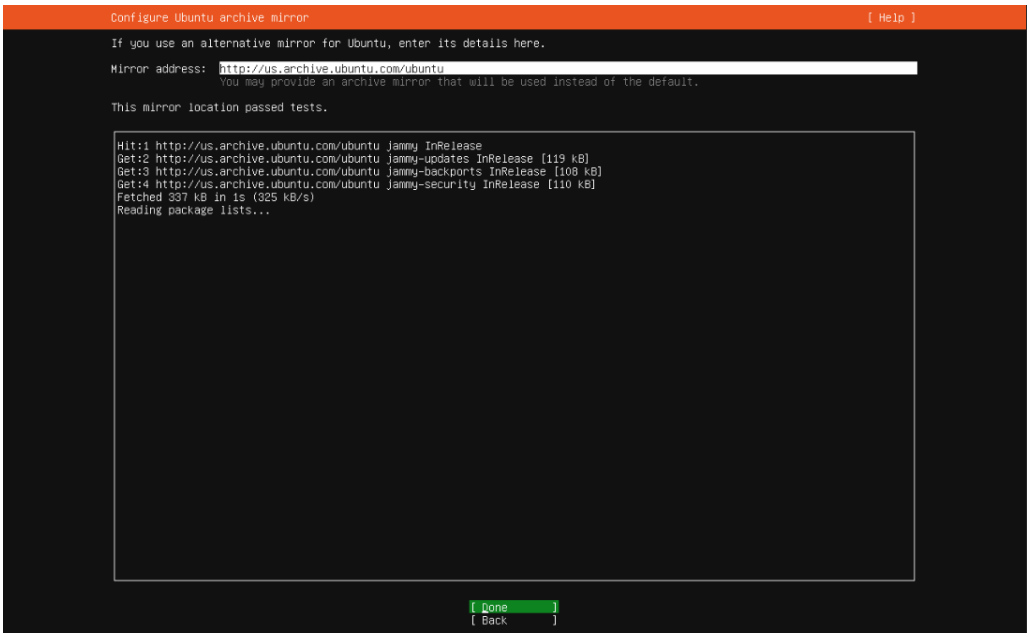
Ubuntu Server boot menu | Original guide page 3



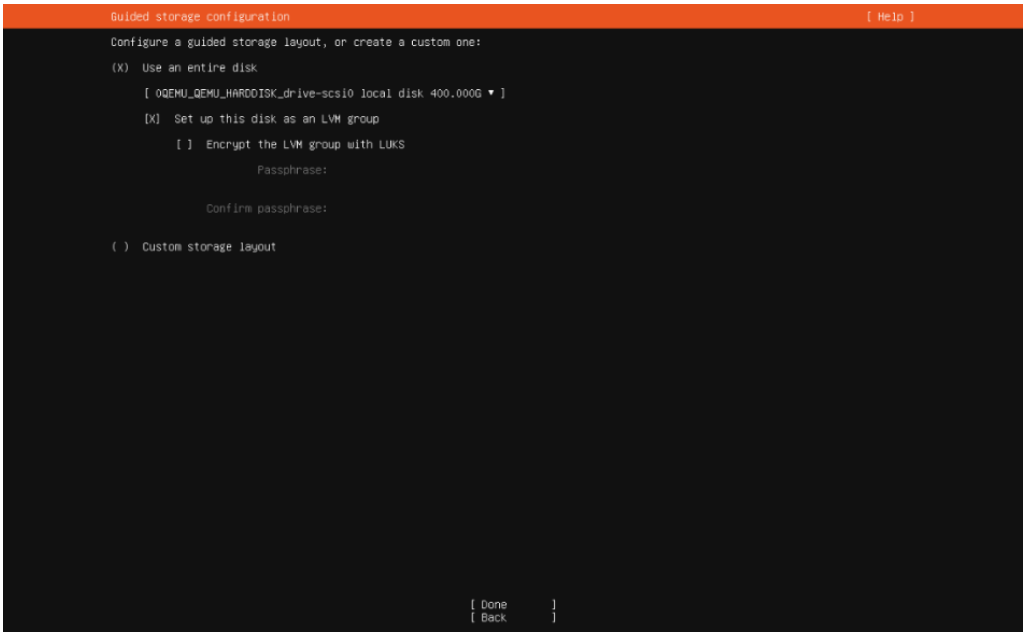
Installer language and keyboard | Original guide page 4



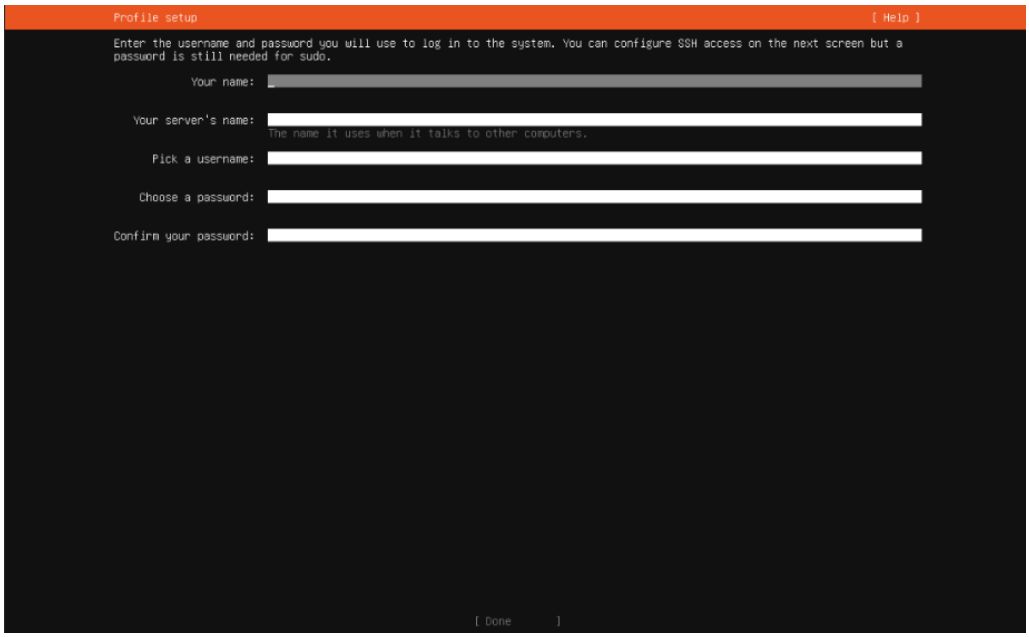
Network interface configuration | Original guide page 5



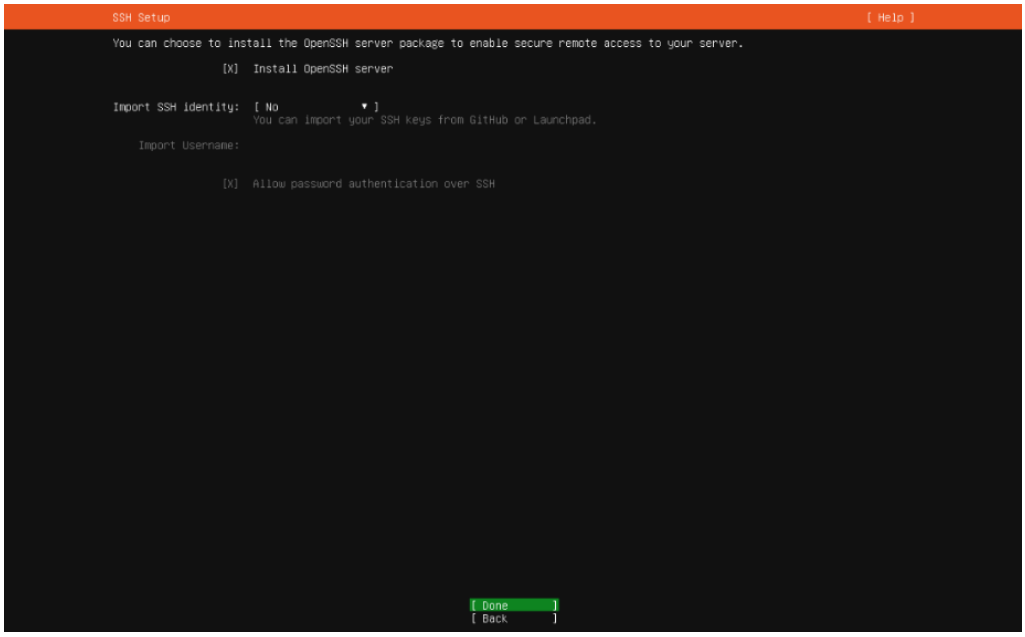
Storage selection - verify the target disk | Original guide page 6



Installer storage and identity prompts | Original guide page 7



Ubuntu profile creation | Original guide page 9



Initial system update | Original guide page 10