lab.bitsmasher.net Manual

franklin

August 2026

Contents

Chapter 1

Introduction

1.1 Purpose and Scope

This manual documents the lab.bitsmasher.net infrastructure – network, hardware, services, and procedures. It is intended as a reference for current operations and a foundation for onboarding new administrators.

Chapter 2

Hardware Inventory

2.1 Desk Setup

Franklin's desk hosts chonk, the primary gateway workstation. Sly's desk hosts a GAMING Windows PC at the main office near music.lan's monitor. Both have access to all lab infrastructure.

2.2 Hardware Inventory

2.2.1 Desk Setup

- **Franklin's desk**: chonk (10.10.8.60) – GAMING Windows PC, primary workstation, Gateway host running OpenClaw
- **Sly's desk**: GAMING Windows computer (Twitch: `https://www.twitch.tv/s1y_b0rg`) – at main office near music.lan monitor
- Both sit at the main office where music.lan's monitor is displayed

2.2.2 Server and Compute Hosts

Core Infrastructure (Debian 12)

Host	IP	User	Role
chonk	10.10.8.60	franklin/openclaw	Gateway host, primary workstation + inference
stargate	10.10.16.66	openclaw	Ansible workspace, build host, DNS master (node3), NFS server
skynet	10.10.16.10	openclaw/franklin	Minecraft server, IP-direct login
time	10.10.12.2	openclaw/franklin	Stratum-1 NTP server with GPS reference
wonderland	178.62.60.55	franklin/openclaw	Public cloud VM (Debian 12, 6.1 kernel), web host
music.lan	192.168.86.38	root	Desktop machine via skynet route
node3	10.10.12.3	openclaw	**Authoritative BIND9 DNS master** (post-server1 migration)

Jetson Cluster (~87 days uptime)

All Jetsons run Ubuntu 18.04 with ed25519 key auth (openclaw permanent, franklin has password `123` as fallback – pending key-only migration):

Host	IP	Status
node900	10.10.12.90	Online, key auth; home dir from stargate:/mnt/clusterfs2
node901	10.10.12.91	Online, key auth; home dir from stargate:/mnt/clusterfs2
node903	10.10.12.93	Online, key auth; home dir from stargate:/mnt/clusterfs2
node902	10.10.12.92	Off / no route to host (likely powered down)

NFS note: Jetson home directories are NFS-mounted from stargate:/mnt/clusterfs2. StrictModes permissions apply (home 0750, .ssh 0700, authorized_keys 0600).

KDC and Directory Services

Host	IP	Status
odroid-c1 / kdc1	10.10.12.254	Online (242d uptime), root key auth, KDC for lab.bitsmasher.net Kerberos realm
ldap/bbb1	10.10.13.1	SSH reachable as root; slapd failed Dec 2025 (TLS cert issue)

2.2.3 Historical / Decommissioned Hosts

- **server1 (10.10.12.12)**: Former DNS master – decommissioned August 2026; role replaced by node3
- **snowy**: Hard disk physically removed; /mnt/snowy now just a raw mounted drive (not SSH-accessible)
- **blowfish**: Previously at 10.10.12.15, reassigned to 10.10.14.85. Key not authorized yet – unreachable from chonk but port 22 is open. Expected role: database host with OpenBSD
- **node902**: Powered off; no route to host from any known peer

2.2.4 Planned Hardware Expansion

- blowfish as the primary database server (OpenBSD planned)
- Full ed25519 key rollout for node900–node903 to remove password fallback
- k3s_server and k3s_agent roles for Kubernetes across node infrastructure

Chapter 3

Network Infrastructure

3.1 Addressing and Subnets

The lab spans multiple private subnets organized by function, from the main office (10.10.8.0/21) to the backbone (10.10.16.0/) and specialized service zones.

3.2 Network Infrastructure

3.2.1 Addressing Scheme

The lab uses private IPv4 addressing across multiple subnets, organized by function:

Subnet	Allocation	Purpose
10.10.8.0/21	chonk, chonk-wifi, music.lan (via skynet)	Main office / desk area
10.10.12.0/	time, node90x, odroid-c1, blowfish, ns1	Core infrastructure
10.10.13.0/	ldap/bbb1	Directory services (moved to working hosts)
10.10.14.0/	blowfish (reassigned)	New allocation for blowfish
10.10.15.0/	femputer	Guest / temporary hosts
10.10.16.0/	stargate, skynet, edge-t	Lab backbone / staging

Note: The full subnet masks vary by role and DHCP scope configuration on the Dream Machine (DHCP server).

3.2.2 DHCP

The `dhcp` Ansible role manages the DHCP server configuration stored at `roles/dhcp/files/dhcpd.conf`. Key reserved hosts include:

- **chonk**: 10.10.8.60, MAC fc:9d:05:01:27:02 – Franklin's desk
- **chonk-wifi**: 10.10.8.61 – wireless client
- **chonk-ten-gig-1**: 10.10.8.?? – dedicated ten-gig link
- **blowfish**: 10.10.8.15 / 10.10.12.15, MAC 98:b7:85:21:ad:77 – database host
- **femputer**: 10.10.15.1, MAC d8:bb:c1:af:21:17 – temporary/guest

3.2.3 DNS

The DNS infrastructure is managed by the `dns` Ansible role (BIND/named). Zones maintained include:

- Forward zone: db.home.lab
- Reverse zones for each subnet
- A records and PTR records for all lab hosts

DNS role molecule testing confirms BIND packages install correctly and named.conf.options are written. The DNS server (ns1) has been offline during recent audits, which explains why many hosts in the 10.10.x.0/24 range cannot resolve – ns1 is a single point of failure for lab DNS resolution.

3.2.4 Firewall and Access Control

- The Dream Machine provides built-in DHCP and handles basic routing between subnets
- odroid-c1 serves as the KDC and likely has pf firewall rules managing cross-subnet Kerberos traffic
- OpenBSD hosts (blowfish planned) use pf for host-level firewalls
- stargate has configured SSH access from chonk via openclaw key pair

3.2.5 Physical Topology

The lab spans multiple physical locations and subnets:

- Main office where Franklin and Sly have their desks monitors (music.lan)
- Server racks hosting core infrastructure (time, odroid-c1/KDC)
- Jetson cluster (node900–node903) for GPU compute
- Public cloud (wonderland/178.62.60.55) for external-facing services

Routing between the 192.168.86.x subnet (music) and the rest of the lab requires hopping through skynet due to the non-standard routing configuration.

3.3 DNS

BIND on ns1 provides authoritative DNS for the lab, with forward and reverse zones plus Kerberos SRV records. Currently offline – see Disaster Recovery for recovery procedures.

3.4 DNS Configuration

3.4.1 BIND Overview – Post-Migration (August 2026)

The DNS infrastructure has been migrated from the legacy `server1` (10.10.12.12) to a new authoritative setup:

- **DNS Master**: **node3** (10.10.12.3) – promoted from client node to authoritative BIND9 master
- **Legacy DNS host**: `server1` (10.10.12.12) has been decommissioned; all zone transfers and updates now target node3

The DNS role in lab-franklin's Ansible collection manages:

- Forward zone: db.home.lab (hostnames to IPs)
- Reverse zones: PTR records for each 10.10.x.0/24 subnet
- SRV records: Kerberos service discovery
- Zone file generation via Jinja2 templates on node3

3.4.2 Zone and Host Realignment (August 2026)

The following entries have been updated in the forward zone (db.home.lab):

Hostname	IP Address	Notes
stargate.research.bitsmasher.net	10.10.16.66	Ansible orchestration host, primary workspace
chonk.lab.bitsmasher.net	10.10.8.60	Gateway host, primary compute + inference

The corresponding reverse zones (PTR records) have been updated to match:

- 10.10.16.66 → stargate.research.bitsmasher.net
- 10.10.8.60 → chonk.lab.bitsmasher.net

3.4.3 Kerberos SRV RFC Compliance

The following SRV records now point directly to FQDN A records rather than CNAMEs, per RFC 2782:

```
_kerberos-adm._tcp.lab.bitsmasher.net IN SRV 0 100 749 kdc1.lab.bitsmasher.net.
_kerberos._tcp.lab.bitsmasher.net     IN SRV 0 0 88  kdc1.lab.bitsmasher.net.
_kerberos._udp.lab.bitsmasher.net     IN SRV 0 0 88  kdc1.lab.bitsmasher.net.
_kdc._tcp.lab.bitsmasher.net          IN SRV 0 0 88  kdc1.lab.bitsmasher.net.
_kdc._udp.lab.bitsmasher.net          IN SRV 0 0 88  kdc1.lab.bitsmasher.net.

```

Note: The $_kerberos-adm._tcprecordnowpointsdirectlytotheArecordfor$`kdc1.lab.bitsmasher.net`$ratherthanviaa$

3.4.4 DNS Failure Impact

When the DNS master (node3) is offline:

- Lab hosts fall back to /etc/hosts entries (authoritative but static -- no dynamic updates)
- Kerberos KDC discovery via SRV records fails -- clients need manual kdc = kdc1.lab.bitsmasher.n
  in /etc/krb5.conf
- Dynamic zone updates cannot propagate; new hosts require manual /etc/hosts sync

3.4.5 DNS Maintenance Procedures

Zone file updates are committed to the DNS Ansible role and deployed via:

```
ansible-playbook -l dns_servers -t dns deploy.yml

```

Forward and reverse zone files should be audited quarterly for IP realignment accuracy.

Chapter 4

Core Services

4.1 Network Time (NTP)

A two-tier NTP architecture provides time synchronization from the Stratum-1 GPS-referenced time host to all lab clients.

4.2 Network Time (NTP)

4.2.1 Overview

Time synchronization is one of the most critical and least-visible aspects of network management. All lab services -- Kerberos authentication, TLS certificate validation, log correlation, distributed databases, and filesystem consistency -- depend on time accuracy. The lab uses a two-tier NTP architecture:

1. **Stratum 1 source**: GPS-referenced clock on the time host (10.10.12.2) provides the lab's authoritative time reference.
2. **Stratum 2+ clients**: All other hosts sync to time.lab.bitsmasher.net.

4.2.2 The Time Host

- **Hostname**: time.lab.bitsmasher.net (alias: time)
- **IP**: 10.10.12.2
- **User**: openclaw
- **OS**: Debian 12 bookworm
- **Role**: Stratum-1 NTP server / reference clock source for the lab

The time host has a GPS unit connected (serial/tty interface) providing UTC discipline. Its ntpd is configured via ntpsec to serve the lab subnet and use the GPS PPS signal as the primary time reference.

4.2.3 Configuration on Client Hosts

The NTP configuration is managed by Ansible via the ntp role in the lab-franklin collection. Here is what a typical client configuration looks like (as seen on stargate):

```
# /etc/ntpsec/ntp.conf -- managed by Ansible
driftfile /var/lib/ntp/ntp.drift
leapfile /usr/share/zoneinfo/leap-seconds.list

statsdir /var/log/ntpstats/

```

```
statistics loopstats peerstats clockstats
filegen loopstats file loopstats type day enable
filegen peerstats file peerstats type day enable
filegen clockstats file clockstats type day enable

server time.lab.bitsmasher.net
restrict 10.10.8.0/21
restrict -4 default
```

Key points:

- driftfile: Tracks oscillator drift between NTP restarts. Critical for stability on hosts that reboot frequently (Jetson devices, test VMs).
- statistics: Logs loop/peer/clock stats daily in /var/log/ntpstats/. Use these to verify sync health.
- server: Points to the lab's time host -- never pool.ntp.org for infrastructure hosts that need deterministic time sources.
- restrict:
  - 10.10.8.0/21 grants query access to the main lab subnet (used by other internal clients or monitoring tools).
  - -4 default blocks all IPv4 queries not explicitly allowed, providing a deny-by-default posture.

4.2.4 Installation and Setup Steps

For a new host that needs NTP:

1. Install ntpsec packages:

```
apt update && apt install -y ntpsec

```

2. Configure /etc/ntpsec/ntp.conf (either manually or via Ansible ntp role):

   (a) Set server time.lab.bitsmasher.net as the upstream reference.

   (b) Set timezone: timedatectl set-timezone America/Denver (or appropriate TZ).

   (c) Create the log directory if it doesn't exist: mkdir -p /var/log/ntpstats.

3. Start and enable the service:

```
systemctl enable --now ntpsec.service
timedatectl  # verify "System clock synchronized: yes"

```

4.2.5 Verification on Clients

Check synchronization status with these commands:

```
# Check if the system clock is synchronized
timedatectl | grep "System clock synchronized"

# Check NTP peer status (if ntpq is available)
ntpq -p

# View recent sync logs
journalctl -u ntpsec --since today

```

4.2.6 Notes for Deployment

- The Ansible ntp role handles all of the above automatically across lab hosts. It installs packages, copies configuration, sets timezone, and creates log directories.
- For Jetson devices (node900--node903), drift compensation is especially important -- they boot frequently and have less stable oscillators than desktop/server hardware.
- The GPS unit on the time host should be verified periodically: confirm it's locking to satellites, not using its internal oscillator as a holdover source. Check this by running ntptime on the time host itself.
- If the time host goes offline and all clients lose sync beyond their maximum correction threshold (typically 128 seconds), systems may need manual time correction before Kerberos/TLS will function again.

4.3 Kerberos Authentication

The KDC on odroid-c1 runs MIT Kerberos for the lab.bitsmasher.net realm, managing authentication for all services.

4.4 Kerberos Authentication

4.4.1 Realm Configuration

- **Realm name**: lab.bitsmasher.net
- **KDC host**: odroid-c1.lab.bitsmasher.net (10.10.12.254)
- **KDC software**: MIT Kerberos KDC running on odroid-c1
- **Uptime**: 242 days (stable, rarely rebooted)

4.4.2 Access and Key Management

The KDC only has SSH key auth configured for the root user. The franklin user has been rejected by the KDC's SSH configuration -- direct root access is required for any management tasks.

Key file: ~/.ssh/id_ed25519_openclaw (the standard openclaw key).

4.4.3 Principal Conventions

Kerberos principals in the lab.bitsmasher.net realm should follow these conventions:

- Service principals: service/FQDN@lab.BITSMASHER.NET
- User principals: username@lab.BITSMASHER.NET
- Host principals: host/FQDN@lab.BITSMASHER.NET

4.4.4 DNS/Kerberos Integration – RFC Compliant (August 2026)

Kerberos relies on DNS SRV records for KDC discovery. The following records point directly to FQDN A records (no CNAME indirection):

```
_kerberos-adm._tcp.lab.bitsmasher.net IN SRV 0 100 749 kdc1.lab.bitsmasher.net.
_kerberos._tcp.lab.bitsmasher.net     IN SRV 0 0 88  kdc1.lab.bitsmasher.net.
_kerberos._udp.lab.bitsmasher.net     IN SRV 0 0 88  kdc1.lab.bitsmasher.net.
_kdc._tcp.lab.bitsmasher.net          IN SRV 0 0 88  kdc1.lab.bitsmasher.net.
_kdc._udp.lab.bitsmasher.net          IN SRV 0 0 88  kdc1.lab.bitsmasher.net.

```

The $_k erberos-adm._tcprecordnowpointsdirectlytotheArecordfor$kdc1.lab.bitsmasher.net, $perRFC2782complian$
These records are authoritative on node3 (10.10.12.3), the new DNS master following the server1 decommissioning. If DNS is unavailable, clients fall back to manual kdc = kdc1.lab.bitsmashe in /etc/krb5.conf.

4.4.5 Keytab Management

Keytabs for service principals should be:

- Generated on the KDC host (odroid-c1) using kadmin.local or kadmin -q
- Transferred securely to target hosts (via scp with SSH keys)
- Set to mode 600 and owned by the appropriate service user

Note: Kerberos keytabs have not been deployed across the infrastructure. NFS exports with sec=krb5i are currently non-functional pending keytab deployment. The Ansible ssh role handles the StrictModes requirements for SSH-based keytab transfer.

4.5 LDAP Directory Services

OpenLDAP (slapd) on bbb1 provides directory-based identity management. Service has been offline since December 2025 due to a TLS certificate issue.

4.6 LDAP Directory Services

4.6.1 Server Configuration

- **Hostname**: ldap.lab.bitsmasher.net (alias: bbb1)
- **IP**: 10.10.13.1
- **Software**: OpenLDAP (slapd)
- **Status**: SSH reachable as root, but slapd service has been **failed since December 2025**

4.6.2 Outage Details

The slapd failure manifests as:

```
TLS init def ctx failed (-1)

```

This indicates a TLS certificate configuration issue -- likely an expired, missing, or misconfigured CA certificate. The server is listening on its network interface (SSH works for root), but the LDAP daemon itself has not recovered since December 2025.

4.6.3 Monitoring Approach

Franklin maintains /home/franklin/status.sh which checks:

- slapd service status on bbb1/ldap
- ldapsearch queries for franklin and sly DN lookups -- both fail because the directory is down

The script provides a simple pass/fail indicator for LDAP health without requiring complex monitoring infrastructure.

4.6.4 Recovery Steps

To restore the LDAP server:

1. SSH to ldap.lab.bitsmasher.net as root (ed25519 key)
2. Check slapd logs: journalctl -u slapd --since "2025-12-13"
3. Verify TLS certificate chain in /etc/ssl or the configured cert path
4. Update or regenerate the CA-signed certificate if expired
5. Restart slapd: systemctl restart slapd
6. Verify with ldapsearch for franklin and sly DNs

4.6.5 Directory Structure (Expected)

The OpenLDAP directory should contain entries for:

- User entries for franklin, Sly (slyborg), and other lab members
- Group/organizational unit entries for lab access control
- Service accounts for LDAP-authorized services (Kerberos integration, web auth)

Once restored, the LDAP directory serves as the central authentication source for the lab, complementing Kerberos for identity management.

Chapter 5

Security

5.1 Authentication and Access Control

SSH key management, password policy, and user accounts define access to the lab infrastructure. All automation hosts share a common ed25519 key pair for the openclaw user. The primary admin (franklin) has sudo NOPASSWD on skynet; root is restricted to KDC, ldap, and music host only.

5.2 Network Security

Subnet segmentation provides implicit isolation across the lab. The ``prereq'' Ansible role handles firewall exceptions for K3s API ports on each host (6443, 2379--2381). Both UFW (Debian) and Firewalld (RedHat-family) are supported.

5.3 Certificate and TLS Management

The ``tls'' role manages certificate provisioning across the lab. The OpenLDAP server experienced a TLS failure in December 2025 due to an expired certificate -- this remains the primary ongoing TLS vulnerability.

5.4 Credential Management

Credentials use pass (password manager) with GPG encryption. The Stash House project manages encrypted credential backups synced via bare git repos. Sensitive Ansible variables are stored in ansible-vault files that should never be committed to version control.

5.5 Backup and Disaster Recovery

NFS-mounted clusterfs2 provides shared storage with redundant copies across hosts. DNS zones are version-controlled in BIND configuration files.

Service availability:

- **NTP stratum-1** (time.lab.bitsmasher.net) -- Online
- **Kerberos KDC** (odroid-c1.lab.bitsmasher.net) -- Online, 242 days uptime
- **DNS BIND** (ns1.lab.bitsmasher.net) -- Offline since December 2025
- **OpenLDAP** (ldap.lab.bitsmasher.net) -- Offline since December 2025
- **Minecraft** (skynet.lab.bitsmasher.net) -- Online

Disaster recovery priority: NTP > DNS > Kerberos KDC > LDAP. Time sync failure cascades to all Kerberos authentication; DNS failure makes hosts unreachable by name.

5.6 Container and Kubernetes Security

The k3s cluster is hardened with version pinning (2.1.x), Containerd runtime with NVIDIA GPU support, TLS for all API communication (k3s default), and pod security policies via Kubernetes native admission controllers.

5.7 Automated Security Scanning

The .github directory contains these automated workflows: bandit.yml (Bandit Python security), tfsec-sarif.yml (Terraform IaC security), trivy.yaml + trivy-cb.yml (container/filesystem vulnerability scanning), bash_check.yaml (GNU autotools build validation with ``make security''), and markdown.yml (markdownlint). All run on every pull request and main branch push.

Chapter 6

Storage

6.1 NFS Mounts and Backup

clusterfs2 provides shared NFS storage across hosts. Credential backups use GPG-encrypted pass entries synced via bare git repos (Stash House project).

6.2 Storage Infrastructure

6.2.1 Active Disk Topology (chonk – August 2026)

All compiled documents and web artifacts target wonderland; chonk serves as local scratch space. Physical topology on chok:

- **/mnt/backup1**: Source directory for Ansible repos, workspace code, and build artifacts
- **/mnt/clusterfs**: Scratch disk for temporary compilation, LaTeX builds, and intermediate files
- **/mnt/snowy**: Bulk storage -- formerly a full NFS share; hard disk physically removed, now mounted as raw drive only

The following mount points have been purged:

- **/mnt/storage1**: Completely removed across all roles and /etc/exports on all hosts
- Legacy /mnt/storage2, /mnt/storage3: Purged from inventory references

6.2.2 NFS Mounts – Current State

The lab relies on NFS for shared storage. Active mounts as of August 2026:

- **stargate:/mnt/clusterfs2**: Primary cluster workspace; mounted on stargate and accessible from wonderland via SSH-sync pipelines
- **Jetson home directories**: Shared from stargate via stargate:/mnt/clusterfs2 -- provides consistent ~/home for node900--node903

6.2.3 StrictModes and Permission Requirements

OpenSSH StrictModes enforces strict permission checks on home directories. Required permissions:

- Home directory: 0750 (rwxr-x---)
- ~/.ssh: 0700 (rwx------)
- ~/.ssh/authorized_keys: 0600 (rw-------)

All Jetson nodes must comply with these permissions or SSH authentication fails silently. NFS-mounted homes require uid/gid consistency between stargate and each Jetson host.

6.2.4 Snapshot and Backup Strategy

The lab uses bare git repos for version control and backup:

- GPG-encrypted repositories synced across hosts for credential protection (Stash House project)
- Pass-based password store with GPG backend
- No LFS -- ABSOLUTE RULE: do not use git-lfs anywhere. Microsoft once held Minecraft chunk files hostage when a repo hit the 10GB GitHub limit with LFS enabled.

6.2.5 LFS Policy Reminder

ABSOLUTE RULE: Do NOT use git-lfs anywhere in any lab-franklin repository.

This applies across all repos: training-ai, minecraft, Ansible collections, everything. The incident with Minecraft chunk files being held hostage by GitHub when the repo hit 10GB with LFS is a real precedent -- never risk it.

6.3 LFS Policy

Absolute rule: no git-lfs in any lab repository -- see the storage section for details on the incident that made this policy necessary.

Chapter 7

Internal PyPI Repository

7.1 Overview

An internal PyPI server provides Python packages to development machines without depending on external PyPI.org servers. Packages are stored on NFS-mounted storage and served via HTTP from the web document root.

7.2 Internal PyPI Repository

The lab runs an internal PyPI package repository to provide Python packages to development machines without depending on external PyPI servers. This avoids network bottlenecks, supports offline development, and enables use of proprietary or lab-specific packages not available on PyPI.org.

7.2.1 Architecture

The PyPI directory lives at /mnt/storage1/LAB/pypi. A symlink is created from the web server's document root (/var/www/html/pypi) pointing to this directory, making packages accessible via HTTP.

Access URL pattern:

```
http://<storage-host>/pypi/simple/

```

This follows the PEP 503 simple repository API structure expected by pip.

7.2.2 Configuration via Ansible

The pypi_internal role in the lab/franklin collection automates this setup:

- **Role path**: roles/pypi_internal/
- **Variables**:
  - html_dir: web root (/var/www/html)
  - pypi_dir: package directory (/mnt/storage1/LAB/pypi)
- **Task**: Creates a symbolic link from $html_dir/pypi → $pypi_dir
- **Playbook integration**: Included in the cluster/storage playbook alongside NFS, LDAP, Kerberos, DNS, and ntp roles

7.2.3 Usage from Client Machines

Configure pip to use the internal PyPI server:

```
# In ~/.pip/pip.conf or /etc/pip.conf:
[global]
index-url = http://<storage-host>/pypi/simple/

# Optionally trust host if not using HTTPS
trusted-host = <storage-host>

```

Or use pip's indexurl flag directly:

```
pip install --index-url http://<storage-host>/pypi/simple/ some-package

```

7.2.4 Package Maintenance

To add packages to the internal PyPI, place them in the pypi directory with PEP 503-compliant directory structure:

- Package directories (lowercase names)
- File links inside each package directory pointing to the actual wheel/sdist files
- The web server must have read access to /mnt/storage1/LAB/pypi

Packages can be built and added using standard pip tools:

```
pip install --no-index --find-links=/mnt/storage1/LAB/pypi/simple/ <package>

```

Chapter 8

Container Orchestration

8.1 k3s Cluster

k3s provides Kubernetes across the infrastructure with version pinning (2.1.x), containerd runtime, and NVIDIA Container Toolkit for GPU workloads.

8.2 Kubernetes (k3s) Cluster

8.2.1 Architecture

The lab uses k3s (Lightweight Kubernetes from Rancher) for container orchestration across the infrastructure:

- **k3s_server**: Master node running k3s control plane (host: head2, referenced but unreachable in current audit)
- **k3s_agent**: Worker nodes that register with the server and receive workloads
- **GPU nodes**: Specialized workers for NVIDIA container runtime and compute workloads

8.2.2 Version Pinning

k3s version is pinned to the 2.1.x series (specifically 2.1.5 as of the latest network_update.sh). This prevents surprise breakage from upstream k3s releases. The pinning policy follows:

```
K3S_VERSION="2.1.5"
curl -sfL https://get.k3s.io | sh -s - \
    --write-kubeconfig-mode 644

```

8.2.3 Container Runtime

k3s uses containerd as the default runtime, with NVIDIA Container Toolkit for GPU workloads:

- **nvidia-container-toolkit**: Installed on GPU nodes via the network_update.sh automation
- **containerd config**: Located at /var/lib/rancher/k3s/agent/etc/containerd/config.toml
- NVIDIA runtime configured in containerd to expose GPU devices to containers

8.2.4 Cluster Topology

Node Role	Target Hosts	Notes
k3s_server	head2 (planned)	Control plane; not yet online
k3s_agent	node900, node901, node903	Jetson nodes as workers
nvidia_nodes	GPU-equipped hosts	Container Toolkit + k3s agent

8.2.5 kubeconfig Access

After installation, kubeconfig is written to /etc/rancher/k3s/k3s.yaml with mode 644. Cluster access requires:

- SSH key auth to the k3s_server host
- Reading the kubeconfig file (or copying it securely)
- kubectl --kubeconfig /etc/rancher/k3s/k3s.yaml cluster-info to verify connectivity

The k3s$_{s}erverroleandk3s_{a}gentroleinlab-franklinhandletheinstallationandconfigurationofthesenodesviaAnsible$

Chapter 9

Playbook Automation

9.1 Ansible Automation

9.1.1 Current Landscape (August 2026)

As of August 2026, the Ansible ecosystem is split between two distribution paths:

- **Ansible community package** -- the traditional monolithic install (pip install ansible). Current version is 13.x (built on ansible-core 2.20). This includes 85+ collections with thousands of modules and plugins bundled together. Follows semantic versioning, releases two major versions per year plus monthly minor releases.
- **ansible-core** -- the minimal core engine. Maintains three versions at a time (current + two older). Versioned independently from the community package (2.19, 2.20, 2.21). Releases every four weeks for minor updates.

EOL versions as of August 2026: Ansible 10--12 and ansible-core 2.15--2.18 are EOL. Do not use these in new work. Only Ansible 13.x / ansible-core 2.20 is current; 14.0.0/2.21 is in development.

9.1.2 Molecule Testing – Deprecated (August 2026)

Molecule has been removed from the active workflow as of August 2026. It no longer serves as a testing target or standard. All role validation now uses:

- **ansible-test --sanity** -- built-in collection sanity checks, run natively on stargate.research.bi with zero external overhead
- **ansible-test --integration** -- integration tests targeting real hosts or containerized environments via ansible-test's native docker driver
- **ansible-playbook --syntax-check** -- YAML structure validation for all playbooks and role task files

Existing Molecule test harnesses (in ansible/collections/ansible_collections/lab/franklin/roles/<r remain as historical artifacts but are no longer invoked or maintained. New roles must use native ansible-test exclusively.

Chapter 10

lab/franklin Collection

The lab/franklin Ansible collection lives at:

```
/mnt/clusterfs2/workspace/lab-franklin/ansible/collections/ansible_collections/lab/
    franklin/

```

It is the central automation artifact for the entire lab.bitsmasher.net infrastructure, managing provisioning, configuration, and testing across all hosts.

10.1 Collection Manifest (galaxy.yml)

The collection follows standard Ansible Galaxy conventions with namespace lab and name franklin:

- **Version**: 1.0.0
- **License**: MIT
- **Author**: franklin <franklin@bitsmasher.net>
- **Repository**: https://github.com/devsecfranklin/lab-franklin
- **Documentation**: https://github.com/devsecfranklin/lab-franklin/docs

10.2 Role Inventory

The collection contains roles organized under the roles/ directory. As of August 2026, the following roles are maintained with native ansible-test harnesses:

```
roles/
 apt_mirror/          # APT mirror configuration
 beagleboard/         # BeagleBoard device provisioning
 common/              # Shared base configuration (bootstrap.sh, common utils)
 container_registry/  # Docker/container registry setup
 ctfd/                # CTFd platform deployment
 desktop/             # Desktop environment provisioning
 docker/              # Docker engine installation
 documentation/       # Documentation generation roles
 dhcp/                # DHCP server configuration (dhcpd.conf)
 dns/                 # BIND/named DNS server
 golang/              # Go language toolchain
 jetson-nano/         # NVIDIA Jetson Nano provisioning
 k3s_agent/           # k3s worker node
 k3s_server/          # k3s control plane server
 k8s/                 # Kubernetes generic setup
 latex/               # LaTeX environment
 logging/             # Centralized logging
 music/               # Music service configuration
```

```
nfs/                  # NFS server/client
nix/                  # Nix package manager
openbsd/              # OpenBSD host bootstrap (blowfish target)
    tasks/main.yml    # Includes files.yml, timezone setup
    files/            # bootstrap.sh, login.conf, hosts entries
paloalto/             # Palo Alto firewall config
prereq/               # Prerequisite packages
pypi_internal/        # Internal PyPI repository setup
    tasks/main.yml    # Symlink pypi_dir -> html_dir/pypi
    vars/main.yml     # html_dir=/var/www/html, pypi_dir=/mnt/storage1/LAB/pypi
python/               # Python environment
raspberrypi/          # Raspberry Pi provisioning
samba/                # Samba file sharing
security/             # Hardening and security baselines
shell/                # Shell configuration
ssh/                  # SSH server hardening
tls/                  # TLS certificate management
chonk/                # chonk-specific configuration
cluster/              # Cluster-wide settings

```

10.3 Workspace Structure

The full project layout:

```
/workspace/lab-franklin/
 Makefile.am                  # Top-level autotools target (Python, Docker, dev)
 configure.ac                 # Autotools config (autoconf)
 bootstrap.sh                 # Cross-platform bootstrapper
 network_update.sh            # One-shot maintenance script (hardened 2026-08-02)
 ansible/                     # ANSIBLE_HOME root
     playbook.yml              # Main playbook
     hosts                     # Inventory file
     collections/              # Collections path
         ansible_collections/ # Galaxy namespace
             lab/franklin/    # The lab/franklin collection
                 galaxy.yml
                 roles/       # All role modules (37 total)
                 docs/latex/  # LaTeX docs build (autotools)
 container/                   # Docker/Podman configs
 terraform/                   # Terraform infrastructure definitions
 bin/                         # Shared utility scripts (common.sh)
 docs/manual/                 # Lab manual documentation (this document)

```

10.4 Build System (Autotools)

The project uses GNU Autotools for the root-level build:

- **configure.ac**: Defines package (lab-franklin v0.2), libtool support, Python $\geq$ 3.9 requirement, podman/ansible/gpg checks, git version tagging
- **Makefile.am**: Top-level targets include test (ansible-lint + ansible-test sanity/integration), security (venv bootstrap), dev (Python venv in BUILDDIR)
- **bootstrap.sh**: Cross-platform bootstrapper that detects OS (Debian/RedHat/OpenBSD/macOS/Linux) and installs platform-specific packages, then runs aclocal $\rightarrow$ autoreconf $\rightarrow$ automake $\rightarrow$ configure
- **docs/manual/Makefile.am**: Standalone autotools stub for LaTeX docs build (clean target removes *.aux files)

`The network_update.sh script was hardened on 2026-08-02 with: set -euo pipefail, auto-resolved paths, pinned K3s version validation, ANSIBLE`$_H OME$ *environment checks, removed dangerous clush/apt-get blast radius, and dead code removal. It runs the main playbook at* `ansible/playbook.yml` *as its primary action.*

Chapter 11

Testing from Stargate

11.1 Native ansible-test on stargate.research.bitsmasher.net

The stargate host (10.10.16.66, Debian 12 bookworm) serves as the exclusive testing platform for the lab/franklin Ansible collection. All test execution runs with zero external billing -- direct shell access to the gateway sandbox eliminates API token overhead.

11.1.1 Testing Standard: ansible-test (August 2026)

Native ansible-test has replaced Molecule as the canonical testing framework:

- **sanity checks**: Built into ansible-core; validates module/plugin imports, YAML structure, and collection metadata without any container dependency
- **integration tests**: Run via ansible-test integration targeting real hosts or the native docker driver (preferred over podman for consistency)
- **syntax validation**: ansible-playbook --syntax-check validates all playbooks and role task files before execution

The legacy Molecule directory structure at each role's molecule/ subdirectory has been deprecated. It remains in-place as historical reference but is excluded from CI pipelines and the Makefile.am test target.

11.1.2 Test Execution Flow (ansible-test)

For a given role, the native test cycle is:

1. **syntax-check**: ansible-playbook --syntax-check playbook.yml validates YAML structure
2. **sanity**: ansible-test sanity runs built-in linting against the collection -- no container, no overhead
3. **integration**: ansible-test integration <role> runs target-specific tests using docker driver containers

11.1.3 Running Tests from Stargate

From stargate's workspace:

```
cd ~/workspace/lab-franklin/ansible/collections/ansible_collections/lab/franklin
ansible-test sanity
ansible-test integration dns --python 3.12
ansible-playbook --syntax-check playbook.yml

```

11.2 Dynamic Scoping and Pathing Convention

Roles use dynamic file inclusion via templated variables to avoid hardcoded paths:

- Roles include task fragments using: _role_name_files and _role_name_templates for dynamically scoped file/paths
- Template references follow the pattern: {include_task "tasks/_role_name.yml"} scoped within dynamically included tasks

This convention eliminates static path dependencies across role invocations and supports multi-target deployment without playbook-level path overrides.

11.3 Package Standards – Deprecated Utilities

The following deprecated utilities have been removed from baseline role manifests:

- **neofetch**: Removed from common role; replaced by standard Linux tools (lsb_release, uname -r)
- **tripwire**: Removed from security role; no longer maintained and superseded by native file integrity monitoring
- Other legacy packages audited during August 2026 cleanup -- verify role manifests before adding new dependencies

11.4 Dynamic Probing Standard

Static inventory reachability assumptions have been replaced with non-interactive batch probes:

```
ssh -o BatchMode=yes -o ConnectTimeout=3 -i ~/.ssh/id_ed25519_openclaw franklin@<
    host> "hostname && whoami"

```

This approach eliminates stale host entries, prevents password prompts in automation pipelines, and provides deterministic connectivity feedback with a 3-second timeout. Never assume reachability from static notes -- always probe on demand.

Chapter 12

Additional Infrastructure

12.1 Blowfish – Database Host

blowfish is the planned database host (OpenBSD). Currently unreachable; SSH key not yet authorized.

12.2 Blowfish – Database Host

12.2.1 Host Identity

- **Hostname**: blowfish.lab.bitsmasher.net (alias: blowfish)
- **IP**: 10.10.12.15
- **MAC**: 98:b7:85:21:ad:77 (from DHCP reservation)
- **OS**: OpenBSD (planned; current DNS/DHCP entry references it as an OpenBSD host)

12.2.2 Role in the Lab

Blowfish is provisioned as a **database host** within the lab infrastructure. It was intended to serve as a dedicated database server, separate from the primary infrastructure hosts (chonk, stargate, skynet).

The OpenBSD role in the lab-franklin Ansible collection includes configuration references specific to blowfish:

- Login configuration uses blowfish password hashing algorithm (localcipher=blowfish,a) in /etc/login.conf.
- The bootstrap.sh for OpenBSD hosts includes a direct /etc/hosts entry for blowfish (10.10.12.15).
- DHCP reservation is configured in the lab's dhcpd.conf role.

12.2.3 Current Status

Blowfish has been **unreachable** from chonk since the initial infrastructure audit. The IP 10.10.12.15 resolved from DNS (stargate's /etc/hosts), but authentication was denied -- no matching SSH key was authorized on that host.

Key references:

- network_update.sh: blowfish is the default target for the openbsd_bootstrap() function.
- OpenBSD role tasks: includes commented-out ansible setup command for blowfish (indicating the host was expected to be online).

The host may need network investigation -- it resolved from DNS, had a DHCP reservation, and MAC address on file, suggesting it was once part of the lab but has since gone offline or been decommissioned.

12.2.4 Planned Configuration (OpenBSD)

When brought online, the expected setup for blowfish is:

- SSH key-based authentication via /.ssh/authorized_keys
- OpenBSD's built-in pf firewall with custom rules
- Database software stack (likely PostgreSQL or MariaDB on OpenBSD)
- Ansible-managed configuration via the lab-franklin collection, using the openbsd role as the base profile.

12.3 Minecraft Server

The Minecraft server runs on skynet, with world data stored on NFS and version-controlled via bare git repos.

12.4 Minecraft Server

12.4.1 Host and Location

The Minecraft server runs on **skynet.lab.bitsmasher.net** (10.10.16.10), one of the core lab infrastructure hosts alongside stargate and chonk. The Minecraft role is managed by the minecraft Ansible role in the lab-franklin collection.

Skynet's dual-role setup:

- Primary: Minecraft game server host
- Secondary: General-purpose infrastructure node (Ansible workspace, CI testing)

Note: skynet uses IP-based SSH login (10.10.16.10) rather than hostname -- the hostname resolves oddly in SSH config due to a double-domain suffix issue.

12.4.2 World and Chunk Storage

Minecraft world data is stored on NFS-mounted storage (clusterfs2). The world files are version-controlled where practical, with the following constraint:

NO GIT-LFS: Chunk files and world data must not use git-lfs. The Minecraft repository hit GitHub's 10GB LFS limit in a prior incident where chunk files were held hostage. Use bare git repos for any versioned world data.

12.4.3 Dreamland Connection

The *Dreamland Manual* referenced in the docs/manual directory covers the Dreamland Minecraft server -- this manual documents that specific server's configuration, players, and game rules. The physical infrastructure (skynet host) is shared between both operations.

12.4.4 Server Management via Ansible

The minecraft role handles:

- Minecraft server installation and version updates
- World backup procedures (NFS snapshots or bare git repos)
- Player whitelist management
- Mod/plugin configuration if applicable

Molecule testing for the minecraft role confirms correct file placement and configuration generation, though live server connectivity is not verified by the test harness.

Chapter 13

Disaster Recovery

13.1 Disaster Recovery

13.1.1 Critical Dependencies Chain

The lab infrastructure has a cascading dependency chain. A failure in one layer propagates:

1. **Time (NTP)** → if time drift exceeds 128 seconds, Kerberos tickets expire and TLS handshakes fail
2. **DNS (ns1/BIND)** → without DNS resolution, all hostname-based services (Kerberos SRV records, NTP upstream, LDAP) break
3. **Kerberos (odroid-c1/KDC)** → without the KDC, authentication fails for all Kerberos-authenticated services
4. **LDAP (bbb1/slapd)** → without LDAP, directory lookups fail and user/service account information is unavailable

Recovery priority should follow this chain in reverse: fix time first, then DNS, then KDC, then LDAP.

13.1.2 Failure Scenarios and Recovery

NTP Failure – All Hosts Lose Time Sync

Symptoms: timedatectl shows "System clock synchronized: no", Kerberos tickets rejected.
Recovery:

1. Verify GPS unit is locked on odroid-c1 (time host)
2. Check ntpd/ntpsec service: systemctl status ntpsec
3. If GPS is offline, enable holdover mode (internal oscillator continues at reduced accuracy)
4. On affected clients, force resync: ntpdate -s time.lab.bitsmasher.net or wait for ntpd's step-sync to kick in
5. Verify with ntpq -p on clients and ntptime on the server

DNS (ns1) Offline – Cascade Failure

Symptoms: hostnames don't resolve, Kerberos SRV discovery fails.
Recovery:

1. Check ns1 reachability: ping from any known-good host
2. If ns1 is up but not answering: check BIND (named) service status

3. If BIND has crashed due to config error: restore from last known-good named.conf
4. As a workaround, add manual entries to /etc/hosts on affected hosts until DNS recovers
5. Update krb5.conf with explicit kdc = odroid-c1.lab.bitsmasher.net

LDAP (bbb1) Down – No Directory Service

Symptoms: slapd failed, status.sh reports both slapd and ldapsearch as failing.
Recovery:

1. SSH to bbb1/lab.bitsmasher.net as root
2. Check TLS certificate validity in the configured cert path
3. Regenerate or replace expired certificates
4. Restart slapd: systemctl restart slapd
5. Verify with ldapsearch for franklin and sly DNs

KDC (odroid-c1) Offline – No Authentication

Symptoms: kinit fails for all realms, service authentication denied.
Recovery:

1. SSH to odroid-c1 as root (franklin user's key is not authorized)
2. Check kdc process: kadmind/krb5kdc status
3. Verify KDC database integrity: kdb5$_{u}tillist$
4. Restart kerberos services if needed
5. Regenerate host keytabs for affected services via kadmin.local

13.1.3 Backup and Restoration

- Ansible playbooks (lab-franklin collection) are the single source of truth -- they can reconstruct any configured state
- Bare git repos with GPG-encrypted pass entries handle credential backup
- KDC database ($KRB5_KDB_FILE) should be backed up regularly on odroid-c1
- DNS zone files should have local copies (in the DNS role's files/ directory)

13.1.4 Prevention: Monitoring

The current monitoring approach is manual via status.sh. For improved resilience, consider:

- Automated ping/SSH checks on critical hosts in HEARTBEAT.md
- Nagios/Zabbix-style monitoring on a dedicated host
- Email/slack alerts for service down conditions

The lab-franklin Ansible collection should eventually include a monitoring role to automate this.

Chapter 14

Troubleshooting Guide

14.1 Troubleshooting Guide

14.1.1 SSH Connection Failures

Host Key Mismatch

Symptom: @@@@@ WARNING: REMOTE HOST IDENTIFICATION HAS CHANGED! @@@@@
Fix:

```
ssh-keygen -R <hostname>

```

This occurs when a host was re-imaged with the same IP. Check /etc/ssh/ssh_config.d/ for stale entries.

Auth Denied (Port 22 Open, Key Rejected)

Symptom: TCP port 22 is reachable but SSH returns "Permission denied".
Common causes:

- No matching key in the remote host's authorized_keys
- The key file permissions are too open on the target (must be mode 600 for .pub and 700 for /.ssh)
- Wrong user context (e.g., trying franklin@ when only root is configured)

Fix: Add the openclaw ed25519 key to the target host's authorized_keys file. The pubkey is:

```
ssh-ed25519 AAAAC3NzaC1lZDI1NTE5AAAAIJb9mV02PpxD8VpzYCnu7192dTwMnGWUc3qh55BIeElN
    openclaw@chonk

```

No Route to Host

Symptom: SSH hangs or returns "No route to host". The host is likely powered off. Common in Jetson nodes (node902, etc.) that are manually managed.

GSSAPI Interference

Symptom: SSH times out waiting for GSSAPI authentication before falling back to key auth.
Fix: Add to /.ssh/config:

```
Host *
    GSSAPIAuthentication no
    IdentitiesOnly yes
    IdentityFile ~/.ssh/id_ed25519_openclaw

```

14.1.2 Service-Specific Diagnostics

Kerberos Authentication Failure

Symptom	Likely Cause
kinit fails with "clock skew"	NTP not synced -- check time.host first
kinit fails with "KDC has no support for encryption type"	Wrong krb5.conf realm or KDC unavailable
kadmin.local works but kadmin (remote) fails	Port 749 firewall rule on odroid-c1
Ticket expired after reboot	Time drift during host downtime

Verify: klist, kinit username@LAB.BITSMASHER.NET, ktutil for keytabs.

NTP Issues

Symptom	Likely Cause
ntpd not running	ntpsec service not enabled
"no server suitable for synchronization"	No upstream reachable (time host offline)
Clock gradually drifting	Missing driftfile or Jetson oscillator drift
Time jumps by several seconds	Large correction applied; step-sync mode engaged

Verify: ntpq -p, timedatectl, check /var/log/ntpstats/.

NFS Mount Issues

Symptom	Likely Cause
mount hangs	NFS server unreachable or export doesn't list client
"Stale file handle"	Server-side data was deleted/recreated
Permission denied on mount	/etc/exports restricts the client IP

Verify: showmount -e <server>, check NFS server's exports, verify client IP matches the export rule.

14.1.3 Common Ansible Troubleshooting

Module Not Found / Missing Collection

```
ansible-galaxy collection install lab.franklin
# or for individual modules:
ansible-galaxy collection install community.general

```

Molecule Test Failures on New Hosts

If molecule tests fail on a new host, the most common causes are:

1. Podman not installed or permissions wrong (openclaw user needs podman group)
2. Docker driver configured instead of podman -- edit molecule.yml to set driver
3. Python version mismatch (molecule requires $\geq$3.9)
4. Role hostname gates preventing test execution (e.g., the DNS role's hostname check for the real server vs. molecule container name)

14.1.4 General Debugging Checklist

When a service is down, work through this checklist:

1. Can you ping the host? (→ network up?)
2. Is SSH responding? (→ host OS running?)
3. Does the service process exist? (systemctl status or ps aux)
4. Are logs showing errors? (journalctl -u <service> --since today)
5. Can you connect to the service port from another lab host? (nc -zv <host> <port>)
6. Is DNS resolving correctly? (dig or getent hosts)
7. Is time synchronized? (timedatectl)

14.1.5 Quick Reference Commands

```
# Check if a host is reachable from chonk
ping -c 1 <hostname>
ssh -o ConnectTimeout=5 openclaw@<host> "echo alive"

# Check service status remotely
ssh openclaw@<host> "systemctl status <service>"

# Verify time sync across hosts
for host in stargate skynet chonk; do
    ssh openclaw@$host "timedatectl | grep synchronized"
done

# Verify DNS resolution from a client
dig @time.lab.bitsmasher.net <hostname>

# Check Kerberos auth
kinit username@LAB.BITSMASHER.NET
klist

```

Infrastructure Services

14.2 Network Time (NTP)

14.2.1 Overview

Time synchronization is one of the most critical and least-visible aspects of network management. All lab services -- Kerberos authentication, TLS certificate validation, log correlation, distributed databases, and filesystem consistency -- depend on time accuracy. The lab uses a two-tier NTP architecture:

1. **Stratum 1 source**: GPS-referenced clock on the time host (10.10.12.2) provides the lab's authoritative time reference.
2. **Stratum 2+ clients**: All other hosts sync to time.lab.bitsmasher.net.

14.2.2 The Time Host

- **Hostname**: time.lab.bitsmasher.net (alias: time)
- **IP**: 10.10.12.2
- **User**: openclaw
- **OS**: Debian 12 bookworm
- **Role**: Stratum-1 NTP server / reference clock source for the lab

The time host has a GPS unit connected (serial/tty interface) providing UTC discipline. Its ntpd is configured via ntpsec to serve the lab subnet and use the GPS PPS signal as the primary time reference.

14.2.3 Configuration on Client Hosts

The NTP configuration is managed by Ansible via the ntp role in the lab-franklin collection. Here is what a typical client configuration looks like (as seen on stargate):

```
# /etc/ntpsec/ntp.conf -- managed by Ansible
driftfile /var/lib/ntp/ntp.drift
leapfile /usr/share/zoneinfo/leap-seconds.list

statsdir /var/log/ntpstats/

statistics loopstats peerstats clockstats
filegen loopstats file loopstats type day enable
filegen peerstats file peerstats type day enable
filegen clockstats file clockstats type day enable

server time.lab.bitsmasher.net
restrict 10.10.8.0/21
restrict -4 default
```

Key points:

- driftfile: Tracks oscillator drift between NTP restarts. Critical for stability on hosts that reboot frequently (Jetson devices, test VMs).
- statistics: Logs loop/peer/clock stats daily in /var/log/ntpstats/. Use these to verify sync health.
- server: Points to the lab's time host -- never pool.ntp.org for infrastructure hosts that need deterministic time sources.
- restrict:
  - 10.10.8.0/21 grants query access to the main lab subnet (used by other internal clients or monitoring tools).
  - -4 default blocks all IPv4 queries not explicitly allowed, providing a deny-by-default posture.

14.2.4 Installation and Setup Steps

For a new host that needs NTP:

1. Install ntpsec packages:

```
apt update && apt install -y ntpsec

```

2. Configure /etc/ntpsec/ntp.conf (either manually or via Ansible ntp role):
   (a) Set server time.lab.bitsmasher.net as the upstream reference.
   (b) Set timezone: timedatectl set-timezone America/Denver (or appropriate TZ).
   (c) Create the log directory if it doesn't exist: mkdir -p /var/log/ntpstats.
3. Start and enable the service:

```
systemctl enable --now ntpsec.service
timedatectl  # verify "System clock synchronized: yes"

```

14.2.5 Verification on Clients

Check synchronization status with these commands:

```
# Check if the system clock is synchronized
timedatectl | grep "System clock synchronized"

# Check NTP peer status (if ntpq is available)
ntpq -p

# View recent sync logs
journalctl -u ntpsec --since today

```

14.2.6 Notes for Deployment

- The Ansible ntp role handles all of the above automatically across lab hosts. It installs packages, copies configuration, sets timezone, and creates log directories.
- For Jetson devices (node900--node903), drift compensation is especially important -- they boot frequently and have less stable oscillators than desktop/server hardware.
- The GPS unit on the time host should be verified periodically: confirm it's locking to satellites, not using its internal oscillator as a holdover source. Check this by running ntptime on the time host itself.
- If the time host goes offline and all clients lose sync beyond their maximum correction threshold (typically 128 seconds), systems may need manual time correction before Kerberos/TLS will function again.

14.3 Storage Infrastructure

14.3.1 Active Disk Topology (chonk – August 2026)

All compiled documents and web artifacts target wonderland; chonk serves as local scratch space. Physical topology on chok:

- **/mnt/backup1**: Source directory for Ansible repos, workspace code, and build artifacts
- **/mnt/clusterfs**: Scratch disk for temporary compilation, LaTeX builds, and intermediate files
- **/mnt/snowy**: Bulk storage -- formerly a full NFS share; hard disk physically removed, now mounted as raw drive only

The following mount points have been purged:

- **/mnt/storage1**: Completely removed across all roles and /etc/exports on all hosts
- Legacy /mnt/storage2, /mnt/storage3: Purged from inventory references

14.3.2 NFS Mounts – Current State

The lab relies on NFS for shared storage. Active mounts as of August 2026:

- **stargate:/mnt/clusterfs2**: Primary cluster workspace; mounted on stargate and accessible from wonderland via SSH-sync pipelines
- **Jetson home directories**: Shared from stargate via stargate:/mnt/clusterfs2 -- provides consistent ~/home for node900--node903

14.3.3 StrictModes and Permission Requirements

OpenSSH StrictModes enforces strict permission checks on home directories. Required permissions:

- Home directory: 0750 (rwxr-x---)
- ~/.ssh: 0700 (rwx------)
- ~/.ssh/authorized_keys: 0600 (rw-------)

All Jetson nodes must comply with these permissions or SSH authentication fails silently. NFS-mounted homes require uid/gid consistency between stargate and each Jetson host.

14.3.4 Snapshot and Backup Strategy

The lab uses bare git repos for version control and backup:

- GPG-encrypted repositories synced across hosts for credential protection (Stash House project)
- Pass-based password store with GPG backend
- No LFS -- ABSOLUTE RULE: do not use git-lfs anywhere. Microsoft once held Minecraft chunk files hostage when a repo hit the 10GB GitHub limit with LFS enabled.

14.3.5 LFS Policy Reminder

ABSOLUTE RULE: Do NOT use git-lfs anywhere in any lab-franklin repository.

This applies across all repos: training-ai, minecraft, Ansible collections, everything. The incident with Minecraft chunk files being held hostage by GitHub when the repo hit 10GB with LFS is a real precedent -- never risk it.

File Services: NFS

14.4 What We Know About NFS

14.5 Storage Infrastructure

14.5.1 Active Disk Topology (chonk – August 2026)

All compiled documents and web artifacts target wonderland; chonk serves as local scratch space. Physical topology on chok:

- **/mnt/backup1**: Source directory for Ansible repos, workspace code, and build artifacts
- **/mnt/clusterfs**: Scratch disk for temporary compilation, LaTeX builds, and intermediate files
- **/mnt/snowy**: Bulk storage -- formerly a full NFS share; hard disk physically removed, now mounted as raw drive only

The following mount points have been purged:

- **/mnt/storage1**: Completely removed across all roles and /etc/exports on all hosts
- Legacy /mnt/storage2, /mnt/storage3: Purged from inventory references

14.5.2 NFS Mounts – Current State

The lab relies on NFS for shared storage. Active mounts as of August 2026:

- **stargate:/mnt/clusterfs2**: Primary cluster workspace; mounted on stargate and accessible from wonderland via SSH-sync pipelines
- **Jetson home directories**: Shared from stargate via stargate:/mnt/clusterfs2 -- provides consistent ~/home for node900--node903

14.5.3 StrictModes and Permission Requirements

OpenSSH StrictModes enforces strict permission checks on home directories. Required permissions:

- Home directory: 0750 (rwxr-x---)
- ~/.ssh: 0700 (rwx------)
- ~/.ssh/authorized_keys: 0600 (rw-------)

All Jetson nodes must comply with these permissions or SSH authentication fails silently. NFS-mounted homes require uid/gid consistency between stargate and each Jetson host.

14.5.4 Snapshot and Backup Strategy

The lab uses bare git repos for version control and backup:

- GPG-encrypted repositories synced across hosts for credential protection (Stash House project)
- Pass-based password store with GPG backend
- No LFS -- ABSOLUTE RULE: do not use git-lfs anywhere. Microsoft once held Minecraft chunk files hostage when a repo hit the 10GB GitHub limit with LFS enabled.

14.5.5 LFS Policy Reminder

ABSOLUTE RULE: Do NOT use git-lfs anywhere in any lab-franklin repository.
This applies across all repos: training-ai, minecraft, Ansible collections, everything. The incident with Minecraft chunk files being held hostage by GitHub when the repo hit 10GB with LFS is a real precedent -- never risk it.

14.6 How to Test the NFS Ansible Role

14.7 Overview

The NFS role at ansible/collections/ansible_collections/lab/franklin/roles/nfs has been migrated from Molecule to native ansible-test. The role validates:

- Server-side export configuration (/etc/exports)
- Client-side fstab management and mount points
- Kerberos security validation (sec=krb5i -- though keytabs remain undeployed; sec=krb5i is non-functional without keytab infrastructure)

14.8 Testing with ansible-test (August 2026)

From stargate:

```
cd ~/workspace/lab-franklin/ansible/collections/ansible_collections/lab/franklin
ansible-test sanity --python 3.12
ansible-test integration nfs --python 3.12
```

Molecule tests remain as historical artifacts only and are no longer invoked.

14.9 Current Validation Focus

Testing now emphasizes:

- **Export template rendering**: Jinja2 templates generate valid /etc/exports content for active roles (chonk's /mnt/storage1, storage2, storage3 exports)
- **Client-side mount validation**: fstab line insertion and idmapd.conf deployment on client nodes
- **Jetson home directory consistency**: NFS-mounted homes from stargate:/mnt/clusterfs2 must maintain uid/gid alignment

14.10 StrictModes Validation

NFS-mounted user homes require OpenSSH StrictModes compliance:

- Home directory 0750, .ssh 0700, authorized_keys 0600 -- verified as part of integration tests
- Kerberos keytabs never deployed; sec=krb5i in exports vars is non-functional

14.11 Integration Test Targets

For thorough testing, add:

- Export verification (check showmount -e)
- Mount/unmount cycle tests on client container
- StrictModes permission validation post-mount

14.12 Molecule Test Checklist – DEPRECATED

The following Molecule-specific items are no longer relevant as of August 2026:

DEPRECATED Server-side converge with hostname override

DEPRECATED Client-side converge (mount directory creation)

DEPRECATED Verify phase checks exports file content

DEPRECATED Idempotence test: second converge

All future work uses ansible-test exclusively.

Automation & Tooling

14.13 Ansible Automation

14.13.1 Current Landscape (August 2026)

As of August 2026, the Ansible ecosystem is split between two distribution paths:

- **Ansible community package** -- the traditional monolithic install (pip install ansible). Current version is 13.x (built on ansible-core 2.20). This includes 85+ collections with thousands of modules and plugins bundled together. Follows semantic versioning, releases two major versions per year plus monthly minor releases.
- **ansible-core** -- the minimal core engine. Maintains three versions at a time (current + two older). Versioned independently from the community package (2.19, 2.20, 2.21). Releases every four weeks for minor updates.

EOL versions as of August 2026: Ansible 10--12 and ansible-core 2.15--2.18 are EOL. Do not use these in new work. Only Ansible 13.x / ansible-core 2.20 is current; 14.0.0/2.21 is in development.

14.13.2 Molecule Testing – Deprecated (August 2026)

Molecule has been removed from the active workflow as of August 2026. It no longer serves as a testing target or standard. All role validation now uses:

- **ansible-test --sanity** -- built-in collection sanity checks, run natively on stargate.research.bi with zero external overhead
- **ansible-test --integration** -- integration tests targeting real hosts or containerized environments via ansible-test's native docker driver
- **ansible-playbook --syntax-check** -- YAML structure validation for all playbooks and role task files

Existing Molecule test harnesses (in ansible/collections/ansible_collections/lab/franklin/roles/<r remain as historical artifacts but are no longer invoked or maintained. New roles must use native ansible-test exclusively.

lab/franklin Collection

The lab/franklin Ansible collection lives at:

```
/mnt/clusterfs2/workspace/lab-franklin/ansible/collections/ansible_collections/lab/
    franklin/

```

It is the central automation artifact for the entire lab.bitsmasher.net infrastructure, managing provisioning, configuration, and testing across all hosts.

14.14 Collection Manifest (galaxy.yml)

The collection follows standard Ansible Galaxy conventions with namespace lab and name franklin:

- **Version**: 1.0.0
- **License**: MIT
- **Author**: franklin <franklin@bitsmasher.net>
- **Repository**: https://github.com/devsecfranklin/lab-franklin
- **Documentation**: https://github.com/devsecfranklin/lab-franklin/docs

14.15 Role Inventory

The collection contains roles organized under the roles/ directory. As of August 2026, the following roles are maintained with native ansible-test harnesses:

```
roles/
 apt_mirror/          # APT mirror configuration
 beagleboard/         # BeagleBoard device provisioning
 common/              # Shared base configuration (bootstrap.sh, common utils)
 container_registry/  # Docker/container registry setup
 ctfd/                # CTFd platform deployment
 desktop/             # Desktop environment provisioning
 docker/              # Docker engine installation
 documentation/       # Documentation generation roles
 dhcp/                # DHCP server configuration (dhcpd.conf)
 dns/                 # BIND/named DNS server
 golang/              # Go language toolchain
 jetson-nano/         # NVIDIA Jetson Nano provisioning
 k3s_agent/           # k3s worker node
 k3s_server/          # k3s control plane server
 k8s/                 # Kubernetes generic setup
 latex/               # LaTeX environment
 logging/             # Centralized logging
 music/               # Music service configuration
 nfs/                 # NFS server/client
 nix/                 # Nix package manager
 openbsd/             # OpenBSD host bootstrap (blowfish target)
     tasks/main.yml   # Includes files.yml, timezone setup
```

```
    files/           # bootstrap.sh, login.conf, hosts entries
paloalto/           # Palo Alto firewall config
prereq/             # Prerequisite packages
pypi_internal/      # Internal PyPI repository setup
    tasks/main.yml   # Symlink pypi_dir -> html_dir/pypi
    vars/main.yml    # html_dir=/var/www/html, pypi_dir=/mnt/storage1/LAB/pypi
python/             # Python environment
raspberrypi/        # Raspberry Pi provisioning
samba/              # Samba file sharing
security/           # Hardening and security baselines
shell/              # Shell configuration
ssh/                # SSH server hardening
tls/                # TLS certificate management
chonk/              # chonk-specific configuration
cluster/            # Cluster-wide settings

```

14.16 Workspace Structure

The full project layout:

```
/workspace/lab-franklin/
Makefile.am                  # Top-level autotools target (Python, Docker, dev)
configure.ac                 # Autotools config (autoconf)
bootstrap.sh                 # Cross-platform bootstrapper
network_update.sh            # One-shot maintenance script (hardened 2026-08-02)
ansible/                     # ANSIBLE_HOME root
    playbook.yml              # Main playbook
    hosts                     # Inventory file
    collections/              # Collections path
        ansible_collections/ # Galaxy namespace
            lab/franklin/    # The lab/franklin collection
                galaxy.yml
                roles/       # All role modules (37 total)
                docs/latex/  # LaTeX docs build (autotools)
container/                   # Docker/Podman configs
terraform/                   # Terraform infrastructure definitions
bin/                         # Shared utility scripts (common.sh)
docs/manual/                 # Lab manual documentation (this document)

```

14.17 Build System (Autotools)

The project uses GNU Autotools for the root-level build:

- **configure.ac**: Defines package (lab-franklin v0.2), libtool support, Python $\geq$ 3.9 requirement, podman/ansible/gpg checks, git version tagging
- **Makefile.am**: Top-level targets include test (ansible-lint + ansible-test sanity/integration), security (venv bootstrap), dev (Python venv in BUILDDIR)
- **bootstrap.sh**: Cross-platform bootstrapper that detects OS (Debian/RedHat/OpenBSD/macOS/Linux) and installs platform-specific packages, then runs aclocal $\rightarrow$ autoreconf $\rightarrow$ automake $\rightarrow$ configure
- **docs/manual/Makefile.am**: Standalone autotools stub for LaTeX docs build (clean target removes *.aux files)

The network_update.sh script was hardened on 2026-08-02 with: set -euo pipefail, auto-resolved paths, pinned K3s version validation, ANSIBLE$_{H}OMEenvironmentchecks, removeddangerousclush/apt-getblastradius, anddeadcoderemoval.Itrunsthemainplaybookat$ansible/playbook.yml$asitsprimaryaction.

Testing from Stargate

14.18 Native ansible-test on stargate.research.bitsmasher.net

The stargate host (10.10.16.66, Debian 12 bookworm) serves as the exclusive testing platform for the lab/franklin Ansible collection. All test execution runs with zero external billing -- direct shell access to the gateway sandbox eliminates API token overhead.

14.18.1 Testing Standard: ansible-test (August 2026)

Native ansible-test has replaced Molecule as the canonical testing framework:

- **sanity checks**: Built into ansible-core; validates module/plugin imports, YAML structure, and collection metadata without any container dependency
- **integration tests**: Run via ansible-test integration targeting real hosts or the native docker driver (preferred over podman for consistency)
- **syntax validation**: ansible-playbook --syntax-check validates all playbooks and role task files before execution

The legacy Molecule directory structure at each role's molecule/ subdirectory has been deprecated. It remains in-place as historical reference but is excluded from CI pipelines and the Makefile.am test target.

14.18.2 Test Execution Flow (ansible-test)

For a given role, the native test cycle is:

1. **syntax-check**: ansible-playbook --syntax-check playbook.yml validates YAML structure
2. **sanity**: ansible-test sanity runs built-in linting against the collection -- no container, no overhead
3. **integration**: ansible-test integration <role> runs target-specific tests using docker driver containers

14.18.3 Running Tests from Stargate

From stargate's workspace:

```
cd ~/workspace/lab-franklin/ansible/collections/ansible_collections/lab/franklin
ansible-test sanity
ansible-test integration dns --python 3.12
ansible-playbook --syntax-check playbook.yml

```

14.19 Dynamic Scoping and Pathing Convention

Roles use dynamic file inclusion via templated variables to avoid hardcoded paths:

- Roles include task fragments using: _role_name_files and _role_name_templates for dynamically scoped file/paths
- Template references follow the pattern: {include_task "tasks/_role_name.yml"} scoped within dynamically included tasks

This convention eliminates static path dependencies across role invocations and supports multi-target deployment without playbook-level path overrides.

14.20 Package Standards – Deprecated Utilities

The following deprecated utilities have been removed from baseline role manifests:

- **neofetch**: Removed from common role; replaced by standard Linux tools (lsb_release, uname -r)
- **tripwire**: Removed from security role; no longer maintained and superseded by native file integrity monitoring
- Other legacy packages audited during August 2026 cleanup -- verify role manifests before adding new dependencies

14.21 Dynamic Probing Standard

Static inventory reachability assumptions have been replaced with non-interactive batch probes:

```
ssh -o BatchMode=yes -o ConnectTimeout=3 -i ~/.ssh/id_ed25519_openclaw franklin@<
    host> "hostname && whoami"

```

This approach eliminates stale host entries, prevents password prompts in automation pipelines, and provides deterministic connectivity feedback with a 3-second timeout. Never assume reachability from static notes -- always probe on demand.

14.22 Ansible-Linting Infrastructure

All ansible roles in the lab collection should be linted on a regular basis to maintain configuration quality and catch issues early. The lab uses three tools: ansible-lint, yamllint, and ansible-test sanity.

14.22.1 Tools and Configuration

Each linter is configured in the test/ directory:

- .ansible-lint.yml - ansible-lint configuration with warnings for deprecated modules and experimental rules disabled
- .yamllint / .yamllintrc - YAML linting extends default rules with lab-specific indentation and spacing constraints

14.22.2 Coverage Status (44 Roles)

Based on the most recent audit of ansible/collections/ansible_collections/lab/franklin/roles/<role>:

Complete (30 roles) apt-mirror, beagleboard, chonk, cluster, common, container-registry, ctfd, desktop, dhcp, dns, docker, documentation, golang, k3s-agent, k3s-server, kerberos, ldap, logging, minecraft, music, nfs, nix, ntp, openbsd, paloalto, prereq, pypi-internal, python, raspberrypi, ssh

Minimal/Functional (9 roles) media, samba, shell, tls, website, apt-mirror-stub, jetson-nano, k8s (partial), dns-stub

Stubs (4 roles) edge, extensions, latex, odroid

Roles with the smallest role definitions tend to have the fewest linting issues. The most problematic roles are usually those that include dynamic template generation or custom module invocation.

14.22.3 Proposed Linting Pipeline

A new test/lint_all.sh script should be created to run all three linters in sequence:

```
# Example lint_all.sh structure:
#!/bin/bash
set -euo pipefail

cd "$(dirname "$0")/.."

echo "=== ansible-lint ==="
ansible-lint roles/ 2>&1 || echo "ansible-lint warnings found"

echo "=== yamllint ==="
yamllint -c test/.yamllint roles/ 2>&1 || echo "yamllint issues found"

echo "=== ansible-test sanity ==="
ansible-test sanity --docker default -v 2>&1 || echo "sanity check failures"
```

The linting pipeline should be integrated into:

- **GitHub Actions**: Run on PR and nightly builds (similar to existing bandit/trivy/tfsec workflows)
- **Hourly cron job on stargate**: Optional for CI-style validation without external triggers

14.22.4 Linting Exceptions by Role

Some roles require special handling during linting:

- tls: Includes shell commands with kss -kfe which may trigger warning 403 (deprecated module) - explicitly skip in config
- samba: Uses custom smb.conf.j2 templates that require validation via testparm post-converge
- k8s: Contains static manifests rather than role tasks - should be moved to a shared files directory or documented as a reference role

Terraform Infrastructure Management

Terraform configurations in this lab manage cloud infrastructure across two providers: DigitalOcean and Google Cloud Platform. All configurations are version-controlled and include automated security scanning via GitHub Actions.

14.23 DigitalOcean Infrastructure

The terraform/digital_ocean/ directory manages the external-facing web presence for bitsmasher.net.

14.23.1 Provisioned Resources

- **Droplet:** 512MB Debian 12 in lon1 region (www.bitsmasher.net)
- **Domain:** bitsmasher.net with full DNS management
- **DNS Records:** A, MX, TXT (SPF/DMARC/DKIM), NS, and ACME challenge records

14.23.2 Key DNS Records

Name	Type	Purpose
www.bitsmasher.net	A	Web server host (178.62.60.55)
@	MX	mail.protonmail.ch 10 (ProtonMail routing)
@	TXT	SPF and DMARC policies
protonmail._domainkey	CNAME	DKIM email verification
_acme-challenge.*.lab	TXT	Let's Encrypt certificate challenges

14.23.3 Configuration Notes

The droplet includes:

- Automated backups enabled (immutable against accidental deletion)
- SSH key injection via terraform data source
- No external monitoring (consider enabling DO agent for uptime tracking)

Provider authentication uses a DigitalOcean API token managed through pass/direnv integration. The token is passed at runtime, never committed to the repository.

14.24 Google Cloud Platform Infrastructure

The terraform/google/ directory manages GCP resources for Stash House free-tier operations.

14.24.1 Resources Managed

- **VPC Network:** stash-house-vpc dedicated network
- **Compute Instance:** e2-micro Debian 12 (Always Free Tier eligible) in us-central1-a
- **Firewall Rule:** allow-ssh opening port 22 to the VPC
- **Billing Budget Alert:** Tripwire alert at $0.01 threshold for free-tier monitoring

The GCP configuration is intentionally minimal, designed around Google's Always Free Tier limits to maintain zero-cost infrastructure.

14.25 Security Scanning

All Terraform configurations are automatically scanned by two GitHub Actions workflows:

- tfsec-sarif.yml - IaC security scanning (generates SARIF reports)
- tfsec-comment.yml - Posts security findings as PR comments

Additionally, the DigitalOcean directory includes existing README.md with doctl and pass integration instructions for manual operations.

14.26 Testing

Terraform configurations are validated through:

- Go-based integration tests (terraform/test/main_test.go)
- Plan dry-run validation via GitHub Actions (markdown.yml workflow)
- Manual state reconciliation checks (compare terraform state against live infrastructure)

State drift detection is recommended as a regular maintenance task - compare terraform state list output against actual cloud resource inventories quarterly.