lab.bitsmasher.net Manual

franklin

August 2026

Contents

1 Introduction **7**
1.1 Purpose and Scope . . . 7

2 Hardware Inventory **9**
2.1 Desk Setup . . . 9
2.2 Hardware Inventory . . . 9
2.2.1 Desk Setup . . . 9
2.2.2 Server and Compute Hosts . . . 9
2.2.3 Historical / Decommissioned Hosts . . . 10
2.2.4 Planned Hardware Expansion . . . 10

3 Network Infrastructure **11**
3.1 Addressing and Subnets . . . 11
3.2 Network Infrastructure . . . 11
3.2.1 Addressing Scheme . . . 11
3.2.2 DHCP . . . 11
3.2.3 IPv6 Architecture . . . 11
3.2.4 DNS Architecture . . . 12
3.2.5 Firewall and Access Control . . . 12
3.2.6 Physical Topology . . . 12
3.2.7 Logging Infrastructure . . . 12
3.3 DNS . . . 12
3.4 DNS Configuration . . . 13
3.4.1 BIND Overview – Post-Migration (August 2026) . . . 13
3.4.2 Zone and Host Realignment (August 2026) . . . 13
3.4.3 Kerberos SRV RFC Compliance . . . 13
3.4.4 DNS Failure Impact . . . 13
3.4.5 DNS Maintenance Procedures . . . 13

4 Core Services **15**
4.1 Network Time (NTP) . . . 15
4.2 Network Time (NTP) . . . 15
4.2.1 Overview . . . 15
4.2.2 The Time Host . . . 15
4.2.3 Configuration on Client Hosts . . . 15
4.2.4 Installation and Setup Steps . . . 16
4.2.5 Verification on Clients . . . 16
4.2.6 Notes for Deployment . . . 17

5 File Services **19**
5.1 Samba AD Domain Controller . . . 19
5.1.1 Deployment Plan . . . 19
5.1.2 Migration Plan (parallel with odroid-c1) . . . 20
5.1.3 Risks & Mitigations . . . 20
5.2 NFS Mounts and Backup . . . 21
5.3 Kerberos Authentication . . . 21
5.4 Kerberos Authentication . . . 21

5.4.1 Realm Configuration . . . 21
5.4.2 Access and Key Management . . . 21
5.4.3 Principal Conventions . . . 21
5.4.4 DNS/Kerberos Integration – RFC Compliant (August 2026) . . . 21
5.4.5 Keytab Management . . . 22
5.5 LDAP Directory Services . . . 22
5.6 LDAP Directory Services . . . 22
5.6.1 Server Configuration . . . 22
5.6.2 Outage Details . . . 22
5.6.3 Monitoring Approach . . . 22
5.6.4 Recovery Steps . . . 22
5.6.5 Directory Structure (Expected) . . . 23

6 Security **25**
6.1 Authentication and Access Control . . . 25
6.2 Network Security . . . 25
6.3 Certificate and TLS Management . . . 25
6.4 Credential Management . . . 25
6.5 Backup and Disaster Recovery . . . 25
6.6 Container and Kubernetes Security . . . 26
6.7 Automated Security Scanning . . . 26

7 Storage **27**
7.1 NFS Mounts and Backup . . . 27
7.2 Storage Infrastructure . . . 27
7.2.1 Active Disk Topology (chonk – August 2026) . . . 27
7.2.2 NFS Mounts – Current State . . . 27
7.2.3 Purged Storage (Historical Reference) . . . 27
7.2.4 StrictModes and Permission Requirements . . . 28
7.2.5 Snapshot and Backup Strategy . . . 28
7.2.6 LFS Policy Reminder . . . 28
7.3 LFS Policy . . . 28

8 Internal PyPI Repository **29**
8.1 Overview . . . 29
8.2 Internal PyPI Repository . . . 29
8.2.1 Architecture . . . 29
8.2.2 Configuration via Ansible . . . 29
8.2.3 Usage from Client Machines . . . 30
8.2.4 Package Maintenance . . . 30

9 Container Orchestration **31**
9.1 k3s Cluster . . . 31
9.2 Kubernetes (k3s) Cluster . . . 31
9.2.1 Architecture . . . 31
9.2.2 Version Pinning . . . 31
9.2.3 Container Runtime . . . 31
9.2.4 Cluster Topology . . . 32
9.2.5 kubeconfig Access . . . 32

10 Playbook Automation **33**
10.1 Ansible Automation . . . 33
10.1.1 Current Landscape (August 2026) . . . 33
10.1.2 Molecule Testing – Deprecated (August 2026) . . . 33

11 lab/franklin Collection **35**
11.1 Collection Manifest (galaxy.yml) . . . 35
11.2 Role Inventory . . . 35
11.3 Workspace Structure . . . 36
11.4 Build System (Autotools) . . . 36

12 Testing from Stargate **39**
12.1 Native ansible-test on stargate.research.bitsmasher.net . . . 39
12.1.1 Testing Standard: ansible-test (August 2026) . . . 39
12.1.2 Test Execution Flow (ansible-test) . . . 39
12.1.3 Running Tests from Stargate . . . 39
12.2 Dynamic Scoping and Pathing Convention . . . 40
12.3 Package Standards – Deprecated Utilities . . . 40
12.4 Dynamic Probing Standard . . . 40
12.5 Centralized Logging Infrastructure . . . 40
12.5.1 Ingestion & Protocol Architecture . . . 40
12.5.2 Client Forwarding Architecture . . . 41
12.5.3 Log Retention & Rotation Policy . . . 41
12.6 Blowfish – Database Host . . . 41
12.6.1 Host Identity . . . 41
12.6.2 Role in the Lab . . . 41
12.6.3 Current Status . . . 41
12.6.4 Planned Configuration (OpenBSD) . . . 42
12.7 Minecraft Server . . . 42
12.7.1 Host and Location . . . 42
12.7.2 Current Configuration . . . 42
12.7.3 Core Plugins . . . 42
12.7.4 Interactive Map . . . 42
12.7.5 Storage . . . 42
12.7.6 Administrative Access . . . 43
12.8 Operational Guardrails & Publishing Standards . . . 43
12.8.1 The No-Git Law . . . 43
12.8.2 Web Publishing Standards . . . 43
12.8.3 ISMG-2050 Quality Gate . . . 43
12.8.4 Canvas as Single Source of Truth . . . 43

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	**logging.lab.bitsmasher.net** — repurposed from node3; 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 (Jetson Nano 2GB L4T)
node901	10.10.12.91	Online, key auth; home dir from stargate:/mnt/clusterfs2 (Jetson Nano 2GB L4T)
node902	10.10.12.92	Offline/decommissioned
node903	10.10.12.93	Offline/decommissioned

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

AI accelerator division: Coral Edge TPU on **edge-t**; Jetson Nano 2GB L4T on node900–node901; CTF/Ollama compute on **puzzle.lab.bitsmasher.net** (10.10.12.1).

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

- **head2** (Retired August 2026; released IPs 10.10.12.14 / 10.10.12.19; NFS exports retired)
- **server1** (10.10.12.12; former BIND DNS master and OpenLDAP master; decommissioned)
- **ns1** (10.10.12.12; former authoritative BIND DNS server; decommissioned)
- **snowy** (Retired; media staging disks pulled; NFS paths retired)
- **thelio, femputer, node5, node6, server3, netlab1, netlab2** (Legacy compute and lab nodes retired in earlier phases)

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	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

The legacy `ns1` (10.10.12.12) has been decommissioned; it no longer appears in any active allocation.

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**: 10.10.12.15, MAC 98:b7:85:21:ad:77 – database host (updated from 10.10.8.15)
- **femputer**: 10.10.15.1, MAC d8:bb:c1:af:21:17 – guest/subnet-aligned

3.2.3 IPv6 Architecture

The lab uses a Hurricane Electric 6in4 SIT tunnel (Tunnel ID: 1039838) for IPv6 connectivity:

- **Upstream WAN**: 136.29.105.20 (eth8 on UDM-SE) → HE Denver 184.105.250.46
- **Transit prefix**: 2001:470:39:294::/64; client ::2/64, server ::1/64, MTU 1480

- **Routed prefix**: 2001:470:4b:294::/64 distributed downstream via SLAAC

UDM-SE persistence is managed by `/etc/systemd/system/he-ipv6.service`. Fleet configuration across lab hosts uses the `lab.franklin.network` role with NetworkManager auto-SLAAC.

3.2.4 DNS Architecture

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

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

The authoritative BIND9 master is **logging.lab.bitsmasher.net** (10.10.12.3), formerly node3, handling both forward and reverse zone transfers following the server1/ns1 decommissioning in August 2026.

Kerberos/KDC authority remains strictly on **odroid-c1** (10.10.12.254). The legacy ns1 host has been fully decommissioned and purged from active allocations.

3.2.5 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
- Jetson cluster edge nodes (node900–node903) require port 514 UDP/TCP opened on logging.lab.bitsmasher.net for syslog forwarders

3.2.6 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 (10.10.12.90, online L2), node901 (10.10.12.91, online L2); node902 (10.10.12.92) and node903 (10.10.12.93) offline/decommissioned
- 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.

3.2.7 Logging Infrastructure

Edge log collection uses a pipeline: syslog forwarders on edge hosts → port 514 UDP/TCP ingress on logging.lab.bitsmasher.net (10.10.12.3) → Alloy → Loki → Grafana dashboard (bound to 127.0.0.1:3000 locally).

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 hosts**: `server1` (10.10.12.12) and `ns1` (10.10.12.12) have 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 $_kerberos-adm._tcp$ record now points directly to the A record for `kdc1.lab.bitsmasher.net` rather than via a C

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.

Chapter 5

File Services

5.1 Samba AD Domain Controller

The lab's Samba AD domain controller runs on chonk (10.10.8.60), providing Kerberos/LDAP backend, DNS forwarding, and file shares. This replaces/supplements odroid-c1 (KDC) and ldap.bbb1 (OpenLDAP) which are currently degraded.

5.1.1 Deployment Plan

Phase 1 --- Preparation (chonk host)

```
# Install Samba with AD support
sudo apt install -y samba samba-common-bin krb5-user ldb-tools python3-ldap

# Verify kernel ACL support (required for AD)
lsmod | grep acl   # should show acl or krad_acl

# Create share directories
sudo mkdir -p /srv/samba/shares/{public,backup,devs}
sudo chown root:root /srv/samba/shares
sudo chmod 0775 /srv/samba/shares
```

Phase 2 --- Samba AD Domain Provisioning (run once)

```
# Set the realm/domain name for lab bitsmasher
export SAMBA_REALM="LAB.BITSMASHER.NET"
export SAMBA_DOMAIN="LAB"
export SAMBA_ADMIN_PASS="<set-strong-password>"

sudo samba-tool domain provision \
  --realm=\${SAMBA_REALM} \
  --domain=\${SAMBA_DOMAIN} \
  --server-role=dc \
  --dns-backend=BIND9_TDB \
  --option="interfaces=lo 10.10.8.0/21" \
  --option="bind interfaces only=yes"

# Verify the configuration
cat /var/lib/samba/private/smb.conf   # backup this!
```

Phase 3 --- DNS Integration The provisioned Samba installs a local BIND instance at /var/lib/samba
Configure chonk to forward external queries to time.lab.bitsmasher.net.

Phase 4 --- Kerberos Configuration Update /etc/krb5.conf on all lab hosts:

```
[libdefaults]
    default_realm = LAB.BITSMASHER.NET
```

```
    dns_lookup_realm = false
    dns_lookup_kdc = true

[realms]
    LAB.BITSMASHER.NET = {
        kdc = chonk.lab.bitsmasher.net:88
        admin_server = chonk.lab.bitsmasher.net:749
    }

[domain_realm]
    .bitsmasher.net = LAB.BITSMASHER.NET
    bitsmasher.net = LAB.BITSMASHER.NET
```

Phase 5 --- Samba Service Setup (via Ansible) Use the existing lab.franklin.samba role to manage shares and users after AD provisioning:

```
- hosts: chonk.lab.bitsmasher.net
  become: true
  vars:
    samba_shares:
      - name: public
        path: /srv/samba/shares/public
        valid_users: franklin,sly
        read_only: no
      - name: backup1
        path: /srv/samba/shares/backup1
        valid_users: franklin
        read_only: no
    samba_create_varwww_symlinks: false
    samba_interfaces: 127.0.0.0/8 10.10.8.0/21
  roles:
    - lab.franklin.samba
```

Phase 6 --- File Share Permissions (post-provision)

```
sudo chown -R root:root /srv/samba/shares
sudo find /srv/samba/shares -type d -exec chmod 0775 {} \;
sudo find /srv/samba/shares -type f -exec chmod 0664 {} \;
sudo setfacl -R -m u::rwx,g::rx,o::- /srv/samba/shares
```

Phase 7 --- Testing & Migration Test the new AD domain controller:

1. kinit franklin@LAB.BITSMASHER.NET from another lab host → should succeed
2. Mount a share: mount -t cifs //chonk/public /mnt/smb -o user=franklin
3. Verify DNS resolution: nslookup chonk.lab.bitsmasher.net

5.1.2 Migration Plan (parallel with odroid-c1)

- Keep odroid-c1 running as the primary KDC for now
- Point new clients at chonk's Samba AD
- If odroid-c1 goes down, chonk takes over immediately
- Once confident in chonk's stability, decommission odroid-c1

5.1.3 Risks & Mitigations

Risk	Mitigation
DNS conflicts with BIND on time	Use Samba DNS only for .bitsmasher.net and lab.bitsmasher.n
SELinux blocks shares (Debian)	Skip --- Debian doesn't ship with SELinux by default
KDC unavailability during provisioning	odroid-c1 remains active until chonk is validated
Disk usage on chonk	Monitor /var/lib/samba/ size quarterly

5.2 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).

5.3 Kerberos Authentication

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

5.4 Kerberos Authentication

5.4.1 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)

5.4.2 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).

5.4.3 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

5.4.4 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 $_kerberos-adm._tcp$ record now points directly to the A record for kdc1.lab.bitsmasher.net, per RFC 2782 compliance.

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.bitsmasher.net in /etc/krb5.conf.

5.4.5 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.

5.5 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.

5.6 LDAP Directory Services

5.6.1 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**

5.6.2 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.

5.6.3 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.

5.6.4 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

5.6.5 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 6

Security

6.1 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.

6.2 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.

6.3 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.

6.4 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.

6.5 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.

6.6 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.

6.7 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 7

Storage

7.1 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).

7.2 Storage Infrastructure

7.2.1 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

7.2.2 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. NFS server: nfs-kernel-server exporting local NVMe (/dev/nvme0n1p1, 469 GB ext4) to cluster subnets 10.10.16.0/21 and 10.10.8.0/21.
- **chonk:/mnt/backup1**: Active NFS export host serving cluster-wide storage to managed hosts
- **Jetson home directories**: Shared from stargate via stargate:/mnt/clusterfs2 -- provides consistent ~/home for node900--node903

7.2.3 Purged Storage (Historical Reference)

The following mounts have been decommissioned and purged from all roles and inventory:

- **/mnt/storage1**: Completely removed across all roles and /etc/exports on all hosts (August 2026)

  - **Legacy /mnt/storage2, /mnt/storage3**: Purged from inventory references (August 2026)
  - **/mnt/snowy**: Formerly a full NFS share; hard disk physically removed, now mounted as raw drive only
  - **head2 exports**: All exports on head2 retired with node decommission

7.2.4 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.

7.2.5 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.

7.2.6 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.

7.3 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 8

Internal PyPI Repository

8.1 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.

8.2 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.

8.2.1 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.

8.2.2 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

8.2.3 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

```

8.2.4 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 9

Container Orchestration

9.1 k3s Cluster

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

9.2 Kubernetes (k3s) Cluster

9.2.1 Architecture

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

- **k3s_server**: Master node running k3s control plane -- target host not yet assigned (planned for Jetson cluster)
- **k3s_agent**: Worker nodes that register with the server and receive workloads
- **GPU nodes**: Specialized workers for NVIDIA container runtime and compute workloads

Operational Status

Control plane orchestration has remained unassigned/pending since the decommissioning of head2 in August 2026. The target host k3s.lab.bitsmasher.net is not yet provisioned; worker nodes are reserved but no control plane has been installed.

9.2.2 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

```

9.2.3 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

9.2.4 Cluster Topology

Node Role	Target Hosts	Notes
k3s_server	k3s.lab.bitsmasher.net (pending)	Control plane; unassigned since head2 decommissioning (August 2026)
k3s_agent	node900, node901, node903	Jetson nodes as workers
nvidia_nodes	GPU-equipped hosts	Container Toolkit + k3s agent

9.2.5 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 10

Playbook Automation

10.1 Ansible Automation

10.1.1 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.

10.1.2 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 11

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.

11.1 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

11.2 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:

The canonical count is **37 active play-ready roles** across 52 total role directories (including deprecated and legacy).

```
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
    meta/main.yml    #   MISSING - pending contract scaffolding
    argument_specs.yml  #   MISSING - pending contract scaffolding
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

```

11.3 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)

```

11.4 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 → autoreconf → automake → 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$_{HOME}$ 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 12

Testing from Stargate

12.1 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.

12.1.1 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.

12.1.2 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

12.1.3 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

```

12.2 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.

12.3 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

12.4 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.

12.5 Centralized Logging Infrastructure

The lab utilizes a centralized telemetry and logging architecture hosted on logging.lab.bitsmasher.n (10.10.12.3, formerly node3). The host operates as an authoritative log receiver, storage aggregator, and telemetry indexer for edge nodes and appliances across the network.

12.5.1 Ingestion & Protocol Architecture

Central log collection is handled via a dual-mode Rsyslog receiver daemon:

- **UDP 514**: Fast datagram ingestion for network appliances (such as the PA-220 firewall) and embedded targets.
- **TCP 514**: Reliable stream ingestion with connection tracking for fleet servers and continuous telemetry.
- **Storage Path**: Remote logs are isolated from local system logs and dynamically written to: /var/log/remote/%HOSTNAME%/%$YEAR%-%$MONTH%-%$DAY%.log using native Rsyslog dynaFile generation.
- **Web UI Security**: Grafana OSS (port 3000) is deployed locally and bound strictly to 127.0.0.1:3000 to prevent unauthenticated network exposure. Remote access is achieved via SSH local port forwarding.

12.5.2 Client Forwarding Architecture

Fleet forwarders (including puzzle.lab.bitsmasher.net and the Jetson cluster nodes node900, node901, and node903) ship telemetry using the lab.franklin.logging role configured in client mode. To ensure resilience against transient network partitions or logging server restarts, forwarders utilize disk-assisted in-memory queueing:

- $ActionQueueType LinkedList
- $ActionQueueFileName fwdRule1
- $ActionResumeRetryCount -1
- $ActionQueueSaveOnShutdown on

Note: node902 is currently offline and excluded from active forwarding plays.

12.5.3 Log Retention & Rotation Policy

Centralized storage growth is managed by logrotate at /etc/logrotate.d/central-syslog:

- **Cadence**: Daily evaluation across /var/log/remote/*/*.log.
- **Retention**: 14 days of retention with compression enabled (compress, delaycompress).
- **Post-rotate**: Reloads daemon handles via /usr/lib/rsyslog/rsyslog-rotate.

12.6 Blowfish – Database Host

12.6.1 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)

12.6.2 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.

12.6.3 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.

12.6.4 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.

12.7 Minecraft Server

12.7.1 Host and Location

The Minecraft server (Dreamland) runs on **chonk.lab.bitsmasher.net** (10.10.8.60), a primary compute host with RX 9070 / Navi 48 GPU. Server files are at /home/minecraft.

Previous host was skynet.lab.bitsmasher.net (10.10.16.10) during the CraftBukkit/early Paper era.

12.7.2 Current Configuration

The server runs **Paper** 1.21.11 on Java 21 (Debian Bookworm). Key settings:

- Gamemode: Survival
- Difficulty: Normal
- Max players: 69
- PvP: Disabled
- View distance: 10 (simulation distance: 12)
- Server port: 25565

The server name is *Dreamland* (MOTD: &aDreamland&7 | &fEst. 2012).

12.7.3 Core Plugins

- **EssentialsX** -- Core, Chat, AntiBuild
- **LuckPerms** -- Permissions (backed by MariaDB)
- **CoreProtect** -- Audit/log (backed by MySQL)

12.7.4 Interactive Map

BlueMap provides an interactive 3D web map: https://chonk.lab.bitsmasher.net:8100

12.7.5 Storage

World data is approximately 13 GB. BlueMap assets are 6.5 GB. All files on local disk (no NFS dependency for world data).

NO GIT-LFS: Chunk files and world data must not use git-lfs. Use bare git repos for any versioned world metadata.

12.7.6 Administrative Access

Operations automation via start.sh with a custom rcon-based controller. Management infrastructure ports: server 25565, BlueMap 8100, Plan analytics 8804.

Ops team: theDevilsVoice (level 4), Perryman (level 4).

Whitelist: slyborg4realz, UnvaluedShoe79, BeefyGamer76y, some$_{g}arlic, KindredFiori, secove, DrewPellino$

12.8 Operational Guardrails & Publishing Standards

The Bitsmasher Lab monorepo operates under strict operational constraints that govern all agent-assisted work. These rules apply across chonk (local authoring), stargate (Ansible control node and local NVMe storage at /mnt/clusterfs2), and wonderland (production edge web host on DigitalOcean VPS).

12.8.1 The No-Git Law

Agents may draft, lint, stage, and compile locally. All version-control mutations-git add, git commit, git push, git checkout, git reset, git stash-are exclusively Franklin's authority. This boundary prevents workspace drift from automated pushes that skip human review.

12.8.2 Web Publishing Standards

When staging or deploying content to wonderland, the following standards are mandatory:

- File ownership: franklin:franklin
- Directory permissions: 0755
- File permissions: 0644
- No raw binary scratch files (.docx) or prompt notes (.txt) in public endpoints; convert to semantic HTML
- External student data import files (.txt, .csv) are permitted

All CSS must use absolute links (no relative paths). The site follows the w3-black palette. Inline style="..." is forbidden on all public pages; styles go in external stylesheets only.

12.8.3 ISMG-2050 Quality Gate

Before any deployment to wonderland, validate with:

```
cd ~/workspace/lab-franklin/ansible && \
ansible-playbook -i inventory/hosts site.yml --tags documentation,audit -v
```

The audit verifies: zero unescaped Unicode literals, correct canonical stylesheet links (/css/www.bitsmasher.net.css), Schema.org JSON-LD presence, and flags deprecated labels. The audit tag is strictly non-mutating.

12.8.4 Canvas as Single Source of Truth

Canvas LMS is authoritative for all course deadlines, module schedules, and late submission penalties. All public web pages must defer to Canvas dates; no hardcoded due dates are permitted.