WEEK 1 OPERATOR QUICK REFERENCE

CLI | systemd | Network Config | Diagnostics

YOUR CELL IS YOUR RESPONSIBILITY

The rule: one command answers one question.

Use these workflows to keep the cell up, connected, and recoverable during labs and incidents.

2AM WEBSITE DOWN FLOW

```
tmux -> systemctl -> journalctl -> ss -> curl
```

MATRIX HOUSE OPERATOR RULES

- Diagnose before fixing
- Start at the bottom of the stack
- Use tmux for every SSH incident
- Use netplan try when remote
- Tight tcpdump filters, always

Layer	Question	Tool
Application	Responding?	curl
Transport	Port open?	ss
DNS	Name resolves?	dig
Network	Route works?	`ip route | ping`
Link	Interface up?	ip link
Physical	Cable/NIC?	`ethtool | dmesg`

Problem	First Question	Command
Service down	Is it running?	systemctl status nginx
No logs?	What changed?	`journalctl -u nginx -n 50`
Port conflict	Who owns it?	`ss -tlnp`
DNS fails	Resolver or zone?	dig +trace name
No route	Path present?	ip route
Packet mystery	Did it arrive?	tcpdump host X and port Y

Daily First Commands	Use
tmux new -s incident	Protect remote session
ip -br addr && ip route	Check identity and path
systemctl status service	Check service state
journalctl -u service -n 50	Read latest evidence
ss -tlnp	Find listening ports

Operator Shortcut	Meaning
systemctl enable --now nginx	start now + boot later
netplan try	safe remote network test
dig +trace name	walk DNS chain
tcpdump 'host X and port Y'	tight packet capture
curl -I https://localhost	verify app response

Week 1 Goal: reduce chaos into evidence, then evidence into action.

CLI OPERATIONS

Small tools. Text streams. Pipes. Compose the answer.

grep - find lines

Filters matching lines. Read-only. First tool for logs.

```
grep ERROR /var/log/syslog
```

sed - edit lines

Substitute or delete text in a stream or file.

```
sed -i 's/debug=true/debug=false/' app.conf
```

awk - columns + math

Tiny language for fields, conditions, and totals.

```
awk '$9==500 {sum+=$10} END{print sum}' access.log
```

pipeline synthesis

Collect -> filter -> extract -> group -> count -> rank.

```
journalctl | grep | awk | sort | uniq -c | head
```

Command	Common Use	Example
ps -ef	List processes	`ps -ef`
grep	Filter lines	grep nginx
awk	Pick columns / math	awk '{print $2}'
xargs	Run command on input	xargs kill
sort	Group duplicates	sort
uniq -c	Collapse + count	`uniq -c`
head	Limit output	`head -10`
cut	Slice fields	`cut -d: -f1`

Applied Pipeline: Failed Auth Analysis

10,000 sshd logs → 450 failed logins → 450 source IPs → 75 unique → Top 10 attackers

```
journalctl -u ssh --since "1 hour ago" | grep "Failed password" | awk '{print $11}'
sort | uniq -c | sort -rn | head -10
```

Need	Command	Meaning
Pause foreground	Ctrl+Z	Stopped, not killed
See jobs	jobs	Shell job table
Run in background	bg %1	Prompt returns
Bring back	fg %1	Foreground again
Detach	disown %1	No shell hangup
Start detached-ish	cmd &	Background immediately
Persistent SSH	tmux new -s incident	Survives disconnect

CLI OPERATOR RULE

- grep finds, sed modifies, awk analyzes
- Job control is for reclaiming your prompt
- If work must survive SSH, use tmux or systemd-run
- Pipelines reduce data until the answer is obvious

SYSTEMD OPERATIONS

systemd is the service orchestration layer. systemctl is the remote control.

Runtime State - right now	Effect
systemctl status nginx	Read health
systemctl start nginx	Launch now
systemctl stop nginx	Terminate now
systemctl restart nginx	Stop + start

Anatomy of a .service File

```
[Unit]
Description=Hexworth API
After=network-online.target postgresql.service

[Service]
User=api
ExecStart=/opt/api/bin/server
Restart=on-failure
RestartSec=5s

[Install]
WantedBy=multi-user.target
```

SYSTEMD GOTCHA

- Edit a unit file? Run systemctl daemon-reload before restart
- Enable controls boot wiring; start controls runtime state
- Timers matter when the job matters

Boot Wiring - next reboot	Effect
systemctl enable nginx	Start at boot
systemctl disable nginx	Do not start at boot
systemctl enable --now nginx	Enable + start now
systemctl daemon-reload	Reload unit cache after edits
systemctl list-timers	Show scheduled timers

journalctl Filter	Meaning
journalctl -u nginx	Only nginx logs
journalctl -u nginx -p err	Errors only
journalctl -fu nginx	Follow live
journalctl -u nginx -n 50	Last 50 lines
journalctl -b 0	Current boot
journalctl -b -1	Previous boot
journalctl -k	Kernel messages
--since "1 hour ago"	Time window

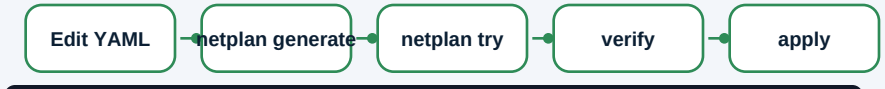
cron	systemd timer
Runs commands	Runs services
Logs to mail / nowhere	Logs in journalctl
Missed jobs skipped	Persistent=true catches up
No dependencies	Requires=/After=
No cgroup limits	CPUQuota/MemoryMax

Configuration determines whether the cell can reach the grid.

Question	Modern Command	Legacy Replaced
Who am I?	ip -br addr / ip addr	ifconfig
Where can I go?	ip route	route
Who is nearby?	ip neigh	`arp -an`
Is link up?	ip link	ifconfig up/down
Tunnels?	ip tunnel	iptunnel
VLANs?	ip link add ... type vlan	vconfig

Decision	Backend
Desktop / laptop / Wi-Fi / VPN / GUI	NetworkManager
Server / headless / container	`systemd-networkd`
Ubuntu source of truth	/etc/netplan/*.yaml
Do not edit generated files	Netplan overwrites them

Netplan: YAML In, Backend Config Out

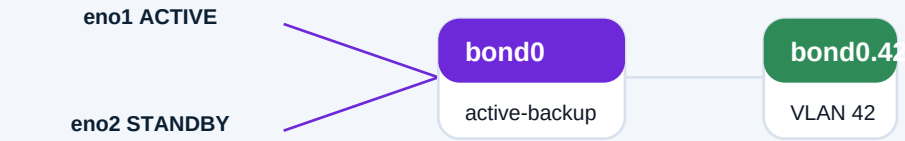



```
network:
  version: 2
  renderer: networkd
  ethernets:
    ens18:
      addresses: [10.0.0.42/24]
      routes: [{to: default, via: 10.0.0.1}]
      nameservers: {addresses: [1.1.1.1]}
```

NETWORK CONFIG OPERATOR RULES

- ip changes are immediate but often ephemeral
- Netplan makes IP changes persistent
- Use netplan try over SSH, not apply
- NetworkManager = roaming desktop; networkd = server
- Use LACP only if the switch is configured for it

Bonding + VLANs: Availability + Segmentation



active-backup: any switch | 802.3ad/LACP: switch must support it

NETWORK DIAGNOSTICS

When the network breaks, do not guess. Walk the layers.

Layer	Question	Tools
Physical	Cable/NIC detected?	ethtool, dmesg
Link	Interface/ARP OK?	ip link, ip neigh
IP/Route	Address/path present?	ip addr, ip route, ping
DNS/Port	Name + socket OK?	dig, ss
Application	Does it respond?	curl

Tool	Question	Example
ping	Is it alive?	`ping -c 4 10.0.0.1`
traceroute	Which path?	`traceroute -n hexworth.com`
mtr	Live path + loss	mtr hexworth.com
dig +short	Just the answer	dig +short hexworth.com
dig +trace	Walk DNS chain	dig +trace hexworth.com
dig @1.1.1.1	Ask resolver directly	dig @1.1.1.1 hexworth.com

ss Question	Command
What is listening?	`ss -tlnp`
All listeners?	`ss -tulnp`
Who is connected?	`ss -tn state established`
HTTPS clients only?	`ss -tn state established '( dport = :443 )'`
Top talkers?	`ss established | awk | sort | uniq -c`

tcpdump Need	Command
Pick interface	`tcpdump -i eth0`
All interfaces	`tcpdump -i any`
Filter host+port	tcpdump 'host 10.0.0.5 and port 443'
Limit packets	`tcpdump -c 1000`
Save pcap	`tcpdump -w cap.pcap`
DNS capture	`tcpdump -i eth0 'port 53'`

TCPDUMP PRODUCTION WARNING

- Tight BPF filters are mandatory on busy hosts
- Use -c to avoid endless captures
- Use -w for Wireshark analysis
- Capture narrow, capture short, capture with purpose

DNS MENTAL MODEL

- dig +short gives the answer
- dig +trace shows root -> TLD -> authoritative
- @resolver bypasses local resolv.conf
- Compare 1.1.1.1, 8.8.8.8, and corporate DNS

INCIDENT PLAYBOOKS

Common failure chains and the command order that finds them

YOUR CELL IS YOUR RESPONSIBILITY

Playbooks keep operators from guessing under pressure.

Website Down

1. `tmux new -s incident`
2. `systemctl status nginx`
3. `journalctl -u nginx -n 30`
4. `ss -tlnp`
5. `curl -I https://localhost`

Cannot Reach Gateway

1. `ethtool eth0`
2. `ip link`
3. `ip -br addr`
4. `ip route`
5. `ping gateway`

DNS Failure

1. `dig name`
2. `dig @1.1.1.1 name`
3. `dig @8.8.8.8 name`
4. `dig +trace name`
5. `fix resolver or delegation`

Port Conflict

1. `systemctl status service`
2. `journalctl -u service`
3. `ss -tlnp`
4. `kill rogue PID / fix config`
5. `systemctl restart service`

Professional administrators do not guess. They gather evidence until the failure becomes obvious.

Your cell is your responsibility. The grid depends on you.