Network Blog

- What a Disconnected Switch Taught Me About Disaster Recovery

What a Disconnected Switch Taught Me About Disaster Recovery

Project Oyster Lab Notes | Incident OYSTER-INC-001

The Incident

One of the easiest ways to learn about infrastructure is to have something break.

During a recent session working on Project Oyster, a switch was disconnected from the firewall. The result was a network outage that took down access to the services running on my Proxmox host.

It was a simple physical change with a much bigger impact than expected. My virtual machines and Docker applications depended on the network being there, so losing that connection made the entire environment appear to disappear.

It also gave me a useful opportunity to work through a real recovery instead of simply reading about one.

Getting the Network Back

After restoring the physical connection, most of the environment recovered. That was reassuring, but it was not the end of the investigation.

I checked the remaining services and found that FreePBX was reachable, yet telephony still was not working. When I checked the Asterisk service, it was inactive.

A manual start brought it back:

```
sudo systemctl start asterisk
```

That small detail turned a basic connectivity outage into a more useful troubleshooting exercise.

A Running VM Is Not a Healthy Service

The most important lesson was that infrastructure has several layers of health.

A physical host can be running while a virtual machine is down. A VM can be running while its operating system is unhealthy. An operating system can be reachable while the application it hosts has stopped.

In this case, being able to reach the FreePBX VM did not mean the PBX was ready to handle calls.

That distinction matters when building monitoring and recovery procedures. A ping test can be useful, but it cannot tell me whether Asterisk is active or whether two phones can successfully complete a call.

What I Want to Improve

The incident highlighted a few improvements for Project Oyster:

- **Document physical dependencies.** Firewall-to-switch links and important ports should be labelled and recorded.
- **Validate applications after recovery.** Restoring connectivity is only the first step.
- **Investigate automatic recovery.** I still need to determine why Asterisk was inactive and test its startup behaviour after a controlled reboot.
- **Improve monitoring.** Uptime Kuma should monitor important service endpoints, not just whether the underlying host responds.
- **Maintain useful runbooks.** Recovery instructions should be available before the next incident, not written from memory while troubleshooting.

Why This Was Worth Experiencing

A homelab is useful because it gives me a place to experiment, but it becomes much more valuable when I treat it like real infrastructure.

This incident reinforced that disaster recovery is not just about having backups. It is also about understanding dependencies, restoring services in the right order, and confirming that everything

actually works afterward.

The next step is to investigate the Asterisk startup issue, improve the recovery documentation, and repeat the relevant checks in a controlled test.

Project Oyster takeaway: Recovery is not complete when the network comes back. It is complete when the services people depend on are working again.