Services

- Infrastructure Services
  - Service Inventory
  - Docker and Portainer
  - DNS and Reverse Proxy Services
  - Uptime Kuma Monitoring
  - FreePBX and SIP Lab

Infrastructure Services

Infrastructure Services

Service Inventory

Overview

Project Oyster runs a collection of self-hosted services on Proxmox VE. These services provide infrastructure management, network visibility, documentation, automation, and telephony.

The majority of services run as Docker containers on the `infra01` Ubuntu Server VM. FreePBX runs in a separate virtual machine.

Service Directory

Service	Hostname / URL	Address / Port	Purpose
Proxmox VE	`pve01.oyster.lab`	`192.168.90.2:8006`	Virtual machine and host management
Portainer	`portainer.oyster.lab`	`192.168.90.4:9443` backend	Docker management
NetBox	`netbox.oyster.lab`	`192.168.90.4:8080` backend	IPAM, infrastructure inventory, and documentation
Uptime Kuma	`kuma.oyster.lab`	`192.168.90.4:3001` backend	Service availability monitoring
Homarr	`homarr.oyster.lab`	`192.168.90.4:7575` backend	Central service dashboard
IT-Tools	`ittools.oyster.lab`	`192.168.90.4:8081` backend	Network and development utilities
NetAlertX	`netalert.oyster.lab`	`192.168.90.4:20211` backend	Network device discovery and visibility
Technitium DNS	`infra01.oyster.lab`	`192.168.90.4:53`	Internal DNS and DNS forwarding
Nginx Proxy Manager	`infra01.oyster.lab`	`192.168.90.4:80/81/443`	Reverse proxy and web access management
BookStack	`docs.oyster.lab`	`192.168.90.4:6875` backend	Technical documentation and runbooks
FreePBX	`freepbx.oyster.lab`	`192.168.90.7`	Asterisk-based PBX and SIP lab

Service	Hostname / URL	Address / Port	Purpose
NetBox Automation	`netautoserver`	Verify current IP	Ansible and Python automation

Note: Backend ports are included for administration and troubleshooting. Normal browser access should use the configured DNS name and reverse proxy where available. NetAlertX's reverse-proxy configuration has not yet been confirmed working.

Service Hosting

- **Proxmox VE:** Physical virtualization host.
- **infra01:** Ubuntu Server VM hosting Docker-based services.
- **FreePBX:** Separate VM for telephony experiments.
- **netautoserver:** Dedicated automation environment for Python, Ansible, and NetBox integration.
- **ca01:** Planned internal certificate authority VM.

Operational Principles

- Prefer internal DNS names instead of memorizing IP addresses and ports.
- Use NetBox as the source of truth for documented infrastructure.
- Monitor important services with Uptime Kuma.
- Back up virtual machines and important service configuration.
- Keep credentials, API tokens, and private keys out of documentation.
- Verify actual service status before making infrastructure changes.

Maintenance

Update this page whenever a service is added, removed, moved, or changes its address or access method.

Docker and Portainer

Overview

Docker provides containerization for several Project Oyster services. Rather than installing every application directly on the Ubuntu operating system, services run in isolated containers with their own images, configuration, and persistent storage.

Portainer provides a graphical interface for managing Docker containers, images, networks, volumes, and application stacks.

Hosting Environment

- **VM:** `infra01`
- **IP address:** `192.168.90.4`
- **Portainer URL:** `https://portainer.oyster.lab`
- **Portainer backend:** `https://192.168.90.4:9443`
- **Stack directory:** `/opt/stacks/`

Each application should have its own directory under `/opt/stacks/`, containing its Compose file and any associated configuration or data directories.

Current Stacks

Stack / Application	Directory
Uptime Kuma	`/opt/stacks/uptime-kuma`
NetBox	`/opt/stacks/netbox`
Homarr	`/opt/stacks/homarr`
IT-Tools	`/opt/stacks/it-tools`
NetAlertX	`/opt/stacks/netalertx`
Technitium DNS	`/opt/stacks/technitium`

Stack / Application	Directory
Nginx Proxy Manager	`/opt/stacks/nginx-proxy-manager`
BookStack	`/opt/stacks/bookstack`

Useful Commands

List running containers:

```
sudo docker ps
```

List all containers, including stopped containers:

```
sudo docker ps -a
```

View Docker resource usage:

```
sudo docker stats
```

View Docker disk usage:

```
sudo docker system df
```

View logs for a container:

```
sudo docker logs --tail 100 <container_name>
```

Follow live logs:

```
sudo docker logs -f <container_name>
```

Check the status of a Compose stack:

```
cd /opt/stacks/<stack-directory>
sudo docker compose ps
```

View stack logs:

```
sudo docker compose logs --tail 100
```

Restart a stack after reviewing the configuration:

```
sudo docker compose restart
```

Apply Compose configuration changes:

```
sudo docker compose up -d
```

Operational Guidelines

- Review the Compose file before applying changes.
- Use persistent volumes or bind mounts for application data.
- Avoid deleting Docker volumes unless the data loss is intentional.
- Do not use `docker system prune --volumes` casually; it can remove data needed by stopped applications.
- Store passwords and tokens in protected environment files rather than committing them to Git.
- Back up application configuration and databases, not just container definitions.
- Pin image versions where predictable upgrades are important.
- After upgrades, verify that the application is healthy and that its data remains intact.

Troubleshooting

If a service is unavailable:

1. Check whether its container is running.
2. Review recent container logs.
3. Verify the service's published port.
4. Test direct access to the backend.
5. Check DNS and Nginx Proxy Manager configuration.
6. Confirm that the VM and network are reachable.

Docker containers running on `infra01` depend on that VM and the underlying Proxmox host being available.

DNS and Reverse Proxy Services

Overview

Project Oyster uses Technitium DNS Server for internal name resolution and Nginx Proxy Manager (NPM) to route browser requests to hosted applications.

These services work together to provide convenient internal URLs without requiring users to remember individual IP addresses and port numbers.

Technitium DNS Server

Purpose: Internal DNS hosting and forwarding.

- **Host VM:** `infra01`
- **Server IP:** `192.168.90.4`
- **DNS zone:** `oyster.lab`
- **DNS port:** TCP/UDP 53
- **Web console:** `http://192.168.90.4:5380`

The `oyster.lab` zone is configured as a Primary Zone. A records map service names to their corresponding internal IP addresses.

Examples:

DNS Name	Address
`pve01.oyster.lab`	`192.168.90.2`
`infra01.oyster.lab`	`192.168.90.4`
`freepbx.oyster.lab`	`192.168.90.7`
`ca01.oyster.lab`	`192.168.90.10`

Other application names currently resolve to `192.168.90.4`, where the relevant service or reverse proxy is hosted.

Client devices must use Technitium, directly or through the appropriate DHCP-provided DNS configuration, to resolve internal Oyster records.

Nginx Proxy Manager

Purpose: HTTP/HTTPS reverse proxy.

- **Host VM:** `infra01`
- **Server IP:** `192.168.90.4`
- **Administration interface:** `http://192.168.90.4:81`
- **HTTP listener:** TCP 80
- **HTTPS listener:** TCP 443

NPM receives requests for configured hostnames and forwards them to the corresponding application backend.

Example:

```
Browser
  |
  | http://docs.oyster.lab
  v
Technitium resolves docs.oyster.lab
  |
  v
192.168.90.4
  |
  v
Nginx Proxy Manager :80/:443
  |
  v
BookStack :6875
```

Troubleshooting

Internal hostname does not resolve

From Windows:

```
nslookup docs.oyster.lab
nslookup docs.oyster.lab 192.168.90.4
```

If the explicit query succeeds but the default query fails, investigate the DNS server configured on the client or supplied by DHCP.

Application opens by IP and port but not by hostname

1. Confirm the DNS record points to the correct address.
2. Confirm the NPM Proxy Host uses the correct domain.
3. Verify the forwarding scheme, address, and port.
4. Check NPM logs.
5. Test the backend directly from the host or NPM container.

Browser certificate warning

A valid DNS record and a working reverse proxy do not automatically guarantee a trusted HTTPS certificate. Project Oyster's internal certificate authority and client trust configuration are separate components of the HTTPS setup.

Security Notes

- Do not expose the Technitium or NPM administration interfaces publicly without a deliberate security design.
- Keep DNS administration and reverse-proxy administration restricted to trusted users.
- Internal DNS names are not a security boundary; firewall rules must still control network access.
- Document certificate renewal and backup procedures once the internal CA is operational.

Infrastructure Services

Uptime Kuma Monitoring

Overview

Uptime Kuma monitors the availability of Project Oyster services and provides a central location to check whether important endpoints are responding.

- **Host VM:** `infra01`
- **Backend URL:** `http://192.168.90.4:3001`
- **Internal URL:** `http://kuma.oyster.lab`
- **Stack directory:** `/opt/stacks/uptime-kuma`

Monitoring Strategy

Monitors should test the service that users actually depend on, not merely whether the physical host is online.

Recommended monitoring targets include:

Target	Suggested Monitor	What It Tests
`pve01`	HTTPS / TCP	Proxmox web interface availability
Portainer	HTTPS	Docker management interface
NetBox	HTTP(S)	Infrastructure inventory interface
BookStack	HTTP(S)	Documentation portal
FreePBX	HTTP(S) or TCP	Web interface availability
Technitium DNS	DNS	Name resolution
Nginx Proxy Manager	HTTP(S) or TCP	Reverse-proxy listener availability

Configure each monitor with the correct protocol, port, and expected response. A successful web request does not necessarily prove that all application functions are healthy.

Incident Response

When a monitor reports a failure:

1. Check whether the Proxmox host is online.
2. Confirm the affected VM is running.
3. Check the service or container status.
4. Review relevant logs.
5. Test direct backend access.
6. Check DNS and reverse-proxy routing.
7. Restore the service and confirm the monitor returns to an operational state.

For Docker services:

```
sudo docker ps
sudo docker logs --tail 100 <container_name>
```

For virtual machines:

```
qm list
```

Run `qm` commands on the Proxmox host.

Limitations

Monitoring is not the same as alerting. Configure and test notification channels if failures need to be reported while nobody is watching the dashboard.

Uptime Kuma itself runs on `infra01`; it may become unavailable during a failure affecting that VM or the Proxmox host. Monitoring hosted on the same infrastructure cannot independently report every infrastructure-wide outage.

Maintenance

Review monitors whenever service addresses, ports, URLs, or dependencies change. Periodically test notification delivery and confirm that the status page reflects actual service health.

FreePBX and SIP Lab

Overview

Project Oyster includes a dedicated FreePBX virtual machine for learning and testing IP telephony. FreePBX provides a web-based management interface for Asterisk, which handles telephony services and call processing.

The lab currently supports internal SIP calls between two Yealink T43U phones.

Virtual Machine

- **VM name:** `FreePBX`
- **IP address:** `192.168.90.7`
- **Web access:** `http://freepbx.oyster.lab` or the configured NPM hostname
- **SIP signalling:** UDP 5060, subject to the active PJSIP configuration

Confirm the actual configured ports and transport before troubleshooting or making firewall changes.

Extensions and Endpoints

Extension	Display Name	Endpoint
100	Test1	Yealink T43U
101	Test2	Yealink T43U

The two phones have been provisioned using MAC-specific configuration files hosted by the FreePBX VM.

Provisioning directory:

```
/var/www/html/provisioning/
```

The filenames are case-sensitive on Linux. For example, the configuration file for MAC address `80:5E:0C:73:60:37` is:

```
805e0c736037.cfg
```

Extension passwords and other secrets must not be stored in this document.

Basic Call Flow

```
Yealink T43U (Extension 100)
           |
           | SIP signalling
           v
        FreePBX
           ^
           | SIP signalling
           |
Yealink T43U (Extension 101)
```

SIP signalling establishes and controls the call. RTP typically carries the actual voice media. A call may therefore connect successfully while audio fails if RTP traffic is blocked or incorrectly routed.

Useful Troubleshooting Commands

Check Asterisk service status on the FreePBX VM:

```
sudo systemctl status asterisk
```

Start Asterisk if it is stopped:

```
sudo systemctl start asterisk
```

Review recent service logs:

```
sudo journalctl -u asterisk -n 100 --no-pager
```

Check whether UDP 5060 is listening:

```
sudo ss -lunp | grep 5060
```

These commands should be run inside the FreePBX VM, not on `infra01` or the Proxmox host.

Known Recovery Observation

During a previous network interruption, the FreePBX VM became reachable again but Asterisk remained inactive. Starting Asterisk manually restored telephony service.

This indicates a recovery behaviour that needs further investigation. Verify why Asterisk did not start automatically after the interruption or restart, and test the service's startup behaviour after making any changes.

Future Work

- Obtain authorized test DID and SIP trunk details from the provider.
- Configure and test inbound and outbound calls.
- Document the SIP transport, codecs, RTP port range, and number format once confirmed.
- Verify Asterisk starts automatically after a VM reboot.
- Back up the FreePBX VM before major telephony changes.
- Keep SIP credentials and extension secrets out of documentation and screenshots.

Security Notes

Do not expose SIP or RTP ports to the public internet without an explicit design and appropriate restrictions. Use test numbers and provider-approved credentials only. Keep the lab isolated from production telephony unless a connection has been deliberately authorized.