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.

Revision #1
Created 2026-10-09 14:07:39 UTC by Admin
Updated 2026-10-09 14:07:58 UTC by Admin