ENTRO
NILE BASIN INITIATIVE
INITIATIVE DU BASSIN DU NIL

Hands-On Training

on

EN-WRMT

Platform Administrator Training for Web Platform

Trainers:

Miss Beimnet Girma,

Mr. Kedir Hajisheko and

Dr. Binyam Tesfaw Hailu,

HydroNova aquacon ENGINEERING PLC

Training Modules

Module 1: Introduction & Overview

- System architecture overview
- User roles and permissions (Public viewers, Entro users, Experts)

Module 2: System Deployment & Hosting

- Environment setup (Production vs Development)
- Deployment pipeline
- Server configuration

Module 3: System Manangment

- User Management
- Backup and Disaster Recovery
- Authentication & Authorization
- Backup strategy

Module 4: System Maintenance

- Software Updates
- Integration with MapStore
- MapStore instance management (running mapstore)
- Syncing user roles and permissions

Module 5: Security & Monitoring

- HTTPS setup and cert renewal
- SOP Documentation

Module 1: Introduction and Overview

This module provides a foundational understanding of the WebGIS platform, focusing on the system architecture and the roles and responsibilities assigned to different user groups. By the end of this module, trainees will understand how various system components interact and the level of access granted to each user category.

1. System Architecture Overview

The WebGIS platform is designed as a modular and scalable system that integrates geospatial visualization, spatial databases, and simulation tools. Its architecture ensures efficient rendering of maps, secure data access, and maintainable backend services. The main components are as follows:

Frontend Interface

The platform's front-end is built using Next.js, a modern React-based framework optimized for performance and SEO. Next.js provides a seamless and interactive user experience, enabling navigation across maps, dashboards, data layers, and analytical tools.

Map Visualization Layer

The map visualization is managed by MapStore, an open-source web mapping framework tailored for geospatial data publishing. MapStore is embedded directly into the Next.js frontend, ensuring users can interact with map layers, time series data, and tools without needing to navigate separate interfaces.

Backend Services

The backend infrastructure supports data storage, rendering, and user session management through the following components:

- **GeoServer** and **MapStore** are deployed on **Apache Tomcat**, which handles Java-based application hosting.
- **PostgreSQL** enhanced with **PostGIS** serves as the spatial database, enabling advanced geospatial queries and storage.
- **PM2**, a Node.js process manager, ensures high availability and automatic restarts for services like Next.js and related Node processes.
- **Nginx** functions as a **reverse proxy**, routing requests to the appropriate service and enforcing HTTPS for secure communication.

This modular setup facilitates streamlined maintenance, robust performance, and secure access across different user groups.

2. User Roles and Permissions

To maintain a secure and organized system, the platform enforces role-based access control (RBAC). Each user group has distinct permissions based on their responsibilities and technical expertise:

a. Public Viewers

- Do not require login credentials.
- Can view publicly available maps, dashboards, and summary data.
- Cannot download data or access internal tools or datasets.
- Ideal for stakeholders or the general public interested in high-level information.

b. Entro Users

- Must authenticate using registered credentials.
- Access extended datasets, interactive dashboards, and downloadable resources.
- Cannot modify spatial data or publish new content.
- Typically include internal staff or partners needing deeper access for analysis and reporting.

c. Entro Experts

- Authenticated users with elevated permissions.
- Full access to **MapStore** and **GeoServer** functionalities, including:
    - Creating and updating spatial models.
    - Publishing analytical tools.
    - Managing and configuring map layers and styles.
- Usually technical staff responsible for data maintenance and system enhancement.

d. Administrators

- Possess unrestricted access to all system components.
- Manage user accounts, roles, and permissions via a custom dashboard integrated with **Clerk**, the authentication provider.
- Oversee platform configuration, maintenance scheduling, and data governance policies.

Module 2: System Deployment

This module provides a detailed walkthrough of how the WebGIS platform is deployed, accessed, and secured. It covers environment setup, server access, the deployment pipeline, configuration of services, and system hardening practices. Understanding this module is essential for DevOps personnel and technical administrators who manage infrastructure and deployment.

1. Environment Setup

The WebGIS platform is designed to be OS-independent, but the production environment is optimized for **Linux-based systems**, particularly **Ubuntu**. The deployment supports flexibility and scalability by accommodating different hosting environments:

Hosting Environments

- **Hydronova's Server**: Initial deployment for development and demonstration purposes.
- **Entro's Local Server**: Intermediate deployment for internal testing and validation.
- **Entro's Cloud Server** (Planned): Final production-grade deployment that ensures high availability and remote scalability.

This modular deployment approach allows for parallel development, testing, and production lifecycles.

2. Accessing the Server

Remote server access is enabled via **SSH (Secure Shell)**, a standard protocol for encrypted command-line communication.

SSH Access Command

To connect:

```
ssh username@[server_domain_or_IP]
```

3. Deployment Pipeline

The deployment process ensures consistency and reliability when updating the platform. It involves retrieving code from version control, building the application, and restarting services.

Source Code Repository

All code is maintained on **GitHub**. Migration to **Entro's GitHub organization** is encouraged for ownership consolidation.

Deployment Steps

1. **Navigate to the deployment directory**:

```
> cd /var/www/entro
```

2. Clone the repository:

```
> git clone https://github.com/[repositoryname]/entro.git .
```

3. Install dependencies and build the Next.js frontend:

```
> npm install
> npm run build
```

```
> ls
```

4. Server Configuration

To serve multiple components through a unified entry point, the platform uses **Nginx** as a reverse proxy. This setup enables service routing, access control, and TLS encryption.

Nginx Reverse Proxy Mapping

- Serve the **Next.js** frontend at: `/`
- Serve **MapStore** (map visualization) at: `/mapstore`
- Serve **GeoServer** (spatial data service) at: `/geoserver`
- Serve **Tomcat Admin Tools** at: `/tomcat`

This configuration keeps URLs organized and allows for centralized routing and security management.

5. Security & SSL Certificates

Security is critical in a multi-user system handling sensitive geospatial data. Several layers of protection are implemented:

User & Platform Security

- **HTTPS Encryption**: All traffic is encrypted using **TLS**, enforced via Nginx.
- **Access Control**: Managed through:
    - Nginx-based **Access Control Lists (ACLs)**
    - Role-based permissions via **Clerk** (user authentication and management).
- **Database Protection**:
    - Use strong, randomly generated passwords.
    - Secure PostgreSQL access with host-based firewall rules (e.g., UFW).

TLS/SSL Certificate Management

- **Let's Encrypt** is used to issue free, automated certificates.
- **Certbot** handles certificate installation and renewal.
    - Automate renewals using a cron job:

```
certbot renew
```

6. Port Management & Firewall Rules

To minimize attack surfaces, the following best practices are enforced:

Publicly Exposed Ports

- **Port 80 (HTTP)**: Used temporarily to issue/renew SSL certificates.
- **Port 443 (HTTPS)**: Default for encrypted web traffic.

Restricted Ports

Services like **GeoServer** and **Tomcat** expose ports internally and should **not** be publicly accessible. Use **UFW** or similar firewall utilities to block direct external access.
Check Exposed Ports

```
> sudo ufw status
```

Allow or Deny Port Access

```
> sudo ufw allow [port][/protocol]
```

```
> sudo ufw deny [port][/protocol]
```

Reverse Proxy Enforcement

Route all external access through **Nginx**, which acts as a security gatekeeper and ensures that no internal ports are exposed unnecessarily.

Module 3: System Management

1. Authentication & Authorization

Secure and flexible access control is vital to maintaining data integrity and system trust. The platform employs **Clerk**, a modern identity management service, to handle all aspects of user authentication and authorization.

Authentication Mechanisms

Clerk supports email/password login as well as third-party Sign-In for convenience and security.

- Clerk automatically handles:
    - Password encryption and recovery
    - Session management with secure tokens
    - Multi-factor authentication (MFA) if enabled

Authorization and Role Assignment

Admin users manage the full user lifecycle via the **Clerk Admin Dashboard**, which provides a web interface for:

- **Adding or approving** new users
- **Assigning roles** based on system privileges:
    - *Viewer*: Public or limited access
    - *Entro User*: Authenticated access with downloads
    - *Expert*: Edit and publish tools within MapStore and GeoServer
- **Suspending or deleting** inactive or compromised accounts

2. Security and Privacy Considerations

Handling user identity and data responsibly is essential to maintaining system compliance and ethical standards.

OAuth and Identity Token Management

Clerk stores **OAuth tokens** securely using encrypted, short-lived sessions. Identity data is isolated and not accessible to unauthorized services.

Admin Access to Sensitive Data

- Administrative access to user activity logs and identity metadata should be:
    - Granted only when necessary
    - Logged for accountability
    - Reviewed regularly for compliance

Data Protection Guidelines

- Avoid storing sensitive user information (e.g., passwords, personal emails) outside Clerk's infrastructure.
- Ensure all communication occurs over **HTTPS**, enforced via Nginx.

3. Backup Strategy

Backups are critical for recovering from data loss, corruption, or unexpected server failure. This section outlines a dual backup approach covering both the database and the file system.

a. PostgreSQL Database Backups

- Use `pg_dump` to create daily backups of the spatial database.

```
pg_dump -U postgres entro_db > /backups/db_backup_$(date +\%F).sql
```

- Store backup files in:
    - A **cloud storage bucket** (AWS S3, Google Cloud Storage, etc.)
    - A **detached volume** or **external drive** mounted on the server

b. GeoStore and MapStore File Backups

- *[covered in Module 4 training]*

4. Static Asset Management

The system includes user-uploaded and administrator-managed assets such as:

- Simulation models
- Toolkits
- Training videos and PDFs

5. Restore Procedures

In the event of a system failure or data loss, follow these steps for recovery:

Database Restore

Use `pg_restore` (for custom-format dumps) or `psql` (for SQL dumps) to reload the PostgreSQL database:

```
psql -U postgres entro_db < /backups/db_backup_YYYY-MM-DD.sql
```

File System Restore

Extract compressed MapStore or asset backups to their original directories:

```
tar -xzf /backups/mapstore_YYYY-MM-DD.tar.gz -C /var/www/mapstore
```

Restart services as needed using PM2 or systemd:

```
pm2 restart all
```

Module 4: System Maintenance

1. Update Procedures

- **MapStore**:
  - *[covered in Module 4 training]*
- **GeoServer**:
  - *[covered in Module 4 training]*
- **Tomcat**:
  - *[covered in Module 4 training]*

Next.js (Frontend) Updates

Update Node modules and rebuild the frontend:

```
npm install
```

```
npm run build
```

```
pm2 restart [frontend_id]
```

2. Monitoring System Health and Logs

Effective system monitoring helps detect and resolve issues early.

Next.js and API Services

Use **PM2** to manage and monitor Next.js services:

```
pm2 status
```

You can check a specific service's log by doing

```
pm2 logs [service id]
```

Tomcat Services (MapStore, GeoServer)

In addition to web-based access (Tomcat Manager) to check service status and application health you can review Tomcat logs for errors or warnings:

```
tail -f /opt/tomcat9/logs/catalina.out
```

3. Integration and Syncing User Roles with MapStore

MapStore has its own internal user and permission system-- separate from the Clerk-managed identity system. Coordinating both is essential to preserve consistent access control.

A. Understanding Role Decoupling

- Clerk manages global roles for the web platform.
- MapStore manages user roles and permissions for map editing and publishing.
- Even though maps are embedded in the frontend, MapStore is a standalone application and must be updated independently.

B. Syncing Expert User Roles

To ensure Experts can use MapStore's editing capabilities:

- Admins must manually **create corresponding user accounts in MapStore** using the same email or username as in Clerk.
- Assign these users to the **"ADMIN" or "USER" roles** depending on their required level of access.

[Optionally, configure LDAP or SSO integration if unified login is needed (requires advanced setup)]

C. Managing Permissions within MapStore

- Use the MapStore Admin UI to:
    - Grant access to specific maps or dashboards
    - Assign resource permissions: view, edit, publish
    - Define **workspaces** for different user groups (e.g., Entro Experts)

Module 5: Security & Monitoring

This module outlines the essential security configurations, certificate management processes, and the development of Standard Operating Procedures (SOPs) for long-term maintainability and knowledge transfer. It ensures the platform remains secure, compliant, and manageable by both current and future administrators.

Most of the concepts here have been discussed in a more detailed manner earlier on in this training manual, but are reiterated in this section for completeness.

1. Platform Security Overview

Security is a foundational element of the WebGIS platform. All components—frontend, backend, and database—must be protected from unauthorized access and misuse.

A. Network and Server Hardening

Firewall Configuration:
Use `ufw` or `iptables` to allow only essential ports:

- Port 80 (HTTP)
- Port 443 (HTTPS)

Block access to internal ports used by Tomcat, GeoServer, and PostgreSQL.

SSH Access:

- Limit access to known IP addresses where possible.
- Disable root login and use key-based authentication.
- Regularly audit active SSH users.

B. Authentication Policies

Enforce strong password policies in Clerk and system-level users.
Require two-factor authentication (2FA) for admin accounts where supported.
Periodically review user roles and permissions in Clerk and MapStore.

C. Data Security

Ensure that PostgreSQL uses encrypted connections where possible (`sslmode=require`).
All file uploads should be scanned or validated before being made public.

2. HTTPS Configuration and TLS Certificate Renewal

Secure communication is enforced through HTTPS using TLS certificates issued by Let's Encrypt.

A. Initial Setup

Use Nginx as a reverse proxy to enforce HTTPS across all services.
Certificates are issued using **Certbot**, which is compatible with Let's Encrypt.

You can install certbot using:

```
sudo apt install certbot python3-certbot-nginx
```

You can setup certbot for your domain by doing:

```
sudo certbot --nginx -d yourdomain.com
```

B. Auto-Renewal Configuration

Let's Encrypt certificates expire every 90 days; automated renewal must be enabled via:

```
certbot renew --quiet
```

Confirm renewals with:

```
sudo certbot certificates
```

C. Post-Renewal Hook (Optional)

To reload Nginx automatically after renewal:

```
certbot renew --post-hook "systemctl reload nginx"
```

3. Monitoring and Logging

To ensure service reliability, basic monitoring and logging should be in place.

A. Log Management

Next.js logs via PM2:

```
pm2 logs frontend
```

Tomcat logs:

```
tail -f /opt/tomcat/logs/catalina.out
```

Nginx logs:

```
tail -f /var/log/nginx/access.log
```

```
tail -f /var/log/nginx/error.log
```

4. Standard Operating Procedures (SOPs)

This manual is designed to serve as a foundational set of Standard Operating Procedures (SOPs) for the EN-WRMT WebGIS platform. Use the checklist below to critically assess whether the documentation meets the purpose and expectations of an effective SOP.

Use this section to reflect on whether each SOP area is fully supported by the manual:

SOP Area	Reflection Questions
System Deployment SOP	Can someone fully set up the system from scratch using the deployment steps? Are repo cloning, environment setup, and service initialization clearly explained?
User Management SOP	Are procedures for creating, approving, and managing users in both Clerk and MapStore well documented?
Backup & Restore SOP	Are backup schedules, storage locations, and restore procedures clearly written and testable?
Update SOP	Are update processes for all major components (Next.js, MapStore, GeoServer, Tomcat) described with commands and risks noted?
Certificate Renewal SOP	Does the manual cover both manual and automated HTTPS renewal (e.g., with Certbot and cron)?
Disaster Recovery SOP	Are emergency response steps for outages, breaches, or data loss available and clear enough to act on quickly?