|  |  |
| --- | --- |
|  |  |

Portal Developer Overview

# Abstract

A simple overview detailing the various parts of the web portal for AIA available at analytics.ericsson.se

# Portal Overview

## What are the Services in the Portal for?

The main website or portal is made up of several services along with the core web elements. The full list of components used in the portal are:

1. Portal, developed using angular, deployed to filesystem on the machine
2. Nginx, deployed as a container, used to host the portal
3. Haproxy used to load balance the Portal. Hosted native to the machine
4. WordPress, deployed as a container, used for LDAP and as a user/community content provided for the webpage.
5. MariaDB, deployed as a container, used as the storage for WordPress
6. Template Manager, deployed as a container. Used to provide all functionality with regards to the management of application templates on the portal.
7. Gogs, deployed as a VM in ITTE open stack, used to store application templates
8. Postgres used as the backend for Gogs, deployed as a container
9. Application Manager, deployed as a container, used to provide all functionality with regards to the management of applications on the portal.
10. Metastore used to provide all storage functions for application manager and template manager metadata, deployed as a container
11. Mongo DB used as the storage provider for the Metastore, deployed as a container.

## Where are the Portals?

Currently the portal is deployed on two servers Live and Staging, these are

VMs hosted in the HUB. If you need to interact with these servers you wlll need to Obtain Access Rights In your browser, open <https://idm.internal.ericsson.com/itim/ssui/>. In the **My Access**tab, search for and select "Rnd\_access\_seli\_Nam\_Analytics". Then select **Obtain this IT Access**. A notification will be sent out to be approved. Once the access has been approved

Live is hosted on: eselivm3v258l.lmera.ericsson.se

Staging is hosted on: eselivm3v260l.lmera.ericsson.se

Internal DNS maps analytics.ericsson.se to live (eselivm3v258l)

The following pages show an overview of the service layouts on live and staging note that the main difference is the use of HA proxy on Live.

The following figure shows the full component list for live today.

Application Manager

MongoDB

Template Manager

Metastore

WordPress

Gogs

ITTE

VM

Hosting

GOGs

NGINX

Postgres

MariaDB

Portal Files

VM Disk

Network Mapped Storage

HA Proxy

The following figure shows the full component list for staging today.

ITTE

VM

Hosting

GOGs

Application Manager

MongoDB

Template Manager

Metastore

WordPress

Gogs

NGINX

Postgres

MariaDB

Network Mapped Storage

VM Disk

Portal Files

## How are the services related?

Current service relations

Portal NGINX

WordPress

Portal App builder

Portal Community

Portal Template builder

Template Manager

Metastore

Application Manager

MariaDB

MongoDB

Gogs

Postgres

On Live Haproxy will sit in front of the NGINX service

User request flow:

1. Portal community

Request -> server:80 -> haproxy forwards to NGINX:18080 -> wordpress: 23307 -> maria DB on 23306

1. Portal Application Builder

Request -> server:80 -> haproxy forwards to NGINX:18080 -> applicationManager:6868 -> metastore:8080/8009->mongo:27017

3. Portal template builder

Request -> server:80 -> haproxy forwards to NGINX:18080 -> templateManager:6767 |-> metastore:8009/8080 -> mongo:27017

|-> gogs:443/80 -> postgres:5432

Note that in the event Nginx on live goes offline, Haproxy will direct requests to staging:80 which is NGINX:80 all other ports and services remain the same.

## How to deploy

It is best to perform a data restore on new servers before bringing the services online.

Live extra step: configure HAproxy.

Copy the haproxy.cfg file into:

/etc/haproxy/haproxy.cfg

Start/Restart

systemctl restart haproxy

systemctl start haproxy

Main steps:

1. check docker storage mapping – (do not change if already set)

in /etc/sysconfig/docker confirm the -g flag is set

OPTIONS='--selinux-enabled -g /local/lib/docker'

1. Confirm the UI is pointing at the correct endpoints

vi /local/data/portal/app/api.js

1. Bring up the services using docker compose

before runing compose the host ip must must be set

export HOSTIP=<serverIP>

docker-compose up -d

1. Update Mongo with the expected user

exec into mongo and create the admin user

$ docker exec -it <container\_id> mongo

>use aia

>db.createUser(

{

user: "admin",

pwd: "admin",

roles: [

{ role: "read", db: "aia" },

{ role: "readWrite", db: "aia" }

]

})

## Data Restore

check for latest backup here <http://10.44.149.3:8080/swift/v1/backup/>

update and run the below command to get the backup:

wget http://10.44.149.3:8080/swift/v1/backup/<backupTarName>.tar.gz

extract the backup:

tar -zxf <backupTarName>.tar.gz -C /local

check the api file is configured for the current server.

/local/data/portal/app/api.js

For mongo, an import needs to be run on the service itself. The mongo container has a mount to allow easy data transfer to and from the container.

Server location /local/data/mongo is mapped to container location /tmp/mongobackup

The follow commands cover all current collections stored in mongo.

docker exec mongo mongoimport -d aia -c aia-application-catalog /tmp/mongobackup/aia-application-catalog.json

docker exec mongo mongoimport -d aia -c aia-template-catalog /tmp/mongobackup/aia-template-catalog.json

docker exec mongo mongoimport -d aia -c aia-templates /tmp/mongobackup/aia-templates.json

docker exec mongo mongoimport -d aia -c aiaServiceCatalog /tmp/mongobackup/aiaServiceCatalog.json

docker exec mongo mongoimport -d aia -c applications-dependencies /tmp/mongobackup/applications-dependencies.json

docker exec mongo mongoimport -d aia -c applications /tmp/mongobackup/applications.json