FORESIGHT

ADVANCED CYBER-SECURITY SIMULATION PLATFORM FOR PREPAREDNESS TRAINING IN AVIATION, NAVAL AND POWER-GRID ENVIRONMENTS

Grant Agreement: 833673

D9.4

Collaboration modules (I)

This project has received funding from the European Union's Horizon 2020 Research and Innovation Programme under Grant Agreement No 833673.

Document Information

Deliverable number:	**D9.4**
Deliverable title:	Collaboration modules (I)
Deliverable version:	1.0
Work Package number:	WP9
Work Package title:	FORESIGHT toolkit development
Due Date of delivery:	M18
Actual date of delivery:	M18
Dissemination level:	Public (P)
Type	Report
Editor(s):	Alkiviadis Giannakoulias (ED)
Contributor(s):	Alkiviadis Giannakoulias (ED)
Reviewer(s):	Damir Haskovic (M&S) Yasen Todorov (CEZ)
Project name:	Advanced cyber-security simulation platform for preparedness training in Aviation, Naval and Power-grid environments
Project Acronym	FORESIGHT
Project starting date:	1/10/2019
Project duration:	42 months
Rights:	FORESIGHT Consortium

Document history

Version	Date	Beneficiary	Description
0.1	05.02.2021	ED	ToC
0.2	26.02.2021	ED	First version of deliverable with all sections populated
0.3	10.03.2021	ED	Received and addressed comments by internal reviewers M&S
0.4	26.03.2021	ED	Received and addressed comments by internal reviewers CEZ
0.5	26.03.2021	ED	Final version submitted to KEMEA for approval and submission to the European Commission
1.0	31.03.2021	KEMEA	Final refinements and submission to the EC

Acknowledgement: This project has received funding from the European Union's Horizon 2020 Research and Innovation Programme under Grant Agreement No 833673.

Executive Summary

One of the main goals of WP9 is to enable the efficient collaboration between cyber-team members during training exercises to assist cyber-security professionals to collaboratively improve their skills and abilities.

This document presents the first version of the cyber-team collaborative (CTC) tools that offer users the ability to interact with each other through:

a) a communication hub in order to exchange knowledge, ideas and have better and more fruitful learning experience;
b) a collaboration hub, allowing FORESIGHT platform users to:
    i. interact with each other through mailing and distribution lists;
    ii. share content and collaborate with other FORESIGHT platform members by accessing, creating, uploading and editing content/documents directly within the system;
    iii. communicate and collaborate asynchronously with other FORESIGHT platform members via forums and discussions;
    iv. search for a particular user;
    v. retrieve the responsibilities for a certain incident, e.g., which teams, superiors, or authorities to contact;
    vi. organise and maintain information from Cyber Ranges (CRs) related to the CRs capabilities, the resources used/available, the available scenarios/systems including their configuration, deployment and availability, the scenarios narrative and objective, the training difficulty, the rating specifications and methods to check completion

All functional requirements have been implemented in form of dedicated software artefacts (components) including: (i) the Identity Service, (ii) the communication hub, and (iii) the collaboration hub. The architecture and functional design of the components are presented in section 3 while section 4 presents details of the prototype implementation, deployment and installation of these components. The prototype description includes installation requirements and manual, configuration and the references to artefact repositories of the code.

The document also collects the results of the unit testing part of the assessment process (section 5), which fulfil the requirements defined in WP2.

Contents

Figures

Tables

Acronyms & Abbreviations

Term	Description
AAA	Authentication, Authorisation, and Accounting
ACS	Alfresco Content Services
API	Application Programming Interface
CA	Certificate Authority
CR	Cyber Range
CTC	Cyber-Team Collaborative tools
HA	High Availability
HTTP	Hypertext Transfer Protocol
HTTPS	Hypertext Transfer Protocol Secure
IdP	Identity Provider
JSON	JavaScript Object Notation
OS	Operating System
RAM	Random-Access Memory
RDBMS	Relational Database Management System
REST	Representational State Transfer
SSL	Secure Sockets Layer
SSO	Single Sign On
UI	User Interface
VM	Virtual Machine
YAML	YAML Ain't Markup Language

1 Introduction

1.1 Overview

One of the main goals of WP9 is to enable the efficient collaboration between cyber-team members during training exercises to assist cyber-security professionals to collaboratively improve their skills and abilities.

Considering that communication and collaboration is very important in distance learning and training the first version of the cyber-team collaborative (CTC) tools will offer users the ability to interact with each other through: a) a communication hub and b) a collaboration hub.

The former is based on RocketChat, an open source communication hub that enables banks, NGOs, startups, and governmental organizations to have their own chat tool, customize its look and feel, choose their users, and securely manage data.[6]

The latter is based on Alfresco Community Edition[1], an open source Enterprise Content Management software that handles any type of content, allowing users to easily share and collaborate on content [7].

The document is accompanied by the actual implementation of the following software components: (i) Reverse Proxy, (ii) Identity Service, (iii) communication hub, and (iv) collaboration hub, in form of software prototypes that are available as explained in Section 4.

1.2 Relation to other tasks and deliverables

This deliverable is related to the following other FORESIGHT tasks and deliverables:

Receives inputs from:

Table 1. Inputs from Deliverables

Deliverable	Description
D2.2	End-User Security Requirements Report (I)
D2.3	FORESIGHT Cyber-Range Requirement Report
D2.4	FORESIGHT Architecture report
D10.2	Federated User Interface

1.3 Structure of the deliverable

The deliverable is structured in six chapters:

- Chapter 1 is the introduction of the document.
- Chapter 2 analyses the system requirements related to the efficient collaboration between cyber-team members.

[1] https://www.alfresco.com/products/community/download

- Chapter 3 presents the architecture and related components, by defining and describing the decomposition of the FORESIGHT Cyber-team collaborative tools into components (ARCADE Component viewpoint [16]).
- Chapter 4 presents the details of the prototype implementation including installation prerequisites, installation guide and information about the artefact repositories of the code.
- Chapter 5 presents the unit tests done to demonstrate requirements compliance.
- Chapter 6 draws conclusions.

2 Requirements Analysis of Cyber-team Collaborative Tools

From **D2.3 FORESIGHT Cyber-Range Requirement Report,** the requirements of the CTC tools have been extracted, both functional and non-functional requirements. Table 2 shows these requirements and how they are addressed/realised by the prototype implementation of the FORESIGHT CTC which is the outcome of the current deliverable.

Table 2. Coverage of FORESIGHT requirements on Cyber-team Collaborative Tools (CTC)

Req ID	Description	How addressed
Req-082	**Access Interface (Federation)** – Make available to the federated access interface the resources used by the original access interfaces, such as scenario availability, availability of activities, etc., at one singular location – the location being the Federated access interface	FORESIGHT-CTC collaboration hub collects, organises and maintains information from Cyber Ranges (CRs) related to their capabilities, the resources used/available, the available scenarios/systems including their configuration, deployment and availability, the scenarios narrative and objective, the training difficulty, the rating specifications and methods to check completion. FORESIGHT platform users can access this information through a dedicated visualisation (see D10.2 section 2.7).
Req-106	**Collaboration module** – The Collaboration modules must enable collaboration between cyber-team members (chat and instant messaging, content management and document services, search and full-text indexing, mailing and distribution lists etc.)	FORESIGHT-CTC communication hub enables FORESIGHT platform users to communicate and collaborate with each other via chat messages. FORESIGHT-CTC collaboration hub: • maintains mailing and distribution lists and enables FORESIGHT platform users to be informed on the available ones, including their description and the list of FORESIGHT platform users ("subscribers") receiving mail, as well as request for new mailing lists regarding topics of interest; • obtains list of users and allows FORESIGHT platform users to search for a particular user; • maintains an incident taxonomy list, based on ENISA's "Reference Incident Classification Taxonomy" and allows FORESIGHT platform users to retrieve the responsibilities for a certain incident; • supports full text search properties, by integrating with the Solr 6 Enterprise search platform for searching within the content repository; • maintains content management and document services enabling FORESIGHT platform users to share content and collaborate with other users

		by accessing, creating, uploading and editing content/documents directly within the system;
Req-171	**Online Collaboration** – The platform should enable users to interact with each other through forums, chat boxes etc in order exchange knowledge, ideas and have a better and more fruitful learning experience.	FORESIGHT-CTC communication hub enables FORESIGHT platform users to communicate and collaborate with each other via chat messages. FORESIGHT-CTC collaboration hub enables FORESIGHT platform users to communicate and collaborate asynchronously with each other via forums and discussions.
Req-203	**Communication lists** – It should be possible to retrieve the responsibilities for a certain incident, e.g. which teams, superiors, or authorities to contact.	FORESIGHT-CTC collaboration hub maintains an incident taxonomy list, based on the "Reference Incident Classification Taxonomy" of ENISA.
Req-204	**Email communication** – The system should support email communication.	As custom email addresses, used only within training platforms do not enhance realism, it was decided to delay delivery of this requirement. Instead, the FORESIGHT-CTC collaboration hub maintains mailing and distribution lists and allows FORESIGHT platform users to obtain their description and the list of FORESIGHT platform users ("subscribers") receiving mail.

As already mentioned, the communication hub is based on RocketChat[2]. Although there are many other open source chat and messaging platforms, such as Zulip[3], Let's Chat[4], Jitsi[5], Mattermost[6], element[7], after consultation with the end users, and considering their familiarity with the tool it was decided to use RocketChat in the first version of the communication hub. RocketChat is a free open source web chat application that allows team collaboration and communication, while ensuring that all conversations are secure. It can potentially replace slack [14].

Regarding the collaboration hub, it will be based on Alfresco Community Edition[8]. Although there are many other open source Enterprise Content Management systems, such as Concrete[9], Drupal[10], ExpressionEngine[11], Joomla[12], WordPress[13], Confluence[14], after consultation with the end users, and

[2] https://rocket.chat/
[3] https://zulip.com/
[4] https://sdelements.github.io/lets-chat/
[5] https://jitsi.org/
[6] https://mattermost.com/
[7] https://element.io/
[8] https://www.alfresco.com/ecm-software/alfresco-community-editions
[9] https://www.concrete5.org/
[10] https://www.drupal.org/
[11] https://expressionengine.com/
[12] https://www.joomla.org/
[13] https://wordpress.com/
[14] https://www.atlassian.com/software/confluence

considering their familiarity with the tool it was decided to use Alfresco Community Edition in the first version of the collaboration hub.

3 Cyber-team Collaborative Tools Architecture

3.1 Architecture Overview

Figure 1 shows the overall architecture of the cyber-team collaborative tools. As shown, FORESIGHT platform users access the cyber-team collaboration tools from their own computers/devices/clients. Their requests are received by the **Nginx HTTPS Reverse Proxy** which sits behind the firewall, in a private network, and acts as an intermediary proxy service taking the client requests, passing it on to the appropriate backend servers hosting the communication and collaboration services, and subsequently delivering the server's response back to the client. It provides an additional level of abstraction and control, ensuring smooth flow of network traffic between clients and servers. It also performs SSL encryption thus take load off of the web servers, thereby boosting their performance.

The **FORESIGHT Identity Service** (authentication and authentication server) is implemented on top of **JBoss Keycloak**, which is both an Identity Provider (IdP) and a token issuer for OAuth 2 tokens. It centralizes the authentication function and supports Single Sign On (SSO). The FORESIGHT Identity Service is responsible for authenticating users so that the cyber-team collaborative tools/services do not have to deal with login forms, authenticating users and storing users. Once a user is logged into the FORESIGHT Identity Service, they don't have to login again to access any application. This also applies to logout, which means that once a user is logged out of FORESIGHT Identity Service they are also automatically logged out of all other applications. An external relational database (RDBMS) namely MySQL is chosen, to persist some metadata about realms, clients, users, and so on.

The **FORESIGHT Communication Hub** is based on RocketChat and uses a **MongoDB** replica set to improve performance via Meteor Oplog tailing, while also providing High Availability (HA).

The **FORESIGHT Collaboration Hub** is based on Alfresco Community Edition and uses a relational database (PostgreSQL) for storing content metadata and a file system to store the actual content.

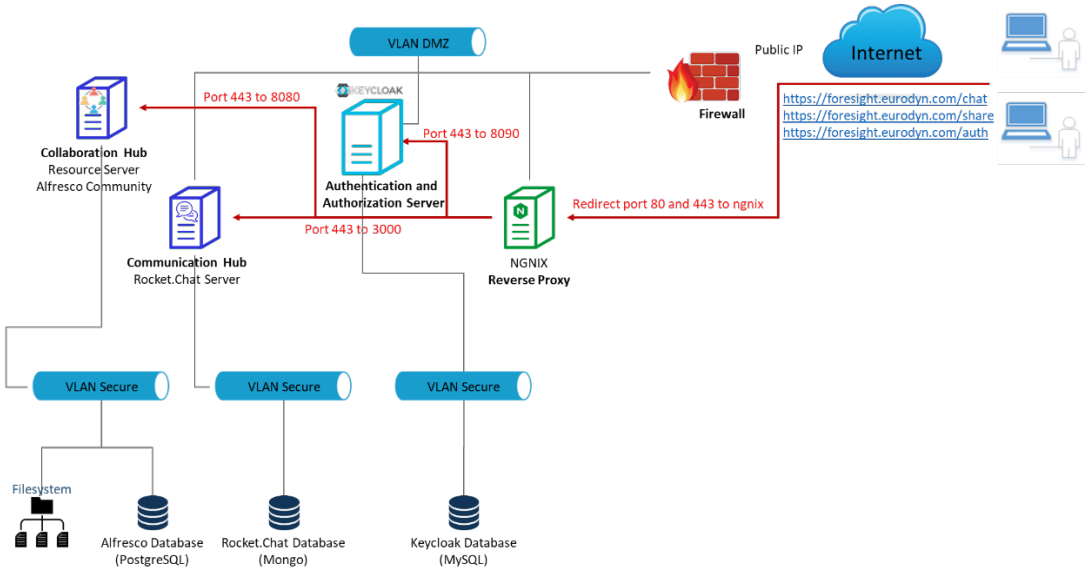


Figure 1. Cyber-Team Collaborative Tools High Level Architecture

3.2 Component Model

3.2.1 Communication Hub

The **FORESIGHT Communication Hub** does not have any internal components to show.

3.2.2 Collaboration Hub

As shown in Figure 2, the three main components of the **FORESIGHT Collaboration Hub** are: the **Platform**, the User Interface (**UI**), and the **Search engine**. The **Platform**[15] is located at the core and provides the repository where content is stored including all associated content services**.** The repository is supported by a server/storage engine that persists content, metadata, associations/relationships, and full text indexes. It provides a store for content, and a wide range of services that can be used by content applications to manipulate the content [8].

The repository is a logical entity that consists of three important parts [9]:

1. The physical content files that are uploaded, which are stored in the file system.
2. The index files created when indexing the uploaded file so it is searchable, which are managed by the Apache Solr server.
3. The metadata/properties for the file, which are stored in the relational database (PostgreSQL).

The **Alfresco Share** provides a web client interface (the UI) for the repository and allows FORESIGHT collaboration resource users to manage their collaboration areas, documents, and so on. Share provides content management capabilities with simple user interfaces, tools to search and browse the repository, content such as thumbnails and associated metadata, previews, and a set of collaboration tools such as wikis and discussions. The **search** functionality is implemented on top of Apache **Solr 6** and provides the indexing of all content, which enables powerful search functionality [8].

The Platform and Share components run in the same **Apache Tomcat** web application server. The Search component runs in its own **Jetty** web application server. The Platform is also integrated with an identity service built on **Keycloak** in order to sync users and groups.

[15] For more information about the internals of the Platform, see here: https://docs.alfresco.com/community/concepts/dev-repository-concepts.html

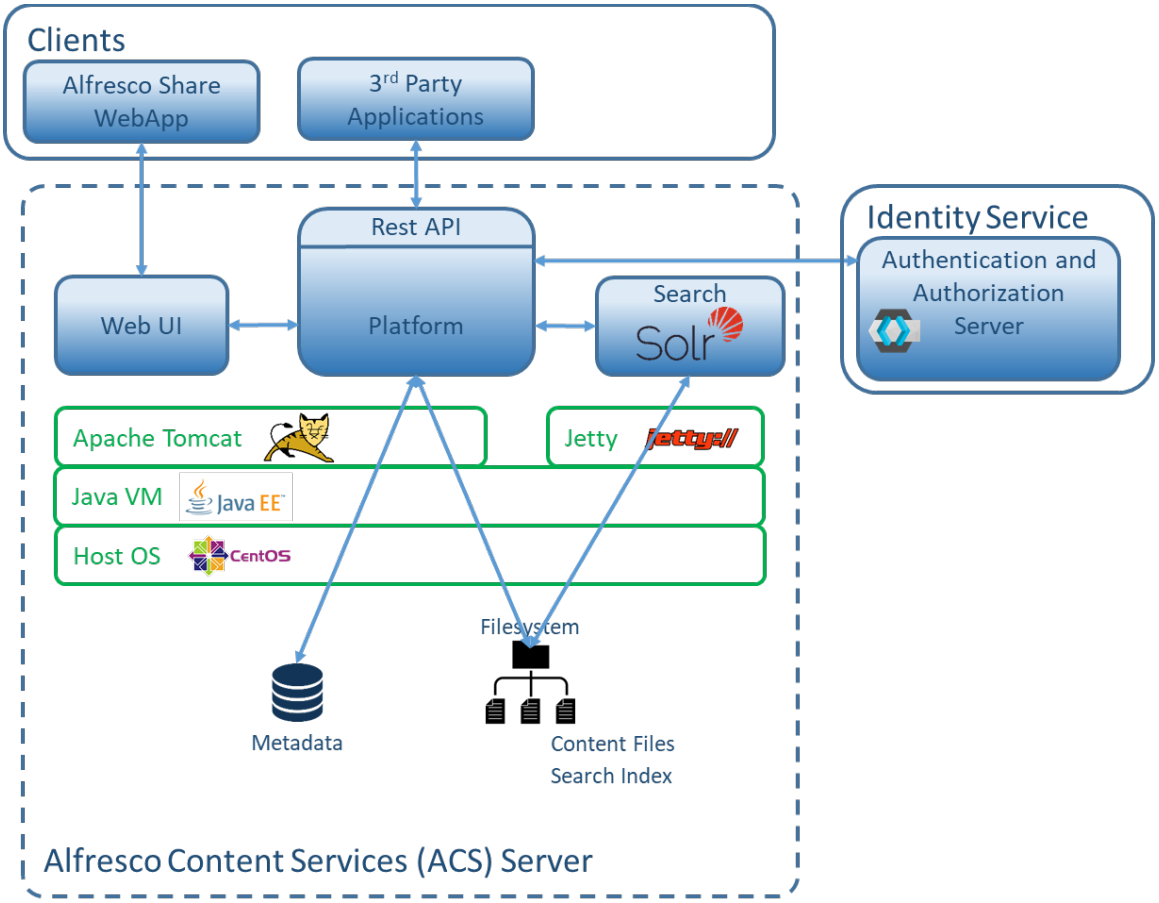


Figure 2. Collaboration Hub Architecture

As mentioned above the FORESIGHT Identity Service is built on Keycloak and uses the OAuth 2.0 standard. Figure 3 illustrates the authentication flow and includes the [10]:

- **Resource Owner**: FORESIGHT user that owns the data to be shared (i.e. folders and files to be shared through its Repository content);
- **Resource Server**: The Alfresco Content Services (ACS) server hosting the protected resources. As shown, it is also capable of dealing with access tokens in protected resource requests;
- **Client Application**: The Alfresco Share client application requesting access to the protected resources (ACS repository content) on behalf of the resource owner;
- **Authorization and Authorization Server**: JBoss Keycloak authorization server, authorizing the client application to access the protected resources of the resource owner. It issues access tokens to the client.

The access token represents the credentials to access protected resources. As a result, the Alfresco Share application stores the access token instead of the resource owner credentials. Consequently, the FORESIGHT authentication and authorization service does not rely on a username/password combination being broadcast but instead it relies on a token exchange mechanism.

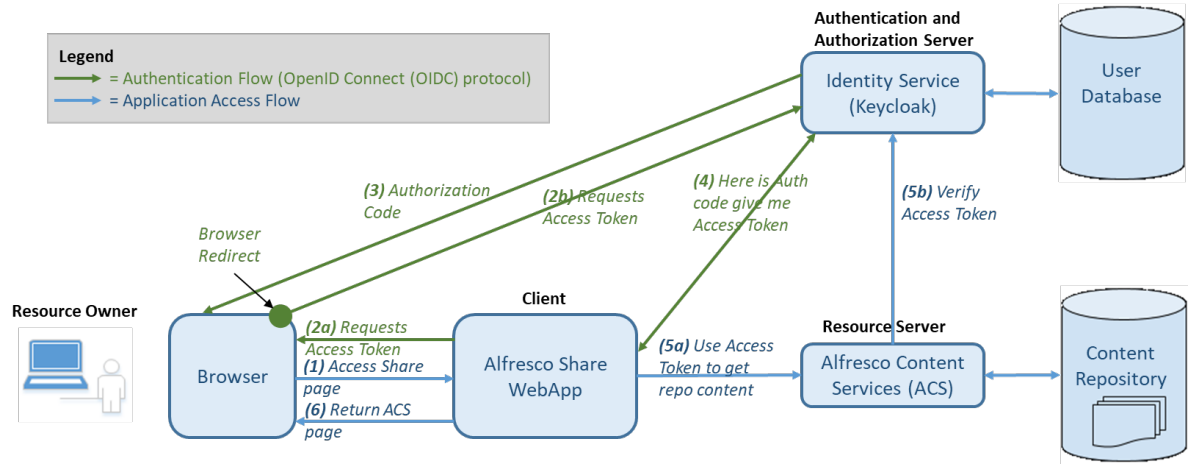


Figure 3. Collaboration Hub Authentication and Application Access Flow

As shown above, the client (i.e. the Alfresco Share WebApp) should be aware of the location of the authorization server and what redirection URL to use. Each resource server (i.e. ACS) should also be aware of the location of the authorization server so it can verify the access token being used to access a protected resource. The authorization server is connected to a database that contains user credentials information.

3.3 Interfaces Model

3.3.1 Communication Hub

If components need to communicate with the FORESIGHT Communication Hub (RocketChat) to access users and their details, communication channels and groups, chat messages and more, the RocketChat Representational State Transfer (REST) Application Programming Interface (API) calls should be used which are documented in [12].

However, we should ensure that the server hosting the FORESIGHT Communication Hub, is running via HTTPS and has a valid SSL Certificate. This is needed as the login method requires to post the username and password in plaintext. Consequently, it is suggested to call the REST login API only over HTTPS.

Before any call is made to the RocketChat REST API endpoints, authentication should be performed. If authentication is successful an authentication token (*authToken*) is returned that can be used in subsequent calls to the API. However, the ticket is valid for a specific time, so if no calls are made for a while, a 401 error is returned, indicating that a new re-authentication should be done in order to receive a new ticket.

To authenticate the following URL should be used, in an HTTP POST request, including the administrator username and password as data ({"user":"user1","password":"pass1"}):

https://foresight.eurodyn.com/chat/api/v1/login

The response is an authentication token inside a JavaScript Object Notation (JSON) object, like the one below:

```
{
    "status": "success",
    "data": {
        "userId": "xBMZQ98x9x8E9QNvX",
```

```
        "authToken": "KfYV7Vt_MUx7m6buKaVHbld-EQSw_ZsQ5uAWtL4RkTZ",
        "me": {
            "_id": "xBMZQ98x9x8E9QNvX",
            "services": {
                "password": {
                    "bcrypt":
"$2b$10$3bDy7hTzSwKQ19lvgjyM/O6QsATAo4jt20oSEPH5qjhp/xBp8d9Ia"
                },
                "email2fa": {
                    "enabled": true
                }
            },
            "emails": [
                {
                    "address": "alkis.gian@gmail.com",
                    "verified": false
                }
            ],
            "status": "online",
            "active": true,
            "_updatedAt": "2021-02-18T13:38:46.563Z",
            "roles": [
                "admin"
            ],
            "name": "Alkiviadis Giannakoulias",
            "statusConnection": "online",
            "username": "admin",
            "utcOffset": 2,
            "avatarUrl": "https://foresight.eurodyn.com/chat/chat/avatar/admin",
            "settings": {
                "preferences": {
                    "enableAutoAway": true,
                    "idleTimeLimit": 300,
                    "desktopNotificationRequireInteraction": false,
                    "audioNotifications": "mentions",
                    "desktopNotifications": "all",
                    "mobileNotifications": "all",
                    "unreadAlert": true,
                    "useEmojis": true,
                    "convertAsciiEmoji": true,
                    "autoImageLoad": true,
                    "saveMobileBandwidth": true,
                    "collapseMediaByDefault": false,
                    "hideUsernames": false,
                    "hideRoles": false,
                    "hideFlexTab": false,
                    "hideAvatars": false,
                    "sidebarGroupByType": true,
                    "sidebarViewMode": "medium",
                    "sidebarHideAvatar": false,
                    "sidebarShowUnread": false,
                    "sidebarSortby": "activity",
                    "showMessageInMainThread": false,
                    "sidebarShowFavorites": true,
                    "sendOnEnter": "normal",
                    "messageViewMode": 0,
                    "emailNotificationMode": "mentions",
                    "newRoomNotification": "door",
```

```
                    "newMessageNotification": "chime",
                    "muteFocusedConversations": true,
                    "notificationsSoundVolume": 100,
                    "sidebarShowDiscussion": true
                }
            }
        }
    }
}
```

Alternatively, personal access tokens could be used, so REST API calls can be made without the need to sign in. RocketChat supports that from the personal access tokens area found in the administrator profile page. Clicking the "Add a new Personal Access Token" results in Figure 4. However, since the generate tokens are irrecoverable, they must be stored in a safe place, such as secure storage location that the operating system offers and limit access to that storage.

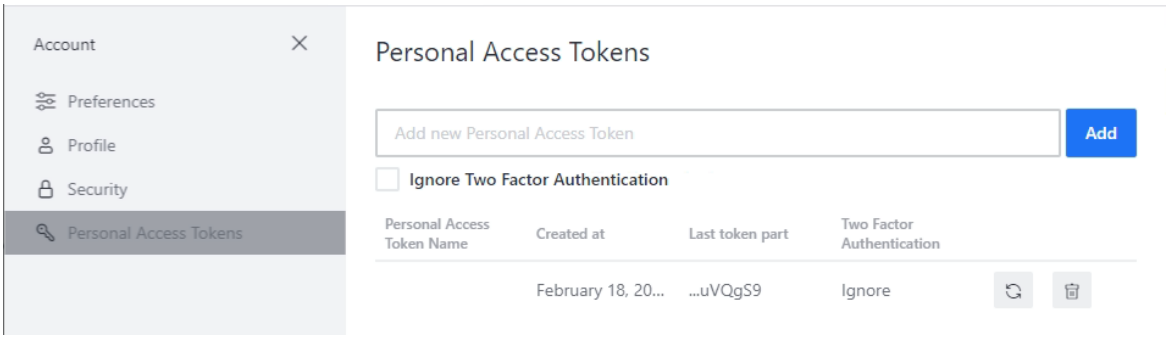


Figure 4. Communication Hub – Personal Access Token

Subsequently any request that is made to the REST API, should include the header `X-Auth-Token` along with the user id `X-User-Id`, as shown below:

```
curl                --location                --request                GET
'https://foresight.eurodyn.com/chat/api/v1/users.list' \
--header 'Accept: application/json' \
--header 'Content-Transfer-Encoding: application/json' \
--header 'X-Auth-Token: KfYV7Vt_MUx7m6buKaVHbld-EQSw_ZsQ5uAWtL4RkTZ' \
--header 'X-User-Id: xBMZQ98x9x8E9QNvX'
```

The above command will return all users and their information, including unique user id, name, surname, email address, status, account created time, last updated time, last login time, roles, rooms, avatar, etc .

3.3.2 Collaboration Hub

If components need to communicate with the FORESIGHT Collaboration Hub (Alfresco Community Edition) to access the repository content, relevant REST API calls should be used and specifically the Alfresco REST API version 1.0.

The reference documentation for the Alfresco REST API is available in what is referred to as the API Explorer application, based on the OpenAPI initiative [11]. It provides full documentation for each

endpoint, and a **"Try it out!"** button so each method can be used. The REST API Explorer can be accessed here: https://foresight.eurodyn.com/api-explorer.

Figure 5 shows what the REST API Explorer looks like.

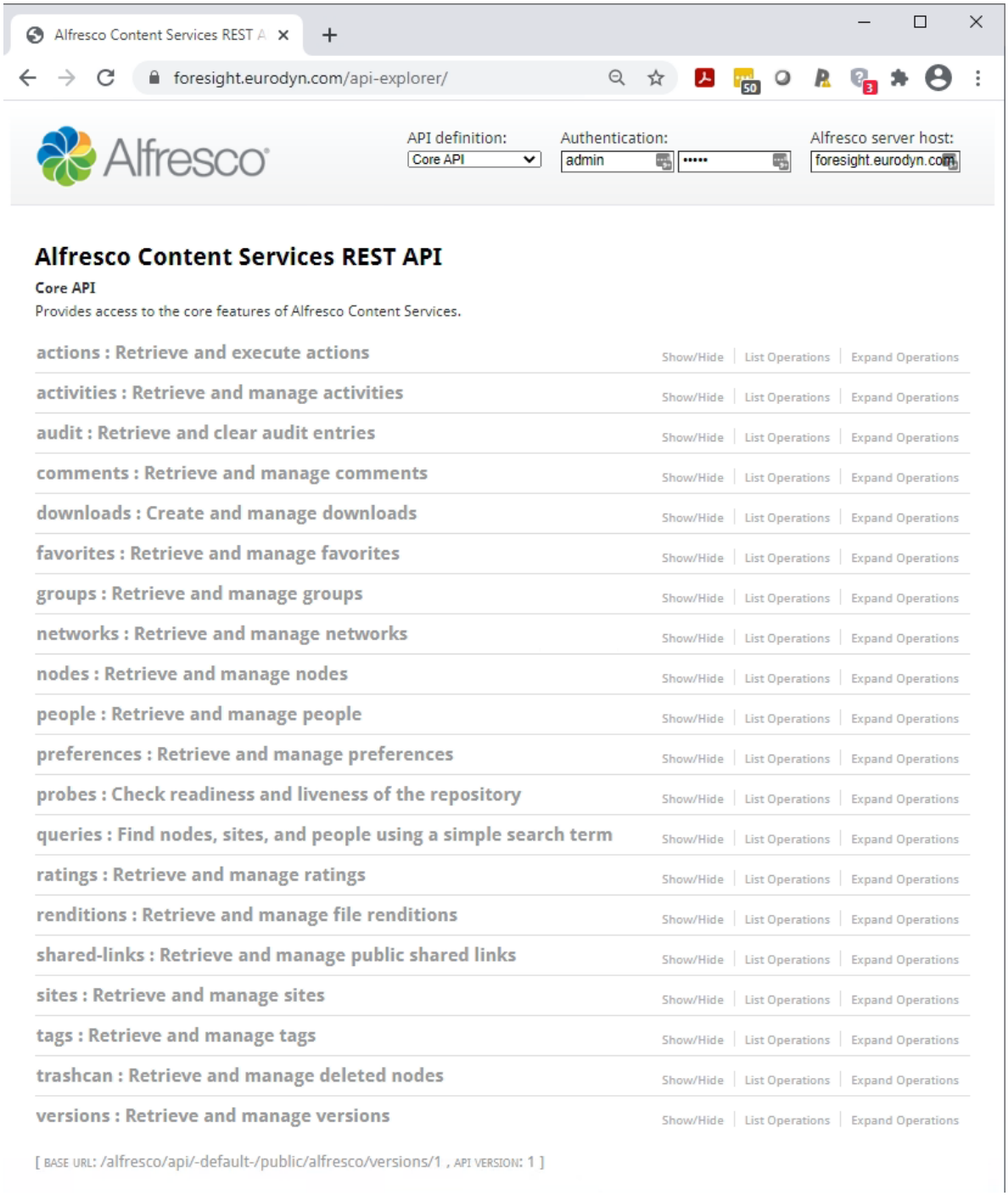


Figure 5. Alfresco REST API Explorer

The Alfresco REST API version 1.0 is a complete application interface providing access to all the features of the Alfresco Repository. The endpoint to access the API has the following format [11]:

Figure 6. Alfresco REST API version 1.0 Endpoint Format

Before any call is made to the Alfresco REST API endpoints, authentication with the Repository should be performed. If authentication is successful a ticket is returned that can be used in subsequent calls to the API. However, the ticket is valid for a specific time, so if no calls are made for a while, a 401 error is returned, indicating that a new re-authentication should be done in order to receive a new ticket.

To authenticate with the Repository the following URL should be used, in an HTTP POST request, including the username and password as data ({"userId":"user1","password":"pass1"}):

https://foresight.eurodyn.com/alfresco/api/-default-/public/authentication/versions/1/tickets

The response is a ticket inside a JSON object, like the one below:

```
{
    "entry": {
        "id": "TICKET_c6db3c0fcece4fc54d9d4d71189039aecd431f4e",
        "userId": "admin"
    }
}
```

Next the ticket should be base64 encoded before it can be used in subsequent calls, using the command below:

```
 $ echo -n 'TICKET_c6db3c0fcece4fc54d9d4d71189039aecd431f4e' | openssl base64
VElDS0VUX2M2ZGIzYzBmY2VjZTRmYzU0ZDlkNGQ3MTE4OTAzOWFlY2Q0MzFmNGU=
```

Subsequently any request that is made to the API, should include the base64 encoded ticket on the Authorization header as shown below:

```
curl -X GET -H 'Accept: application/json' -H 'Authorization: Basic
VElDS0VUX2M2ZGIzYzBmY2VjZTRmYzU0ZDlkNGQ3MTE4OTAzOWFlY2Q0MzFmNGU='
'https://foresight.eurodyn.com/alfresco/api/discovery' | jq
```

4 Prototype Implementation

4.1 Nginx HTTPS Reverse Proxy

4.1.1 Prerequisites and Installation

The hardware and operating system prerequisites are:

- A 2-core processor
- 4GB RAM Memory
- 50GB of disk space or more

The software prerequisites include:

- Centos 7 Operative System (OS);
- docker and docker-compose;
- Let's Encrypt SSL/TLS Certificates;

The following steps are executed within the NGNIX server virtual machine (VM):

1. Install Docker. According to the Docker installation[16] we should:
   - Set up the Docker repository by typing:

```
$sudo yum install -y yum-utils
$sudo yum-config-manager \
    --add-repo \
    https://download.docker.com/linux/centos/docker-ce.repo
```

   - Install the latest version of Docker Engine and containerd by typing:

```
$sudo yum install docker-ce docker-ce-cli containerd.io
```

   - Finally start and enable docker by typing:

```
$sudo systemctl start docker
$sudo systemctl enable docker
```

2. Install Docker compose. Docker Compose is a tool for defining and running multi-container Docker applications. With Compose, we use a YAML Ain't Markup Language (YAML) file to configure our application's services. Then, with a single command, we create and start all the services from our configuration. According to the docker compose installation[17] we should:
   - Download the current stable release of Docker Compose, by typing:

```
$ curl -L
"https://github.com/docker/compose/releases/download/1.28.2/docker-
compose-$(uname -s)-$(uname -m)" -o /usr/local/bin/docker-compose
```

   - Apply executable permissions to the binary, by typing:

```
$sudo chmod +x /usr/local/bin/docker-compose
```

3. Use the following yml file (docker-compose-nginx.yml), located under /opt/nginx/ to configure all aspects of the nginx container:

```
version: '3.7'
```

[16] https://docs.docker.com/engine/install/centos/
[17] https://docs.docker.com/compose/install/

```
networks:
  foresight-network:
    driver: bridge

services:
  foresight-nginx:
    image: nginx:1.19.5-alpine
    container_name: foresight-nginx
    restart: unless-stopped
    volumes:
      - ./docker-data/nginx:/etc/nginx/conf.d
      - ./docker-data/certbot/conf:/etc/letsencrypt
      - ./docker-data/certbot/www:/var/www/certbot
    ports:
      - "80:80"
      - "443:443"
    networks:
      - foresight-network
    command: "/bin/sh -c 'while :; do sleep 6h & wait $${!}; nginx -s reload;
done & nginx -g \"daemon off;\"'"

  foresight-certbot:
    image: certbot/certbot:v1.9.0
    restart: unless-stopped
    volumes:
      - ./docker-data/certbot/conf:/etc/letsencrypt
      - ./docker-data/certbot/www:/var/www/certbot
    entrypoint: "/bin/sh -c 'trap exit TERM; while :; do certbot renew; sleep
12h & wait $${!}; done;'"
```

4. Create the following directory structure /opt/nginx/docker-data/nginx and place inside it the following file (app.conf)

```
server {
    listen 80;
    server_name foresight.eurodyn.com;
    server_tokens off;

    location / {
        return 301 https://$host$request_uri;
    }

    location /.well-known/acme-challenge/ {
        root /var/www/certbot;
    }
}

server {
    listen 443 ssl;
    server_name foresight.eurodyn.com;

    ssl_certificate
/etc/letsencrypt/live/foresight.eurodyn.com/fullchain.pem;
    ssl_certificate_key
/etc/letsencrypt/live/foresight.eurodyn.com/privkey.pem;
```

```
    include /etc/letsencrypt/options-ssl-nginx.conf;
    ssl_dhparam /etc/letsencrypt/ssl-dhparams.pem;

    location / {
        proxy_pass http://foresight.eurodyn.com:8080;
        proxy_no_cache 1;
        proxy_cache_bypass 1;
    }
}
```

5. Use the following script (init-letsencrypt.sh) to obtain a let's encrypt ssl certificate:

```
#!/bin/bash

## The following shell script is a variation of the shell script provided in
the GitHub repo: https://github.com/wmnnd/nginx-certbot

if ! [ -x "$(command -v docker-compose)" ]; then
  echo 'Error: docker-compose is not installed.' >&2
  exit 1
fi

domains=(foresight.eurodyn.com)
rsa_key_size=4096
data_path="./docker-data/certbot"
email="alkiviadis.giannakoulas@eurodyn.com"  #  Adding  a  valid  address  is
strongly recommended
staging=0 # Set to 1 if you're testing your setup to avoid hitting request
limits

if [ -d "$data_path" ]; then
  read -p "Existing data found for $domains. Continue and replace existing
certificate? (y/N) " decision
  if [ "$decision" != "Y" ] && [ "$decision" != "y" ]; then
    exit
  fi
fi


if   [   !   -e   "$data_path/conf/options-ssl-nginx.conf"   ]   ||   [   !   -e
"$data_path/conf/ssl-dhparams.pem" ]; then
  echo "### Downloading recommended TLS parameters ..."
  mkdir -p "$data_path/conf"
  curl  -s  https://raw.githubusercontent.com/certbot/certbot/master/certbot-
nginx/certbot_nginx/_internal/tls_configs/options-ssl-nginx.conf            >
"$data_path/conf/options-ssl-nginx.conf"
  curl                                                                      -s
https://raw.githubusercontent.com/certbot/certbot/master/certbot/certbot/ssl-
dhparams.pem > "$data_path/conf/ssl-dhparams.pem"
  echo
fi

echo "### Creating dummy certificate for $domains ..."
path="/etc/letsencrypt/live/$domains"
mkdir -p "$data_path/conf/live/$domains"
docker-compose -f docker-compose-nginx.yml run --rm --entrypoint "\
  openssl req -x509 -nodes -newkey rsa:$rsa_key_size -days 1\
    -keyout '$path/privkey.pem' \
```

```
    -out '$path/fullchain.pem' \
    -subj '/CN=localhost'" foresight-certbot
echo

echo "### Starting nginx ..."
docker-compose -f docker-compose-nginx.yml up --force-recreate -d foresight-
nginx
echo

echo "### Deleting dummy certificate for $domains ..."
docker-compose -f docker-compose-nginx.yml run --rm --entrypoint "\
  rm -Rf /etc/letsencrypt/live/$domains && \
  rm -Rf /etc/letsencrypt/archive/$domains && \
  rm -Rf /etc/letsencrypt/renewal/$domains.conf" foresight-certbot
echo

echo "### Requesting Let's Encrypt certificate for $domains ..."
#Join $domains to -d args
domain_args=""
for domain in "${domains[@]}"; do
  domain_args="$domain_args -d $domain"
done

# Select appropriate email arg
case "$email" in
  "") email_arg="--register-unsafely-without-email" ;;
  *) email_arg="--email $email" ;;
esac

# Enable staging mode if needed
if [ $staging != "0" ]; then staging_arg="--staging"; fi

docker-compose -f docker-compose-nginx.yml run --rm --entrypoint "\
  certbot certonly --webroot -w /var/www/certbot \
    $staging_arg \
    $email_arg \
    $domain_args \
    --rsa-key-size $rsa_key_size \
    --agree-tos \
    --force-renewal" foresight-certbot
echo

echo "### Reloading nginx ..."
docker-compose  -f  docker-compose-nginx.yml  exec  foresight-nginx  nginx  -s
reload
```

Then issue the following commands:

```
$cd /opt/
$sudo chmod +x init-letsencrypt.sh
$./init-letsencrypt.sh
```

4.2 FORESIGHT Identity Service

4.2.1 Prerequisites and Installation

The hardware and operating system prerequisites are:

- A 2-core processor
- 4GB RAM Memory
- 50GB of disk space or more

The software prerequisites include:

- Centos 7 Operative System (OS);
- docker and docker-compose;

The following steps are executed within the Keycloak server virtual machine (VM):

1. Install Docker.
2. Install Docker compose.
3. Update the nginx configuration by adding the following entry in the server section (443 ssl) of the app.conf file described above:

```
    location /auth/ {
        proxy_pass          http://foresight.eurodyn.com:8090/auth/;
        proxy_set_header    Host               $host;
        proxy_set_header    X-Real-IP          $remote_addr;
        proxy_set_header    X-Forwarded-For    $proxy_add_x_forwarded_for;
        proxy_set_header    X-Forwarded-Host   $host;
        proxy_set_header    X-Forwarded-Server $host;
        proxy_set_header    X-Forwarded-Port   $server_port;
        proxy_set_header    X-Forwarded-Proto  $scheme;
    }
```

4. Use the following yml file (docker-compose-keycloak.yml), located under /opt/keycloak/ to configure all aspects of the keycloak container:

```
version: '3.7'

networks:
  foresight-network:
    driver: bridge

services:
  foresight-keycloak:
    image: jboss/keycloak:11.0.3
    container_name: foresight-keycloak
    environment:
      - KEYCLOAK_USER=${KEYCLOAK_ADMIN_USERNAME}
      - KEYCLOAK_PASSWORD=${KEYCLOAK_ADMIN_PASSWORD}
      - PROXY_ADDRESS_FORWARDING=true
    restart: unless-stopped
    volumes:
      - ${KEYCLOAK_THEME_PATH}:/opt/jboss/keycloak/themes/custom-theme
    ports:
      - "${KEYCLOAK_PORT}:8080"
    networks:
      - foresight-network
```

```
```

5. Create the following .env file under /opt/keycloak directory

```
# OAuth2 / Keycloak
OAUTH2_CLIENT=foresight-client
OAUTH2_CLIENT_SECRET=1eee1474-950a-466a-9991-6b5ae9f737dc
KEYCLOAK_ADMIN_USERNAME=admin
KEYCLOAK_ADMIN_PASSWORD=admin
KEYCLOAK_THEME_PATH=/opt/keycloak-theme

# Ports
KEYCLOAK_PORT=8090
```

6. Execute

```
$ docker-compose -f docker-compose-keycloak.yml up
```

7. If no errors are displayed, we can access Keycloak web app through: https://foresight.eurodyn.com/auth

Resulting in the next screen to appear:

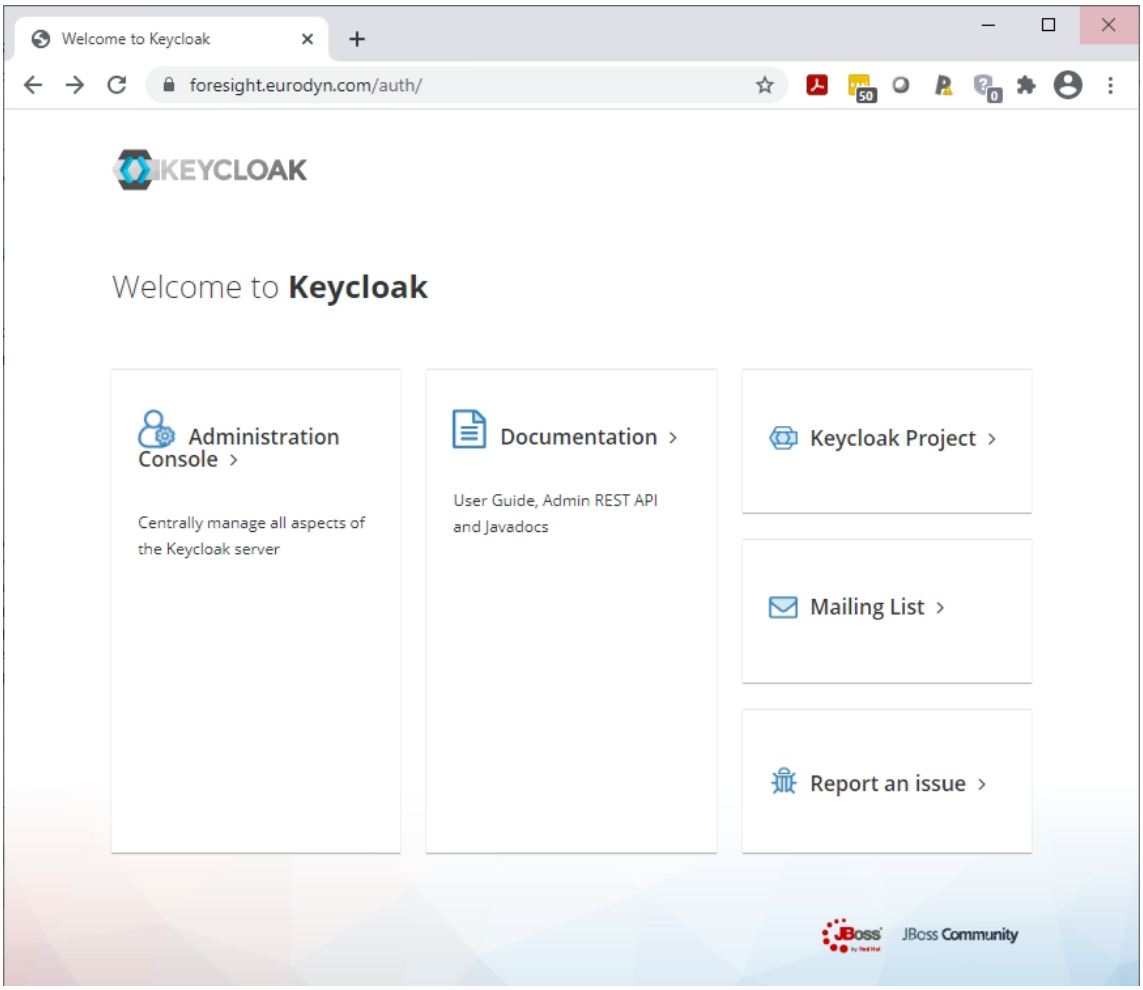


Figure 7. Identify Service

Clicking on the "Administration Console" redirects us to the custom login page shown below, where we have to use the credentials as per .env file.

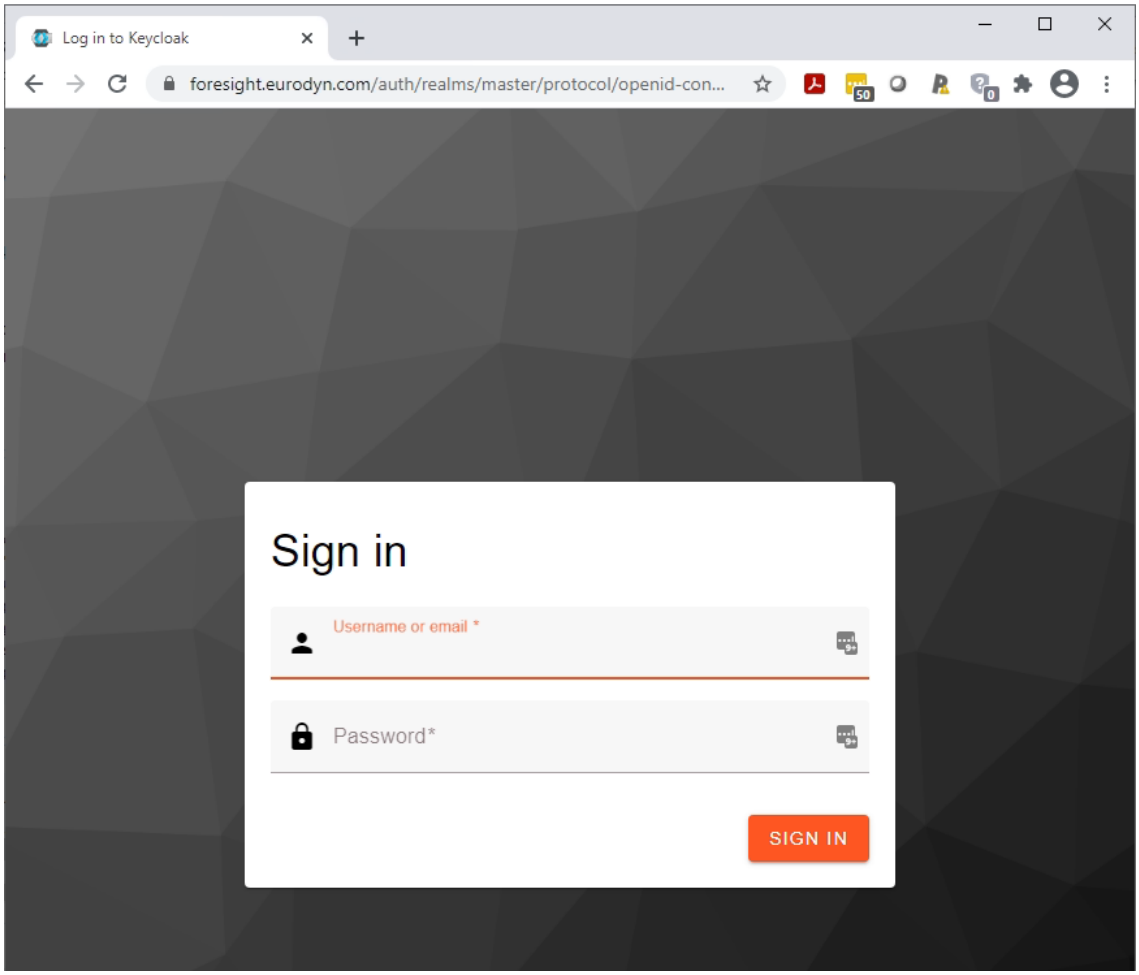


Figure 8. Identify Service – Login Page

As soon as we hit the “Sign In” button we are re-directed to the identity service main page, shown below.

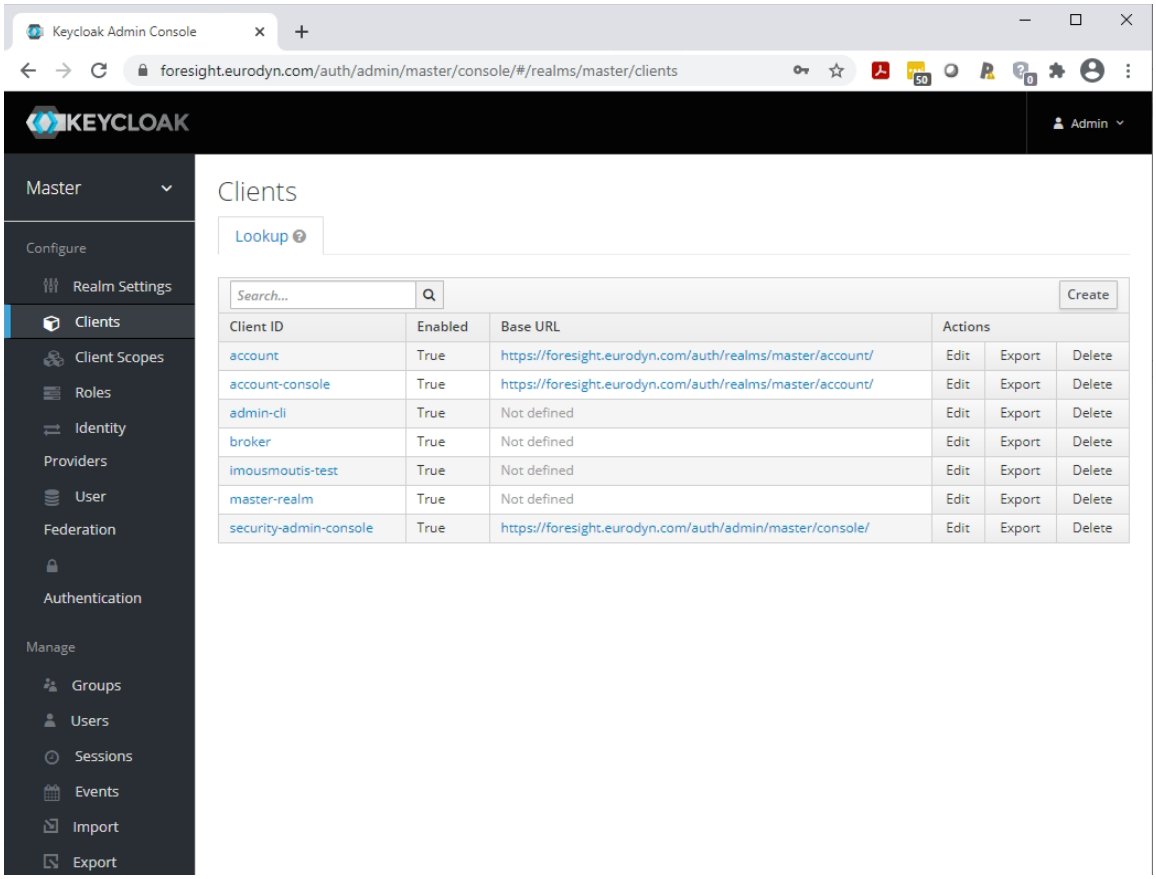


Figure 9. Identity Service – Main Page

4.3 Communication Hub

4.3.1 Prerequisites and Installation

The hardware and operating system prerequisites are:

- A 2-core processor
- 4GB RAM Memory
- 50GB of disk space or more

The software prerequisites/dependencies include:

- Centos 7 Operative System (OS);
- GCC (C and C++ Compiler) and Development Tools;
- curl;
- MongoDB v4.4 database;
- npm (Node Package Manager) the default package manager for NodeJS;
- NodeJS v12.x a cross-platform JavaScript run-time environment;
- EPEL repository;
- GraphicsMagick rpm package;

To install the necessary dependencies:

1. Install build tools by typing:

```
$sudo yum install -y gcc-c++ make
```

2. Install curl by typing:

```
$sudo yum install -y curl
```

3. Update package list and configure yum to install the official MongoDB packages:

```
cat << EOF | sudo tee -a /etc/yum.repos.d/mongodb-org-4.4.repo
[mongodb-org-4.4]
name=MongoDB Repository
baseurl=https://repo.mongodb.org/yum/redhat/7/mongodb-org/4.4/x86_64/
gpgcheck=1
enabled=1
gpgkey=https://www.mongodb.org/static/pgp/server-4.4.asc
EOF
```

Install MongoDB by typing:

```
$sudo yum install -y mongodb-org
```

4. Install the NodeSource yum repository to the system, by typing:

```
$sudo curl -sL https://rpm.nodesource.com/setup_12.x | sudo bash -
```

Install NodeJS and npm by typing:

```
$ sudo yum install -y nodejs
```

5. Install Extra Packages for Enterprise Linux repository configuration by typing:

```
$ sudo yum install -y epel-release
```

6. Install GraphicsMagick (image processing CLI tool) by typing:

```
$ sudo yum install -y GraphicsMagick
```

7. Install inherits and n, and the node version required by RocketChat, using npm, by typing:

```
$ sudo npm install -g inherits n && sudo n 12.18.4
```

To install RocketChat we should first download and extract the latest RocketChat version by typing:

```
$curl -L https://releases.rocket.chat/latest/download -o /tmp/rocket.chat.tgz
$tar -xzf /tmp/rocket.chat.tgz -C /tmp
```

Install RocketChat under the /opt/ directory using npm by typing:

```
$cd /tmp/bundle/programs/server && npm install
$sudo mv /tmp/bundle /opt/Rocket.Chat
```

Once installation is complete, we should configure the RocketChat service. Initially we should add the *rocketchat user*, set the right permissions on the RocketChat folder and create the RocketChat service file, by typing:

```
$sudo useradd -M rocketchat && sudo usermod -L rocketchat
$sudo chown -R rocketchat:rocketchat /opt/Rocket.Chat

cat << EOF |sudo tee -a /lib/systemd/system/rocketchat.service
[Unit]
Description=The FORESIGHT RocketChat server
After=network.target     remote-fs.target     nss-lookup.target     nginx.target
mongod.target
[Service]
ExecStart=/usr/local/bin/node /opt/Rocket.Chat/main.js
```

```
StandardOutput=syslog
StandardError=syslog
SyslogIdentifier=rocketchat
User=rocketchat
Environment=MONGO_URL=mongodb://localhost:27017/rocketchat?replicaSet=rs01
MONGO_OPLOG_URL=mongodb://localhost:27017/local?replicaSet=rs01
ROOT_URL=http://foresight-server.eurodyn.com:3000/chat/ PORT=3000
[Install]
WantedBy=multi-user.target
EOF
```

As shown from the above service configuration RocketChat runs in a subfolder (/chat/) as indicated by the environment variable ROOT_URL. This is necessary since we should have a different base path (context path) in order to be proxied.

Following we should Setup the storage engine and replication for MongoDB by typing:

```
$sudo sed -i "s/^#  engine:/  engine: mmapv1/"  /etc/mongod.conf
$sudo   sed   -i   "s/^#replication:/replication:\n      replSetName:   rs01/"
/etc/mongod.conf
```

Next we should enable and start MongoDB and RocketChat services by typing:

```
$sudo systemctl enable mongod && sudo systemctl start mongod
$mongo --eval "printjson(rs.initiate())"

$sudo systemctl enable rocketchat && sudo systemctl start rocketchat
```

Additionally we could test whether the services are running as expected, by typing:

```
$systemctl status mongod
```

The response should be:

```
● mongod.service - MongoDB Database Server
   Loaded: loaded (/usr/lib/systemd/system/mongod.service; enabled; vendor preset: disabled)
   Active: active (running) since Thu 2021-02-04 15:53:43 EET; 1 weeks 6 days ago
     Docs: https://docs.mongodb.org/manual
 Main PID: 32152 (mongod)
    Tasks: 55
   Memory: 266.3M
   CGroup: /system.slice/mongod.service
           └─32152 /usr/bin/mongod -f /etc/mongod.conf
```

```
$systemctl status rocketchat.service
```

The response should be:

```
● rocketchat.service - The FORESIGHT Rocket.Chat server
   Loaded: loaded (/usr/lib/systemd/system/rocketchat.service; enabled; vendor preset: disabled)
   Active: active (running) since Thu 2021-02-04 17:51:55 EET; 1 weeks 6 days ago
 Main PID: 6329 (node)
    Tasks: 11
   Memory: 507.8M
   CGroup: /system.slice/rocketchat.service
           └─6329 /usr/local/bin/node /opt/Rocket.Chat/main.js
```

Finally, we should point our web browser to https://foresight.eurodyn.com/chat, and then register the first user for administration. Once this matter is settled and we access the communication hub the following screen is presented.

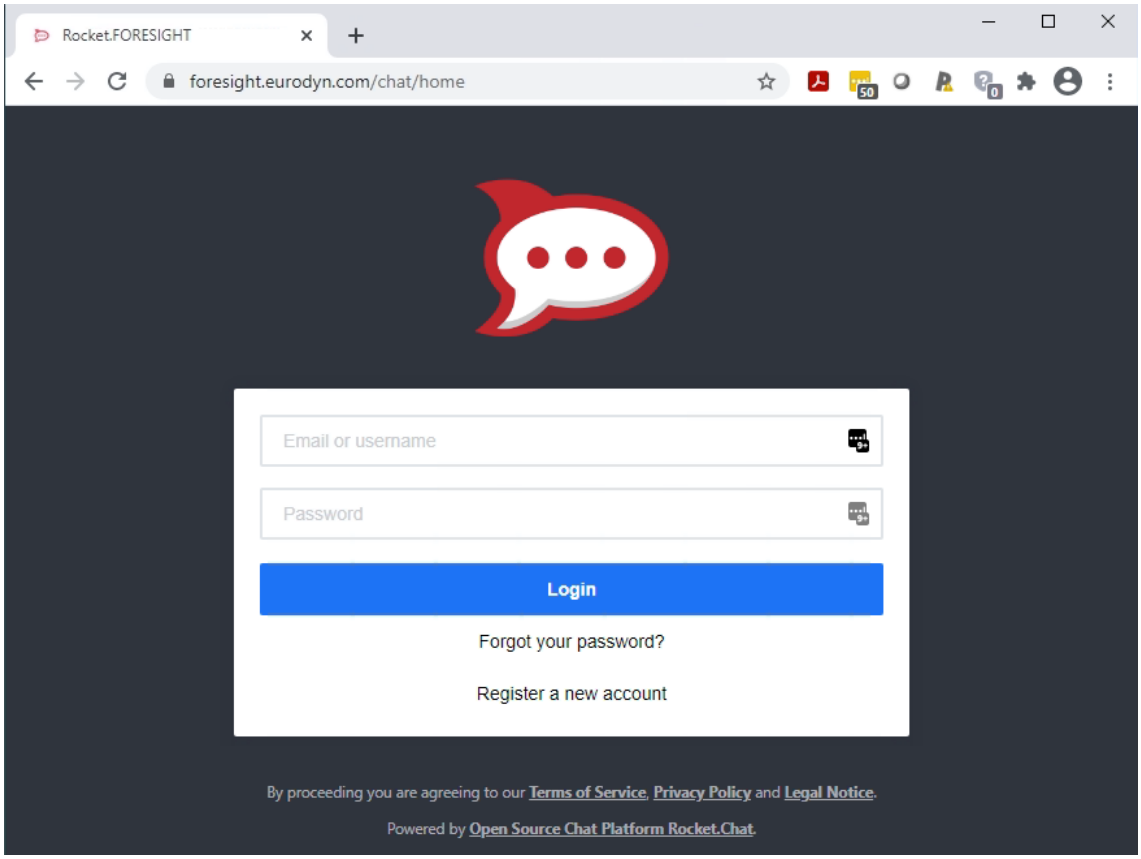


Figure 10. Communication Hub – Login Screen

4.3.2 Configuring Keycloak Authentication

To integrate RocketChat with Keycloak the following steps should be followed.

Initially we should create a client in Keycloak, following the steps described in [15]. Figure 11 shows the minimal configurations needed to setup Keycloak as an Identity Provider to RocketChat.

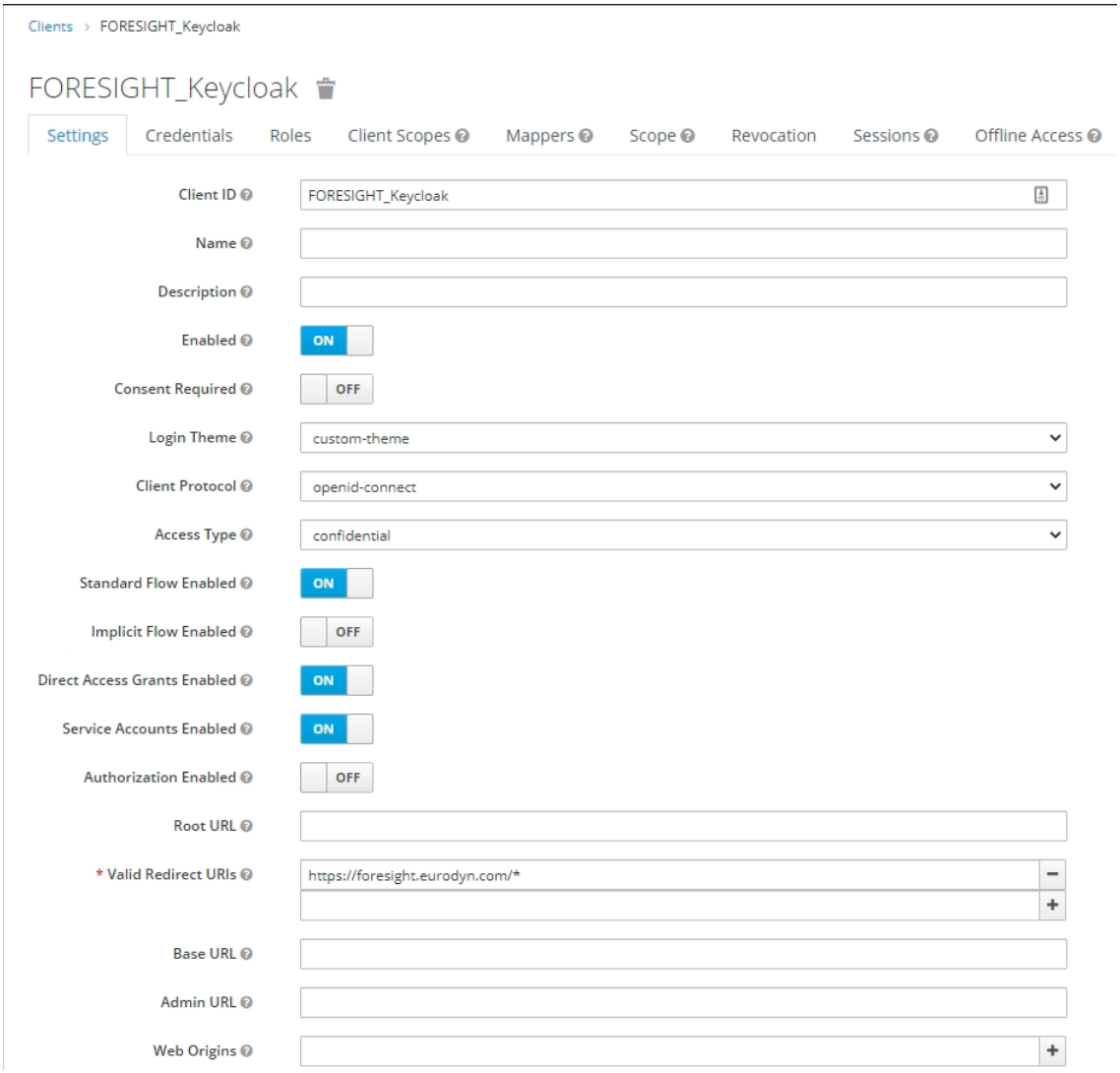


Figure 11. Keycloak Client Configuration

As soon as we save the changes, a new set of client secrets (credentials) will be created and be made available to us under the credentials tab. The client secrets will be used later on when configuring RocketChat.

To configure RocketChat we should initially login with an administrator account, navigate to OAuth page, shown in Figure 12, and click the “Add custom OAuth button”, resulting in a dialog window (Figure 13) asking to provide a unique name for the custom OAuth. Once we type ”Keycloak” and press the “Send” button the OAuth page is updated with the new entry, allowing us to configure it, as per [15] instructions.

At the end, if configuration was successful, we should enable the new Keycloak provider, and logout from RocketChat in order to view the keycloak based login option visible in the login page, as shown in Figure 14.

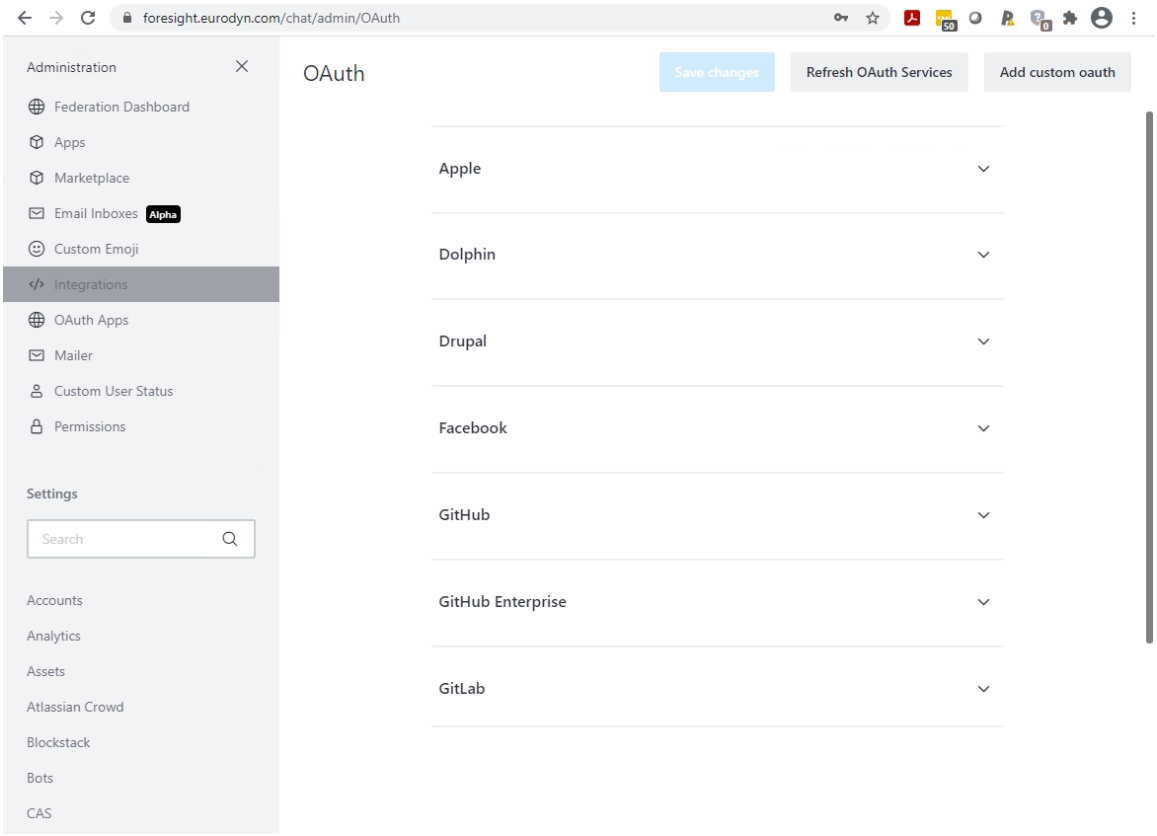


Figure 12. RocketChat OAuth

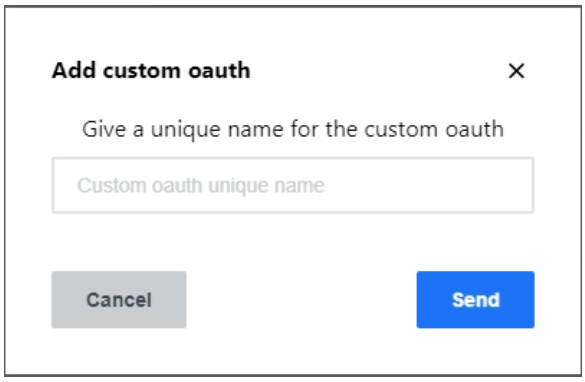


Figure 13. RocketChat – Add Custom OAuth

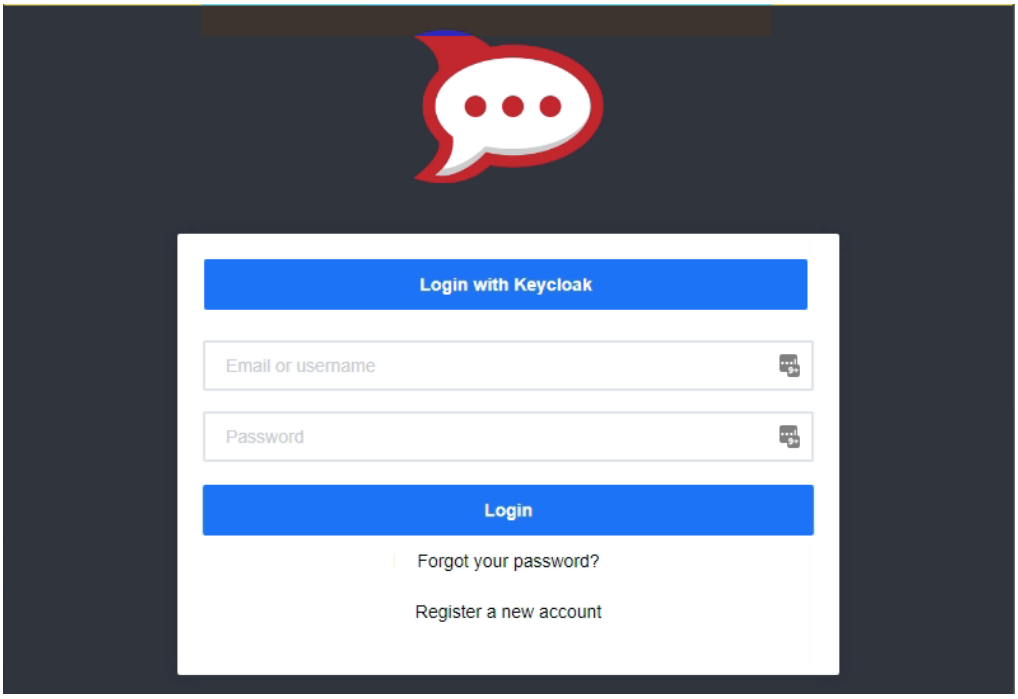


Figure 14. RocketChat Login Screen (Keycloak OAth)

4.4 Collaboration Hub

As an entirely Java application, the collaboration hub (Alfresco Community Edition system) runs on virtually any system that can run Java Enterprise Edition. There are two main options for deploying Alfresco Community Edition: using containerized deployment or using the distribution zip. In this first version of the cyber-team collaborative (CTC) tools we will deploy Alfresco Community Edition using Docker images that are packaged in Docker Compose. Figure 15 shows how Alfresco Content Services look when deployed with Docker.

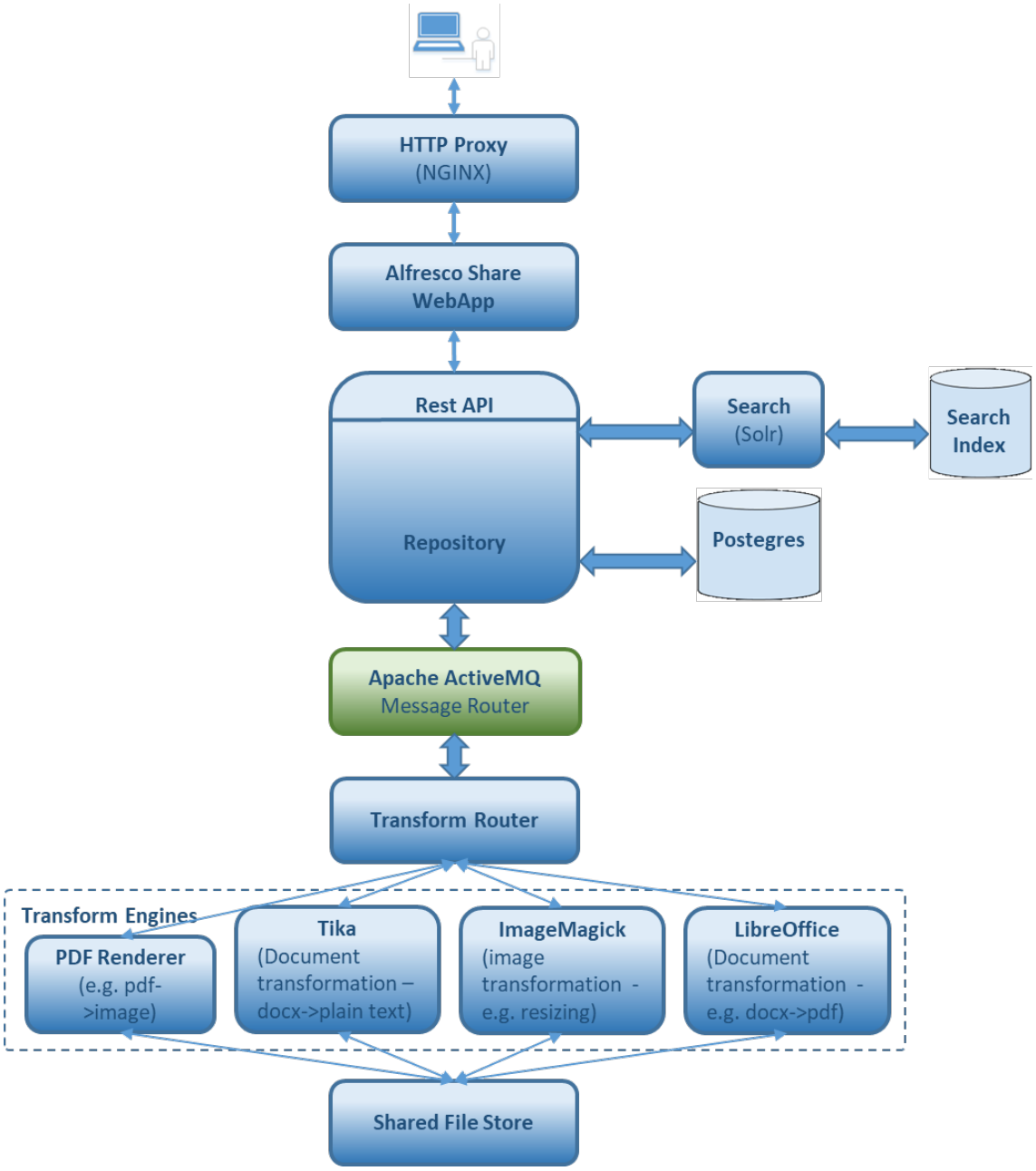


Figure 15. Collaboration Hub – Deployment Architecture (Docker)

However, since Alfresco Community Edition does not support integration with keycloak, an add-on[18] was used aiming to provide Keycloak-related extensions / customisations for the Alfresco Repository and Share tiers.

We can either built the add-on by downloading and building the source files or we can get it from Maven Central by downloading repositories: a) de.acosix.alfresco.keycloak and b) de.acosix.alfresco.utility.

After the add-on was obtained, we should built custom docker images for alfresco and share as described in the section below.

[18] https://github.com/Acosix

4.4.1 Prerequisites and Installation

The hardware and operating system prerequisites are:

- A 2-core processor
- 4GB RAM Memory
- 100GB of disk space or more

The software prerequisites include:

- Centos 7 Operative System (OS);
- docker and docker-compose;
- de.acosix.alfresco.utility.share-1.2.5.amp file[19];
- de.acosix.alfresco.utility.repo-1.2.5.amp file[19];
- de.acosix.alfresco.keycloak.share-1.1.0-rc5.amp[19];
- de.acosix.alfresco.keycloak.repo-1.1.0-rc5.amp[19];

The following steps are executed within the collaboration server virtual machine (VM):

1. Install Docker.
2. Install Docker compose.

4.4.1.1 Custom Alfresco Repository

In order to include the amp files in the Docker based deployment, we should build a custom Docker image with the modules included and installed by running the **alfresco-mmt tool**. The documentation in the ACS packaging project[20] about creating customised images was used. Consequently, to create the **custom alfresco repository** we used the following folder and file structure:

```
custom-alfresco-repository                                  (Dir)
│
└──Dockerfile                                               (File)
│
└──amps                                                     (Dir)
    │
    └──de.acosix.alfresco.keycloak.repo-1.1.0-rc5.amp       (File)
    │
    └──de.acosix.alfresco.utility.repo-1.2.5.amp            (File)
```

Add the following lines to the Dockerfile and replace “keycloak.adapter.resource” and “keycloak.adapter.credentials.secret” with the Keycloak created in 4.3.2:

```
### Apply AMPs to the latest Community repository image.
FROM alfresco/alfresco-content-repository-community:6.2.0-ga

# Default user and group are used to setup permissions for Tomcat process, see
parent Dockerfile: Alfresco/acs-community-packaging/docker-alfresco/Dockerfile
ARG GROUPNAME=Alfresco
ARG USERNAME=alfresco
ARG TOMCAT_DIR=/usr/local/tomcat
```

[19] At the time of writing this was the official release number

[20] https://github.com/Alfresco/acs-packaging/blob/master/docs/create-custom-image-using-existing-docker-image.md

```
# Alfresco user does not have permissions to modify webapps or configuration.
Switch to root.
# The access will be fixed after all operations are done.
USER root

# Add services configuration to alfresco-global.properties
RUN echo -e '\n\
sample.site.disabled=true\n\
\n\
authentication.chain=keycloak1:keycloak,alfrescoNtlm1:alfrescoNtlm\n\
\n\
keycloak.adapter.auth-server-url=https://foresight.eurodyn.com/auth\n\
keycloak.adapter.realm=master\n\
keycloak.adapter.resource=FORESIGHT_Keycloak\n\
keycloak.adapter.credentials.provider=secret\n\
keycloak.adapter.credentials.secret=244ceb1d-a371-48d5-8439-7bd1ee52c891\n\
\n\
# localhost in auth-server-url won't work for direct access in a Docker
deployment\n\
keycloak.authentication.directAuthHost=https://foresight.eurodyn.com\n\
\n\
keycloak.synchronization.userFilter.containedInGroup.property.groupPaths=/Test
A\n\
keycloak.synchronization.groupFilter.containedInGroup.property.groupPaths=/Test
A\n\
\n\
' >> ${TOMCAT_DIR}/shared/classes/alfresco-global.properties


### Copy the AMPs from build context to the appropriate location for your
application server
COPY amps ${TOMCAT_DIR}/amps

### Install AMPs on alfresco
RUN java -jar ${TOMCAT_DIR}/alfresco-mmt/alfresco-mmt*.jar install \
              ${TOMCAT_DIR}/amps/de.acosix.alfresco.utility.repo-1.2.5.amp
${TOMCAT_DIR}/webapps/alfresco -nobackup -force
RUN java -jar ${TOMCAT_DIR}/alfresco-mmt/alfresco-mmt*.jar install \
              ${TOMCAT_DIR}/amps/de.acosix.alfresco.keycloak.repo-1.1.0-rc5.amp
${TOMCAT_DIR}/webapps/alfresco -nobackup -force

# All files in the tomcat folder must be owned by root user and Alfresco group as
mentioned in the parent Dockerfile
RUN chgrp -R ${GROUPNAME} ${TOMCAT_DIR}/webapps && \
    find ${TOMCAT_DIR}/webapps -type d -exec chmod 0750 {} \; && \
    find ${TOMCAT_DIR}/webapps -type f -exec chmod 0640 {} \; && \
    chmod -R g+r ${TOMCAT_DIR}/webapps && \
    chgrp -R ${GROUPNAME} ${TOMCAT_DIR}

# Switching back to alfresco user after having added amps files to run the
container as non-root
USER ${USERNAME}
```

Open a Terminal and run the following command to build the custom docker image.

```
$cd /opt/alfresco/custom-alfresco-repository
$docker build . -t foresight/custom-alfresco-repository:0.0.1
```

4.4.1.2 Custom Share Repository

To create the **custom share repository,** we used the following folder and file structure:

```
custom-share-repository                                (Dir)
│
└──Dockerfile                                                (File)
│
└──amps                                                      (Dir)
    │
    └──de.acosix.alfresco.keycloak.share-1.1.0-rc5.amp       (File)
    │
    └──de.acosix.alfresco.utility.share-1.2.5.amp            (File)
```

Add the following lines to the Dockerfile and replace “<resource>” and “<secret>” with the Keycloak created in 4.3.2:

```
### Apply AMPs to the latest Community repository image.
FROM alfresco/alfresco-share:6.2.0

# Default user and group are used to setup permissions for Tomcat process, see
parent Dockerfile: Alfresco/acs-community-packaging/docker-alfresco/Dockerfile
ARG GROUPNAME=Alfresco
ARG USERNAME=alfresco
ARG TOMCAT_DIR=/usr/local/tomcat

# Alfresco user does not have permissions to modify webapps or configuration.
Switch to root.
# The access will be fixed after all operations are done.
USER root

# Update configuration in share-config-custom.xml
# delete last line
RUN sed -i '$d' ${TOMCAT_DIR}/shared/classes/alfresco/web-extension/share-config-
custom.xml

RUN echo -e '\n\
    <config evaluator="string-compare" condition="SecurityHeadersPolicy">\n\
        <headers>\n\
            <header>\n\
                <name>X-Frame-Options</name>\n\
                <value>SAMEORIGIN</value>\n\
            </header>\n\
            <header>\n\
                <name>X-Content-Type-Options</name>\n\
                <value>nosniff</value>\n\
            </header>\n\
            <header>\n\
                <name>X-XSS-Protection</name>\n\
                <value>1; mode=block</value>\n\
            </header>\n\
        </headers>\n\
    </config>\n\
    \n\
    <config evaluator="string-compare" condition="Keycloak">\n\
        <keycloak-auth-config>\n\
            <enhance-login-form>true</enhance-login-form>\n\
```

```
            <enable-sso-filter>true</enable-sso-filter>\n\
            <force-keycloak-sso>false</force-keycloak-sso>\n\
            <perform-token-exchange>true</perform-token-exchange>\n\
            <session-mapper-limit>2000</session-mapper-limit>\n\
        </keycloak-auth-config>\n\
        <keycloak-adapter-config>\n\
            <auth-server-url>https://foresight.eurodyn.com/auth</auth-server-
url>\n\
            <always-refresh-token>true</always-refresh-token>\n\
            <connection-pool-size>123</connection-pool-size>\n\
            <realm>master</realm>\n\
            <resource>FORESIGHT_Keycloak</resource>\n\
            <ssl-required>none</ssl-required>\n\
            <public-client>false</public-client>\n\
            <credentials>\n\
                <provider>secret</provider>\n\
                <secret>244ceb1d-a371-48d5-8439-7bd1ee52c891</secret>\n\
            </credentials>\n\
        </keycloak-adapter-config>\n\
    </config>\n\
\n\
</alfresco-config>\n\
' >> ${TOMCAT_DIR}/shared/classes/alfresco/web-extension/share-config-custom.xml

### Copy the AMPs from build context to the appropriate location for your
application server
COPY amps_share ${TOMCAT_DIR}/amps_share

### Install AMPs on alfresco
RUN java -jar ${TOMCAT_DIR}/alfresco-mmt/alfresco-mmt*.jar install \
              ${TOMCAT_DIR}/amps_share/de.acosix.alfresco.utility.share-
1.2.5.amp ${TOMCAT_DIR}/webapps/share -nobackup -force
RUN java -jar ${TOMCAT_DIR}/alfresco-mmt/alfresco-mmt*.jar install \
              ${TOMCAT_DIR}/amps_share/de.acosix.alfresco.keycloak.share-1.1.0-
rc5.amp ${TOMCAT_DIR}/webapps/share -nobackup -force
```

Open a Terminal and run the following command to build the custom docker image.

```
$cd /opt/alfresco/custom-share-repository
$docker build . -t foresight/custom-share-repository:0.0.1
```

Update the Dockerfile entries for alfresco service with the one below:

```
#image: alfresco/alfresco-content-repository-community:6.2.0-ga
image:  foresight/custom-alfresco-repository:0.0.1
```

Update the Dockerfile entries for share service with the one below:

```
#image: alfresco/alfresco-share:6.2.2
image:  foresight/custom-share-repository:0.0.1
```

Execute

```
$cd /opt/alfresco/acs-community-deployment/docker-compose
$ docker-compose up –d
```

4.4.1.3 Deploying Alfresco Content Services Community

With the custom Alfresco Repository and Share tiers ready, next step is to deploy Alfresco Content Services Community using Docker Compose. First step is to clone the https://github.com/Alfresco/acs-community-deployment repository, by running the following commands:

```
cd /opt/alfresco/
git clone https://github.com/Alfresco/acs-community-deployment.git
```

Update the Dockerfile (/opt/alfresco/acs-community-deployment/docker-compose/docker-compose.yml) entries for alfresco service with the one below:

```
#image: alfresco/alfresco-content-repository-community:6.2.0-ga
image:  foresight/custom-alfresco-repository:0.0.1
```

Update the Dockerfile entries for share service with the one below:

```
#image: alfresco/alfresco-share:6.2.2
image:  foresight/custom-share-repository:0.0.1
```

Finally execute:

```
$cd /opt/alfresco/acs-community-deployment/docker-compose
$docker-compose up –d
$docker-compose logs -f
```

If no errors are seen, this means that collaboration hub was successfully deployed. We can validate that by typing in a web-browser: https://foresight.eurodyn.com/share resulting in the next screen.

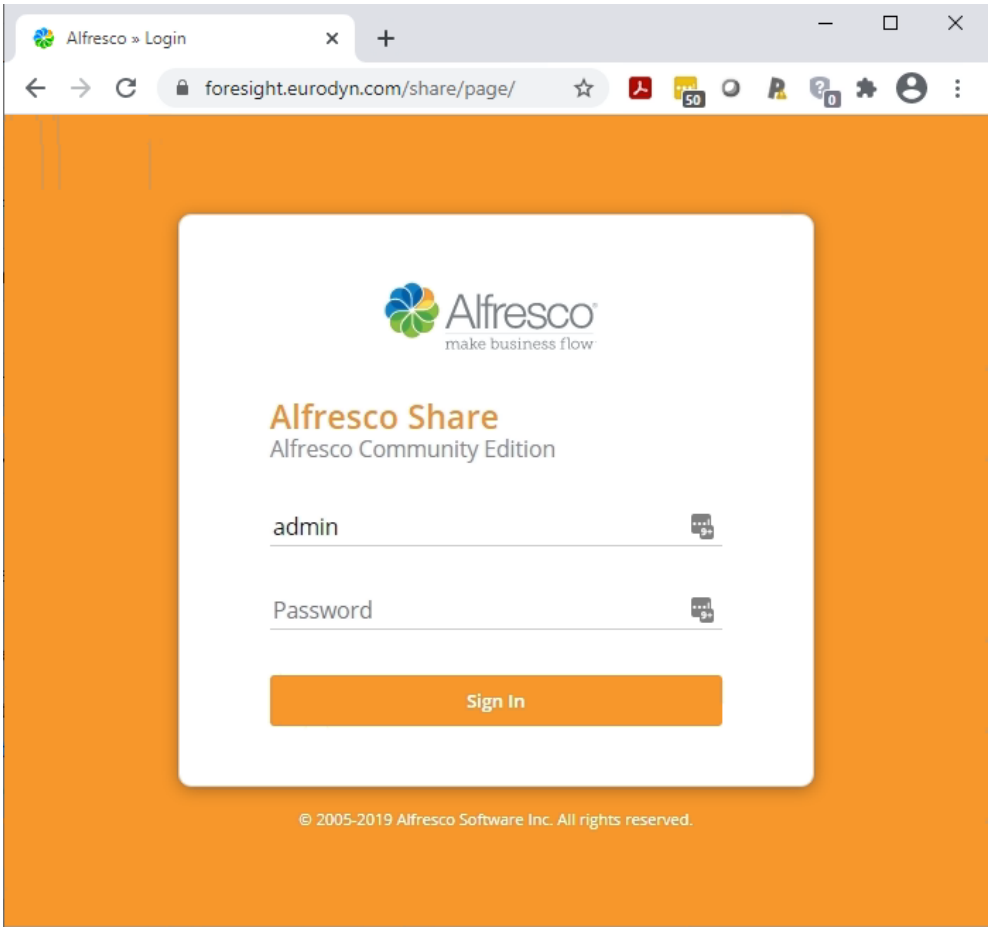


Figure 16. Collaboration Hub – Login

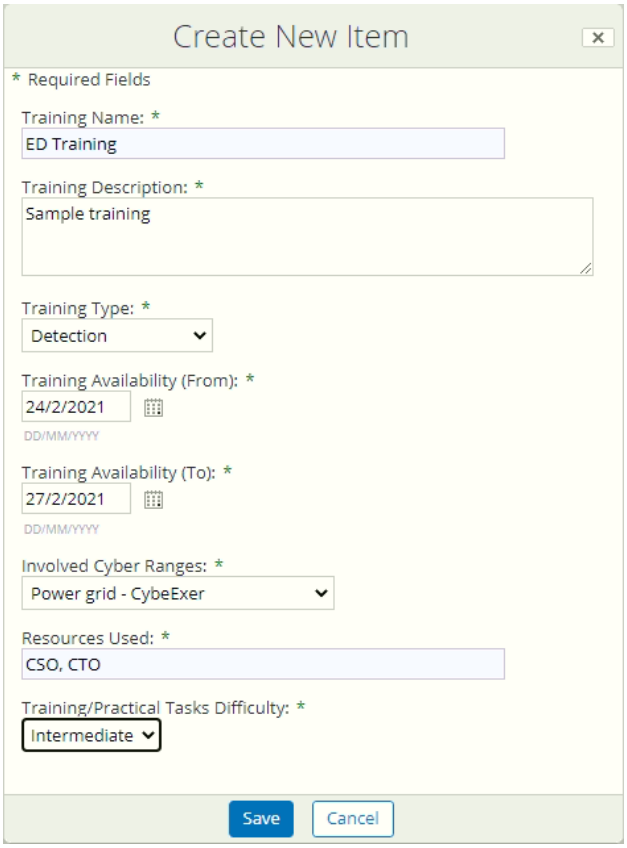
4.5 GitHub Repository

The installation manual, scripts and files of the FORESIGHT Cyber-Team Collaborative Tools, NGNIX reverse proxy and identity service (Keycloak) components are published in a private GitHub repository managed by ED: https://github.com/european-dynamics-rnd/FORESIGHT-CTC. They are available for authorised internal use of FORESIGHT partners.

5 Unit Testing

In this section the unit test cases for the developed components are defined, executed and their results presented. They include references to the system functional and non-functional requirements as defined in D2.2.

Test Case ID	CTC-01	Component	Collaboration Hub
Description	Maintain information related to the Cyber Ranges (CRs) capabilities, the resources used/available, the available scenarios/systems including their configuration, deployment and availability, the scenarios narrative and objective, the training difficulty, the rating specifications and methods to check completion.		
Req ID	Req-082	Priority	High
Prepared by	ED	Tested by	ED
Pre-condition(s)	Collaboration Hub is up and running		
Test steps			
1	Login to collaboration hub and navigate to the “FORESIGHT Collaboration” area		
2	Click the “**Data lists**” menu item on the **banner**		
3	Click on the “**Scenarios and Availability**” list entry in the lists explorer panel		
4	Click on the “**new item**” button and provide the training scenario details		

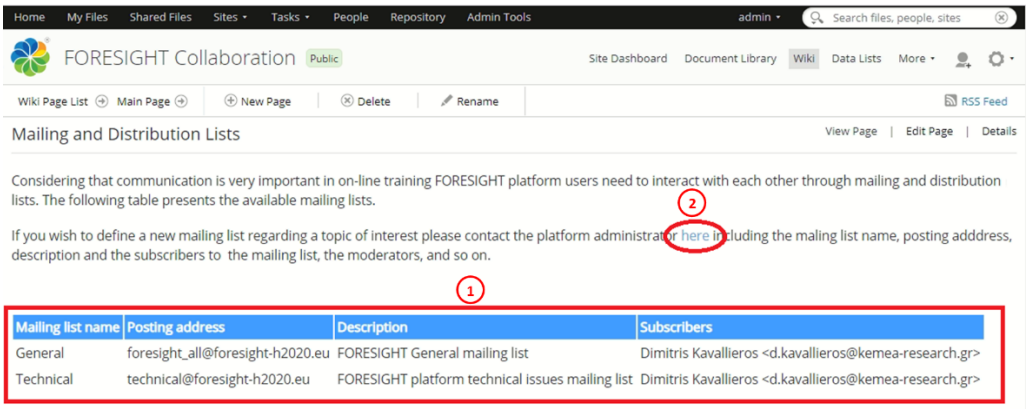
Input data	
Result	Newly created CR information related to their capabilities is uploaded Training Name | Training Description | Training Type | Training Availability (From) | Training Availability (To) | Involved Cyber Ranges | Resources Used | Training/Practical Tasks Difficulty AIA Training | AIA Training | Forensic Analysis | Wed 17 Feb 2021 | Thu 18 Feb 2021 | Aviation - Airbus Cybersecurity SAS | CSO, CTO | Intermediate ED Training | Sample training | Detection | Wed 24 Feb 2021 | Sat 27 Feb 2021 | Power grid - CybeExer | CSO, CTO | Intermediate
Test Case Result	Achieved

Test Case ID	CTC-02	Component	Collaboration Hub
Description	Present information collected from the original access interfaces (CRs) through a dedicated visualisation.		
Req ID	Req-082	Priority	Medium
Prepared by	ED	Tested by	ED
Pre-condition(s)	Successfully execute CTC_01		
Test steps			
1	Login to collaboration hub and navigate to the “FORESIGHT Collaboration” area		
2	Click the “**Data lists**” menu item on the banner		

3	Click on the "**Scenarios and Availability**" list entry in the lists explorer panel
Input data	
Result	User is able to see all information uploaded from CR admins related to their capabilities.
Test Case Result	Achieved

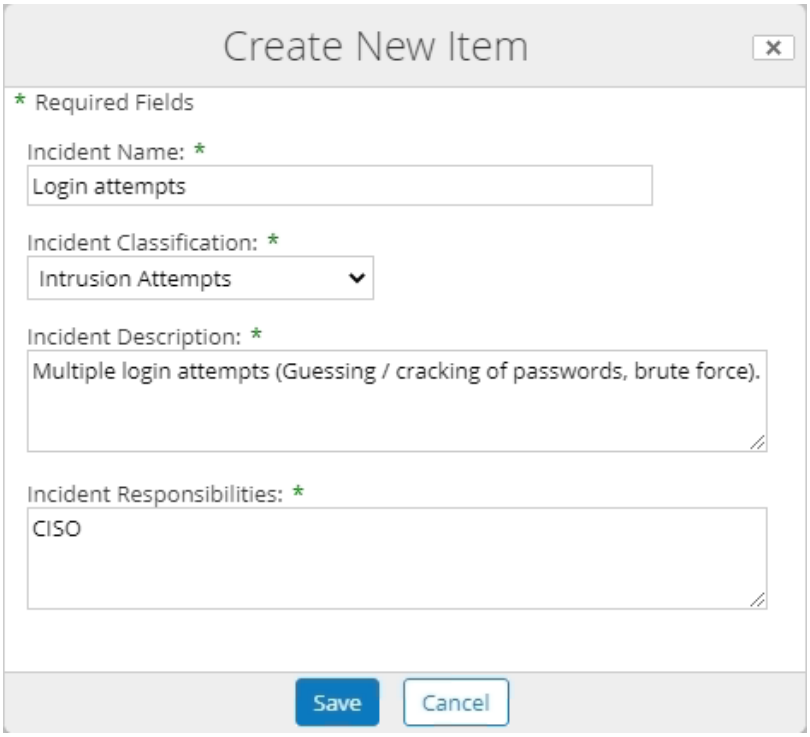
Test Case ID	CTC-03	Component	Communication Hub
Description	Allow FORESIGHT platform users to communicate and collaborate with each other via chat messages		
Req ID	Req-106, Req-171	Priority	High
Prepared by	ED	Tested by	ED
Pre-condition(s)	Communication Hub is up and running		
Test steps			
1	Login to Communication Hub		
2	Navigate to the channel (public/private) or group or the FORESIGHT member we want to send a message		
3	Type in the **message box** our message and click the Enter or the Send Button		
Input data			
Result	The message is posted to the FORESIGHT channel (public/private) or the FORESIGHT member		
Test Case Result	Achieved		

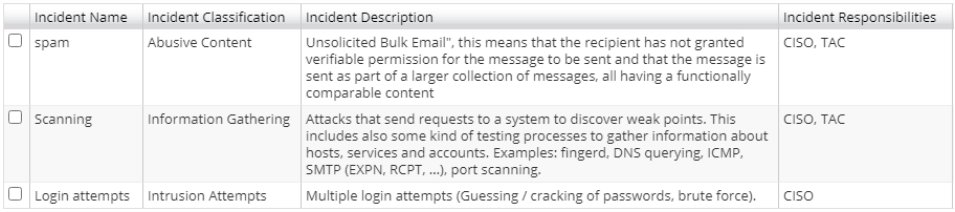
Test Case ID	CTC-04	Component	Collaboration Hub
Description	Maintain mailing and distribution lists and enable FORESIGHT platform users to be informed on the available ones, including their description and the list of FORESIGHT platform users ("subscribers") receiving mail, as well as request for new mailing lists regarding topics of interest		
Req ID	Req-106	Priority	Medium
Prepared by	ED	Tested by	ED
Pre-condition(s)	Collaboration Hub is up and running		
Test steps			

1	Login to collaboration hub and navigate to the “FORESIGHT Collaboration” area
2	While on the **personal dashboard** access the “Mailing and Distribution Lists” **wiki** page
Input data	
Result	User is able to see available mailing lists (1), including their description and the list of FORESIGHT platform users ("subscribers") receiving mail. User can request for new mailing lists regarding topics of interest (2). 
Test Case Result	Achieved

Test Case ID	CTC-05	Component	Collaboration Hub
Description	Allow FORESIGHT platform users to search for a particular user		
Req ID	Req-106	Priority	High
Prepared by	ED	Tested by	ED
Pre-condition(s)	Collaboration Hub is up and running		
Test steps			
1	Login to collaboration hub and navigate to the “FORESIGHT Collaboration” area		
2	While on the **personal dashboard**, access the “People Finder” page		
3	Type in the **search box** the full or partial name of the desired user and click the “**Search**” button.		
4	Type in the **search box** the profile property that we are searching for, for example, "location:London", "jobtitle:Manager" and click the “**Search**” button.		

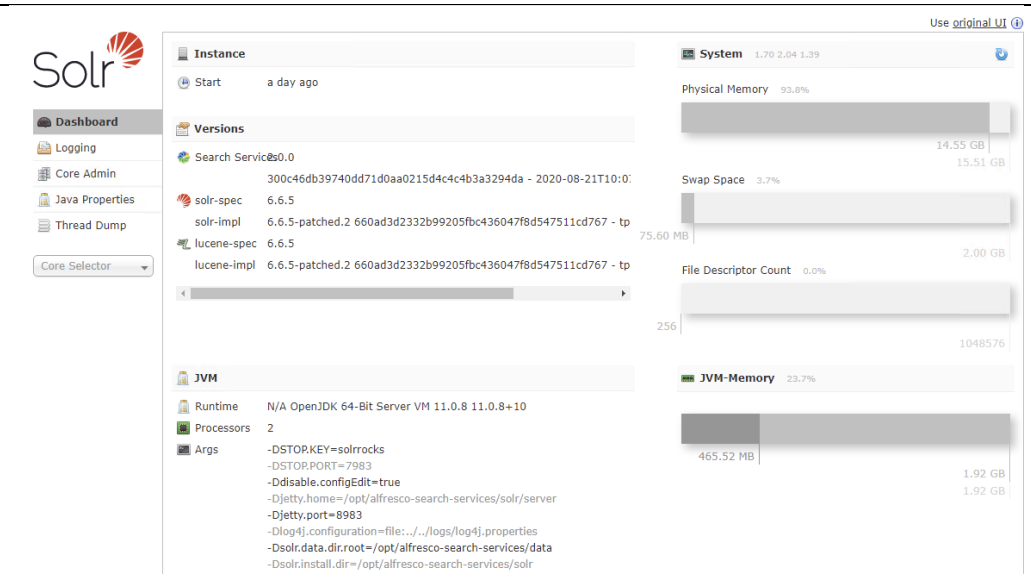
Input data	
Result	The results list appears beneath the search box and displays all users matching the search criteria provided. From within this list, a user can click a user name to display that user's profile
Test Case Result	Achieved

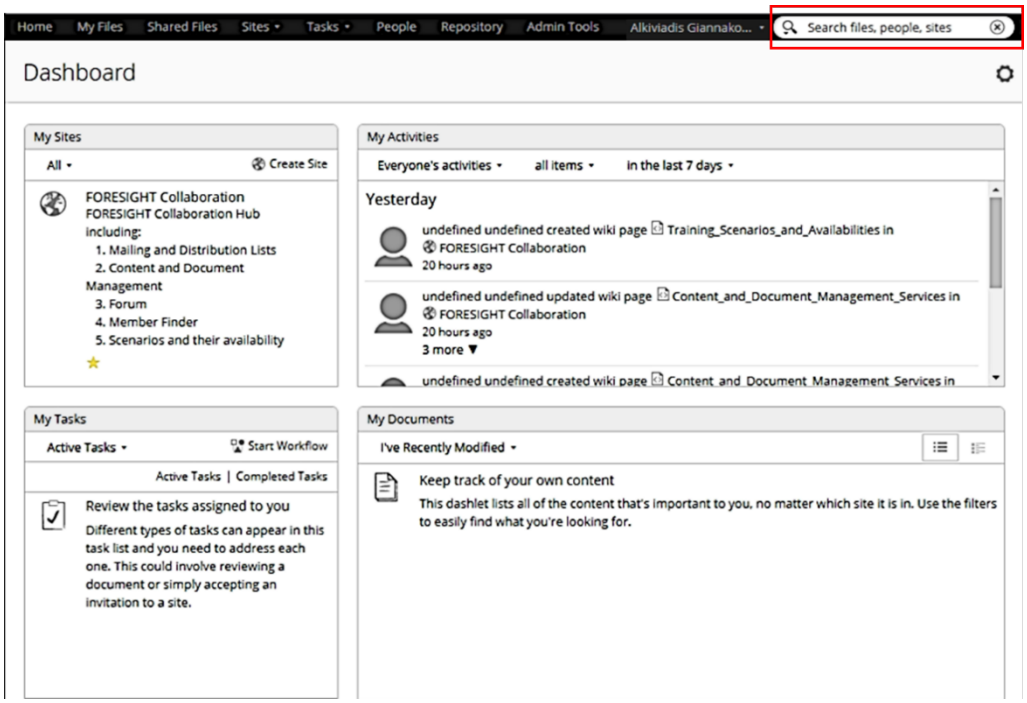
Test Case ID	CTC-06	Component	Collaboration Hub
Description	Maintain an incident taxonomy list and enable FORESIGHT platform users to retrieve the responsibilities for a certain incident.		
Req ID	Req-106, Req-203	Priority	High
Prepared by	ED	Tested by	ED
Pre-condition(s)	Collaboration Hub is up and running		
Test steps			
1	Login to collaboration hub and navigate to the "FORESIGHT Collaboration" area		
2	Click the "**Data lists**" menu item on the **banner**		
3	Click on the "**Incident Responsibilities**" list entry in the lists explorer panel		
4	Click on the "**new item**" button and provide the responsibilities for a certain incident, including their classification, description and list of roles responsible for incident response 		

Input data	
Result	Newly created incident responsibility item is created 
Test Case Result	Achieved

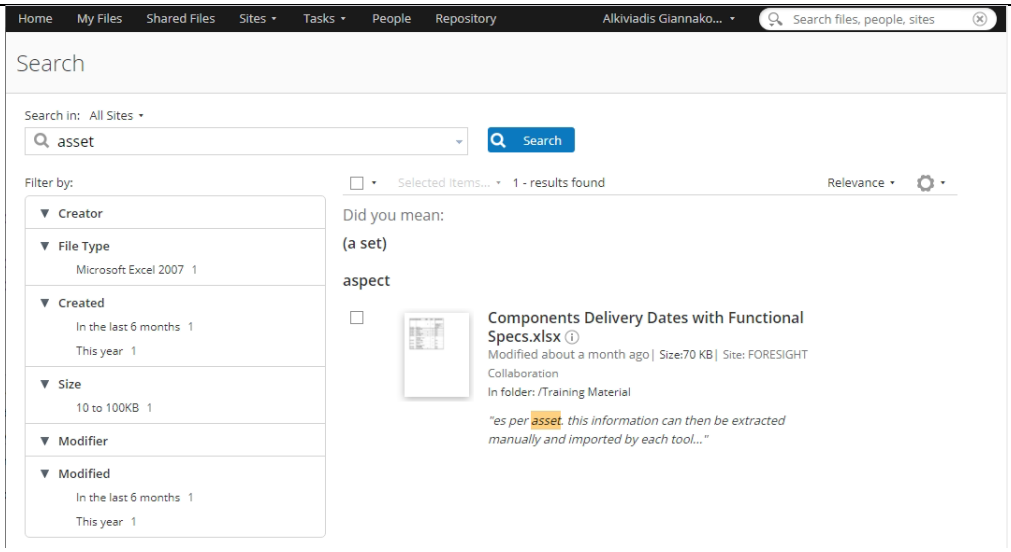
Incident Name	Incident Classification	Incident Description	Incident Responsibilities
spam	Abusive Content	Unsolicited Bulk Email", this means that the recipient has not granted verifiable permission for the message to be sent and that the message is sent as part of a larger collection of messages, all having a functionally comparable content	CISO, TAC
Scanning	Information Gathering	Attacks that send requests to a system to discover weak points. This includes also some kind of testing processes to gather information about hosts, services and accounts. Examples: fingerd, DNS querying, ICMP, SMTP (EXPN, RCPT, ...), port scanning.	CISO, TAC
Login attempts	Intrusion Attempts	Multiple login attempts (Guessing / cracking of passwords, brute force).	CISO

Test Case ID	CTC-07	Component	Collaboration Hub
Description	Support full text search properties		
Req ID	Req-106	Priority	High
Prepared by	ED	Tested by	ED
Pre-condition(s)	Collaboration Hub is up and running		
Test steps			
1	Integrate with the Solr 6 Enterprise search platform for searching within the content repository.		
2	Login to collaboration hub		
3	While on the **personal dashboard**, navigate to the “Search Input“ and		
4	Type in the **search box** the text that we are searching (for example “asset”) and hit enter.		
Input data			
Result	Navigate to http://foresight-server.eurodyn.com:8083/solr we should be able to see the Solr6 platform up and running.		

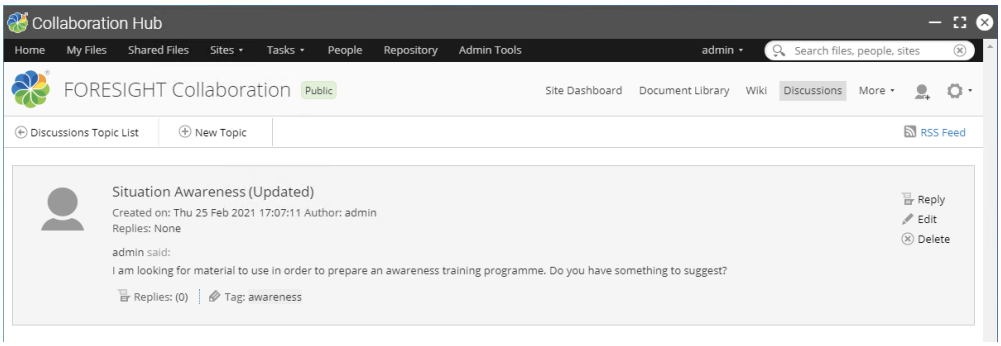
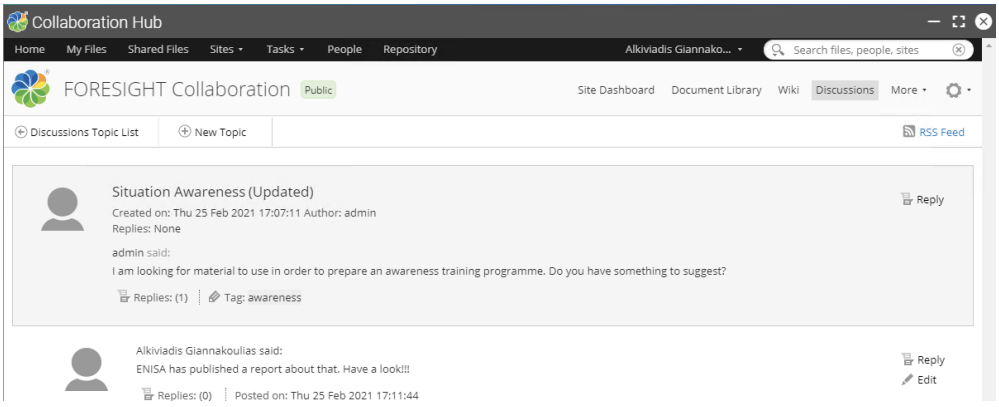




List of items matching search criteria appear, as illustrated below:

	
Test Case Result	Achieved

Test Case ID	CTC-08	Component	Collaboration Hub
Description	Maintain content management and document services and enable FORESIGHT platform users to access, create, upload and edit content directly within the system		
Req ID	Req-106	Priority	High
Prepared by	ED	Tested by	ED
Pre-condition(s)	Collaboration Hub is up and running		
Test steps			
1	Login to collaboration hub and navigate to the “FORESIGHT Collaboration” area		
2	While on the **personal dashboard**, access the “FORESIGHT Collaboration” site and its dashboard by clicking on it		
3	User access the **Document Library**, to store and manage content, such as documents, images, and videos.		
Input data	Users uploads content to share and work on with others.		
Result	Users can view and work on content, depending upon their permission settings.		
Test Case Result	Achieved		

Test Case ID	CTC-09	Component	Collaboration Hub
Description	Allow FORESIGHT platform users to communicate and collaborate asynchronously with each other via forums and discussions		
Req ID	Req-171	Priority	High
Prepared by	ED	Tested by	ED
Pre-condition(s)	Collaboration Hub is up and running		
Test steps			
1	Login to collaboration hub and navigate to the “FORESIGHT Collaboration” area		
2	While on the **personal dashboard**, access the “Discussions” page on the banner, resulting in the “Discussions” page to appear		
Input data	1. Users create a new topic for discussion by clicking the “new Topic” button and provides the topic title, text and tags. 2. Other users reply to the topic to take part in the discussion		
Result	1. Discussion topic is created and presented in the “**Topic list**”  2. Discussion topic is updated with responses 		
Test Case Result	Achieved		

6 Conclusions

This deliverable presented the first version of the cyber-team collaborative (CTC) tools architecture, offering users the ability to interact with each other through: a) a communication hub and b) a collaboration hub.

The development of the FORESIGHT CTC tools was based on the result of the analysis of the FORESIGHT CTC requirements provided in this deliverable (Section 2).

The communication hub is based on **RocketChat**, an open source communication hub and uses a MongoDB replica set to improve performance via Meteor Oplog tailing, while also providing High Availability (HA). The collaboration hub is based on **Alfresco Community Edition**, an open source Enterprise Content Management software, and uses a relational database (PostgreSQL) for storing content metadata and a file system to store the actual content.

Additionally, the FORESIGHT Identity Service, implemented on top of **JBoss Keycloak** was presented, centralizing the authentication function and authenticating users while freeing the CTC tools from dealing with login forms, authenticating users and storing users.

Furthermore, the deliverable explained the details of the software prototypes developed for each of the components. The prototype description included installation and configuration details, and the references to the artefact repositories.

Finally, the deliverable also presented the results of the FORESIGHT unit tests performed over the CTC tools namely the communication hub and the collaboration hub, thus showing compliance to the requirements defined in WP2.

References

[1] Wikipedia, User interface specification

[2] Bridging the gap, Laura Brandenburg, How to Create a User Interface Specification

[3] Wikipedia, Role-based access control

[4] Keycloak, open source Identity and Access Management solution, http://www.keycloak.org/

[5] Reference Incident Classification Taxonomy, ENISA 2018, https://www.enisa.europa.eu/publications/reference-incident-classification-taxonomy

[6] RocketChat Press, https://rocket.chat/press/

[7] Alfresco Community Edition, https://www.alfresco.com/ecm-software/alfresco-community-editions#:~:text=Alfresco%20Community%20Edition,Business%20Intelligence%2C%20Analytics%2C%20and%20Insights

[8] Alfresco Documentation: Alfresco Community Edition (201911 GA) architecture

[9] Alfresco Documentation: Alfresco repository concepts

[10] Getting Started with Alfresco Identity Service EA (Keycloak)

[11] Alfresco Documentation: API Guide (https://docs.alfresco.com/community/concepts/dev-api-intro.html)

[12] RocketChat REST API, https://docs.rocket.chat/api/rest-api

[13] Alfresco Documentation: Managing Alfresco Search Services

[14] RocketChat - UVicDSA18

[15] RocketChat, Authentication, Keycloak, https://docs.rocket.chat/guides/administrator-guides/authentication/open-id-connect/keycloak

[16] Stav, E., S. Walderhaug, and U. Johansen, ARCADE - An Open Architectural Description Framework. December 2013, SINTEF ICT. Available at: http://www.arcade-framework.org/wp-content/uploads/2013/12/ARCADE-Handbook.pdf

