

APRIL 12, 2022



iSHARE

ISHARE SATELLITE DEPLOYMENT GUIDE

VM-DOCKER BASED MODEL

ISHARE FOUNDATION

Table of Contents

1. Pre-requisites	2
2. Hardware Requirements.....	2
2.1. Minimum.....	2
2.2. Recommendation.....	2
3. Network Requirements.....	2
4. Getting Started.....	3
4.1. Configure Passwords for the services	4
4.2. Installing Hyperledger Fabric node.....	7
4.3. Register your node.....	10
4.4. Join the network	11
4.5. Deploy the UI and middleware applications	14
5. Initial user setup	17
6. Commands for managing Docker services.....	23

1. Pre-requisites

The proposed model requires virtual machines provisioned on cloud providers and/or on-prem with prescribed operating systems and following software's installed in them. Make sure that you have then before beginning

Note: the scripts and components are tested in versions mentioned in the brackets, usually it should work well in higher version as well

1. Ubuntu Linux (20.04.4 LTS (Focal Fossa))
2. Docker (v20.10.14)
3. Docker-compose (version 1.24.0)
4. Create/Update rights to your DNS service to manage satellite URLs in DNS
5. SSL certificates of your domain for applications (applications by default use HTTPS so having certificates on hand is required. You may also use free Letsencrypt certificates, please refer to its website on how to get them)

2. Hardware Requirements

2.1. Minimum

Virtual Machine with 2 CPU's and 8 GB Memory

2.2. Recommendation

Virtual Machine with 4 CPU's and 16 GB Memory

3. Network Requirements

Following ports are used by following applications, so make sure that you configure your firewall and network settings to allow access via these ports

- 443/TCP,80/TCP for Application middleware and UI
- 7051/TCP and 8051/TCP for HLF Peers
- 8443/TCP for keyCloak instance (user management backend)
- 8081/TCP for explorer instance (hyperledger explorer) – open to outside only if require access from outside your own network, maybe necessary when hosted on cloud

4. Getting Started

iSHARE satellite is based on Hyperledger fabric where participants registered via satellite are directly registered on the shared ledger and participant is trusted across the network. The iSHARE satellite is composed of following sub-components

- Hyperledger fabric node
- Application UI
- Keycloak (user management backend)
- Satellite Middleware (APIs and other relevant functions)

To install iSHARE satellite and configure it to run follow the steps:

1. Download and prepare your scripts/passwords
2. Install Hyperledger fabric node
3. Register your node
4. Join the network
5. Deploy the UI and middleware applications
 - a. Deploy keycloak
 - b. Deploy middleware
 - c. Deploy UI
6. Setup satellite admin and login to UI

4.1. Configure Passwords for the services

Download the scripts from GitHub

```
git clone https://github.com/iSHAREScheme/iSHARESatellite.git  
cd iSHARESatellite
```

If you want to change the default passwords please follow these steps, else skip this chapter.

Following are the services which uses username and password as credentials for authentication:

- Postgres DB for Application
- Explorer DB for Explorer
- Postgres DB for Keycloak
-

Steps to configure password for changing HLF Explorer's DB:

```
cd iSHARESatellite/templates
```

Open explorer-docker-compose-template.yaml in the text editor.

Under *services* section, change the password for explorerdb by referring below snippet under environment:

```
explorerdb:  
  image: hyperledger/explorer-db:latest  
  container_name: explorerdb  
  hostname: explorerdb  
  environment:  
    - DATABASE_DATABASE=fabricexplorer  
    - DATABASE_USERNAME=hppoc  
    - DATABASE_PASSWORD=password
```

Same Password has be configured for explorer service under environment, refer below snippet

```
explorer:  
  image: hyperledger/explorer:latest  
  container_name: explorer  
  hostname: explorer  
  environment:  
    - DATABASE_HOST=explorerdb  
    - DATABASE_DATABASE=fabricexplorer  
    - DATABASE_USERNAME=hppoc  
    - DATABASE_PASSWD=password
```

Open app-mw-config-template.yaml in the text editor, configure password in explorer db connection string. Check below



explorerDb: postgresql://hppoc:password@explorerdb:5432/fabricexplorer?sslmode=disable

Steps to configure password for Application middleware postgres DB

```
cd iSHARESatellite/templates
```

Open docker-compose-mw-template.yaml in a text editor and look for below snippet :

```
app-postgres:
  image: postgres:9
  restart: always
  environment:
    POSTGRES_USER: admin
    POSTGRES_PASSWORD: adminpw
```

Change password for admin user under environment section and save.

Open hlf-mw-config-template.yaml file in a text editor and look for the below snippet and lower end of the file.

```
ishareConfig:
  middlewareConfig:
    postgres_connection_url: postgresql://admin:adminpw@app-
    postgres:5432/ishare?sslmode=disable
```

Change the password adminpw in connection string with password configured in docker-compose-mw.yaml.

Open app-mw-config-template.yaml file in a text editor and look for the below snippet.

```
ishareConfig:
  middlewareConfig:
    postgres_connection_url: postgresql://admin:adminpw@app-
    postgres:5432/ishare?sslmode=disable
```

Change the password adminpw in connection string with password configured in docker-compose-mw.yaml.

Steps to configure password for Application Keycloak service

```
cd iSHARESatellite/keycloak/postgres
```

Open init-db.sql in a text editor



```
CREATE USER keycloak WITH ENCRYPTED PASSWORD 'keycloak';
```

Change password from 'keycloak' to desired password. Password should be under single quotes.

```
cd iSHARESatellite/keycloak
```

Open keycloak-docker-compose.yaml in a text editor, look for below snippet keycloak:

```
image: jboss/keycloak:11.0.2
container_name: keycloak
ports:
  - 8443:8443
environment:
  - KEYCLOAK_USER=admin
  - KEYCLOAK_PASSWORD=admin
  - DB_VENDOR=postgres
  - DB_ADDR=keycloakpostgres
  - DB_DATABASE=keycloak
  - DB_USER=keycloak
  - DB_PASSWORD=keycloak
```

Change DB_PASSWORD value keycloak with desired password value

4.2. Installing Hyperledger Fabric node

Hyperledger Fabric

The fabric component for each satellite will contain :

1. 2 Peers
2. Shared Orderer
3. 2 CouchDB
4. 1 Fabric CA

These components have to be pulled as images ,installed and initialized as per the directions mentioned here.

Steps and sequence for creating HLF components:

Download the scripts from GitHub (if not done so already)

```
git clone https://github.com/iSHAREScheme/iSHARESatellite.git  
cd iSHARESatellite
```

To install and ensure all the necessary packages are available in the system, use the below script to verify(reboot the system on completion, if necessary).

```
bash prerequisites.sh
```

```
cd iSHARESatellite/scripts
```

Configure environment variables to initialize scripts. See env variables with example below:

Environment Variables	Description
ORG_NAME	satellite which is going to be a part of HLF network. It should be a word without special characters (only alphanumeric) and max 17 characters. ex: mysatellite
SUB_DOMAIN	Sub-domain reserved in DNS service for this satellite. No special characters. ex: uat.mydomain.com
ENVIRONMENT	Name of the infra environment like uat, test, prod. No special characters.



```
export ORG_NAME=newsatellite
export SUB_DOMAIN=test.example.com
export ENVIRONMENT=test
```

To generate HLF fabric ca certs, use the below command :

```
bash fabric-ca-cert.sh
```

Creating HLF Fabric CA instance:

```
bash fabric-ca.sh
```

Validation

Navigate to iSHARESatellite/hlf/<Environment>/<orgName>/fabric-ca directory.
Check the presence of the docker_data folder.

To ensure all the HLF Fabric is running, use below commands and check the status.

```
docker-compose -f iSHARESatellite/hlf/<Environment>/<orgName>/fabric-
ca/docker-compose-fabric-ca.yaml ps
```

Register and Enroll users and peers

```
bash registerAndEnroll.sh
```

Create HLF Peer instances, it will spun up instances of HLF Peers and CouchDB

```
bash peer.sh
```

Validation

Navigate to iSHARESatellite/hlf/<Environment>/<orgName>/peers directory.

Check the presence of the docker_data folder.

To ensure all the HLF Peer is running, use below commands to check the status.

```
docker-compose -f
iSHARESatellite/hlf/<Environment>/<orgName>/peers/docker-compose-hlf.yaml
ps
```

HLF Peer instances needs to be a part of iSHARE Foundation HLF network, for that HLF peers needs to be reachable over the internet. Previous script peer.sh creates two HLF peer instance with hostname peer<num>.<ORG_NAME>.<SUB_DOMAIN> and they listen at port 7051 and 8051 TCP. Make sure that necessary firewall settings are updated to allow access to these peers over internet . Map dns entries for Peers hostname with server IP address ex:

If ORG_NAME=newsatellite and SUB_DOMAIN=test.example.com

```
peer0.<orgname>.<subdomain> - peer0.newsatellite.test.example.com
```

```
peer1.<orgname>.<subdomain> - peer1.newsatellite.test.example.com
```



Full Record Name	Record Type	Value	TTL
peer0.<orgname>.<subdomain>	A	12.9.7.6.5	1min
peer1.<orgname>.<subdomain>	A	12.9.7.6.5	1min

The Hyperledger fabric node is deployed, now follow next chapter to register your node in the network.

4.3. Register your node

Share details with iSHARE foundation to register your node on iSHARE network

Now generate the organization definition file which iSHARE Foundation (HLF Channel Admin) would require in order to onboard your new satellite. Use the below command to create an org definition.

```
bash orgDefinition.sh
```

Find the <orgName>.json at hlf/<ENVIRONMENT>/<ORG_NAME> folder

Note: <orgName>.json file has to be shared securely. As it contains x509 certificates of new Satellite.

4.4. Join the network

Join the network with information received from iSHARE foundation. You should receive following information and files from iSHARE foundation

Files: Copy these files in the VM

- CA cert file of HLF ordering service (ORDERER_TLS_CA_CERT)
- Genesis.block file
- Channel tx file

Note: you need to copy Genesis.block and channel .tx file to the same folder which contains the iSHARESatellite folder

Values for following variables

- ORDERER_ADDRESS - one of the ordering services hostname and port
- CHANNEL_NAME - channel name in which a particular satellite is on boarded
- CHAINCODE_NAME - chaincode (smart contract) which is defined in chaincode definition committed
- CHAINCODE_VERSION - chaincode version defined in chaincode definition committed
- CHAINCODE_SEQUENCE - sequence number associated with chaincode
- CHAINCODE_POLICY - chaincode policy string used while committing chaincode definition in the HLF network

Once the above details is known, move the copy of ORDERER_TLS_CA_CERT file in the VM and follow below steps

Export these Environment variables:.

Environment Variables	Description
ORG_NAME	satellite which is going to be a part of HLF network. ex: mysatellite. You already set this value in chapter 4.2, make sure to use the same value here
SUB_DOMAIN	Sub-domain reserved in DNS service for this satellite.ex: uat.mydomain.com. You already set this value in chapter 4.2, make sure to use the same value here
PEER_COUNT	Number of HLF Peer nodes in a satellite, default = 2
ENVIRONMENT	Name of the infra environment like



	uat,test,prod. You already set this value in chapter 4.2, make sure to use the same value here
ORDERER_TLS_CA_CERT	Path to ca cert file of ordering service which you received from iSHARE foundation and copied to your VM
ORDERER_ADDRESS	Ordering service hostname with port ex: orderer1.example.aks.io:443
CHANNEL_NAME	Name of the channel in which new satellite in on boarded
ANCHOR_PEER_HOSTNAME	HLF peer node hostname of a new satellite ex: peer0.example.com. The peer name from chapter 4.2
ANCHOR_PEER_PORT_NUMBER	HLF peer node listening port ex: 7051, corresponding port of the peer set in ANCHOR_PEER_HOSTNAME
CHAINCODE_NAME	Chaincode (smartcontract) name defined in chaincode definition committed
CHAINCODE_VERSION	Chaincode version defined in chaincode definition committed
CHAINCODE_SEQUENCE	Sequence number defined in chaincode definition committed
CHAINCODE_POLICY	Chaincode policy defined in chaincode definition committed
PEER_ADMIN_MSP_DIR	Admin user msp directory of satellite ex: app/< ENVIRONMENT>/crypto/peerOrganization/<subdomain>/users/Admin@subdomain/msp

```
export ORG_NAME=<orgname>
export SUB_DOMAIN=<sub-domain>
export PEER_COUNT=2
export ORDERER_COUNT=0
export ENVIRONMENT=test
export ORDERER_TLS_CA_CERT=<path-to-orderer-ca-cert>
export ORDERER_ADDRESS=<orderer-hostname-with-port>
export CHANNEL_NAME=<channelname>
export ANCHOR_PEER_HOSTNAME=<hostname-of-one-the-hlf-peer>
```



```
export ANCHOR_PEER_PORT_NUMBER=<port-number-of-one-the-hlf-peer>
export CHAINCODE_NAME=<chaincode-name>
export CHAINCODE_SEQUENCE=<chaincode-version>
export CHAINCODE_VERSION=<chaincode-version>
export CHAINCODE_POLICY=<chaincode-policy>
export PEER_ADMIN_MSP_DIR=/<path>/app/<orgname>/users/Admin/msp
```

Join the HLF channel using below script:

```
cd iSHARESatellite/scripts
bash joinchannel.sh
```

Anchor peer for new satellite:

```
bash anchorPeer.sh
```

Create HLF explorer instance:

```
bash explorer.sh
```

Install chaincode in new satellite peer

```
bash installChaincode.sh
```

Approve the chaincode for new satellite peer

```
bash approveChaincode.sh
```

Create chaincode instance

```
bash chaincode.sh
```

Your node is now on network!

4.5. Deploy the UI and middleware applications

Note: Steps to configure Private key (of the eIDAS) certificate in production environment could differ to accommodate your organizations policies.
For test environments currently we configure the private keys in VM itself which is not very secure. But since we issue you test eIDAS certificates it is usually no issue.
Please contact us if you wish to configure private keys differently.

Export these environment variables to initialize scripts:

Environment Variables	Description
ORG_NAME	satellite which is going to be a part of HLF network. ex: mysatellite. You already set this value in chapter 4.2, make sure to use the same value here
SUB_DOMAIN	Sub-domain reserved in DNS service for this satellite.ex: uat.mydomain.com. You already set this value in chapter 4.2, make sure to use the same value here
CHANNEL_NAME	Name of the channel in which new satellite in on boarded. Same value as used in previous chapter
CHAINCODE_NAME	Chaincode (smartcontract) name defined in chaincode definition committed. Same value as used in previous chapter
UIHostName	DNS name for Application UI ex: mysatellite.example.com. Base URL of your satellite application
MiddlewareHostName	DNS name for Application middleware ex: mysatellite-mw.example.com. Base URL of your satellite APIs
KeycloakHostName	DNS name for Application keycloak service ex: mysatellite-keycloak.example.com. Base URL of your keycloak for user administration. Internal use only.

```
export ORG_NAME=<myorg>
export SUB_DOMAIN=<test.example.com>
export KeycloakHostName=<myorg-keycloak-test.example.com>
export CHANNEL_NAME=<mychannel>
export CHAINCODE_NAME=<ccname>
```



```
export UIHostName=<myorg-test.example.com>
export MiddlewareHostName=<myorg-mw-test.example.com>
export ENVIRONMENT=<test>
```

Configure HTTPS (SSL/TLS):

To configure SSL (HTTPS) you would need the certificate and private key file from your CA. Copy SSL certs for the application inside “ssl” directory with a cert file named tls.crt and a private key file named tls.key.

Copy the public certificate (pem file) to {base_dir}/ssl/tls.crt
Copy the private key (key file) to {base_dir}/ssl/tls.key

Note: ssl certs should be a wild card certificate of your domain name like *.example.com.

Configure Signing certificate (e.g. your eIDAS certificate):

Copy/Move RSA public cert and private key (e.g. your eIDAS certificate) in jwt-rsa folder. Jwt-rsa folder should contain following files with given names below:

- jwtRSA256-private.pem – RSA private key file for jwt signing
- jwtRSA256-public.pem - RSA public cert for jwt signing

Configure environment and deploy applications:

Copy “genesis.block” and “channel.tx” files in middleware folder. These files were shared by iSHARE foundation to you earlier

Create Keycloak Instance using below command, It listens at port 8443

```
cd iSHARESatellite/scripts
bash keycloak.sh
```

Create middleware instances using below command

```
bash middleware.sh
```

Create UI and Nginx instances:

```
bash deployUI.sh
```

Configure DNS for your applications:

Map dns entries for application services with server IP address for these environment variables:

UIHostName (ex: myorg-test.example.com A record 123.0.0.23, example.com is the domain name for the applications UI and 123.0.0.23 is the server's public IP)

MiddlewareHostName (ex: myorg-mw-test.example.com A record 123.0.0.23, example.com is the domain name for the applications middleware and 123.0.0.23 is the server's public IP)



KeycloakHostName (ex: myorg-keycloak-test.example.com A record
123.0.0.23, example.com is the domain name for the applications keycloak
and 123.0.0.23 is the server's public IP)

5. Initial user setup

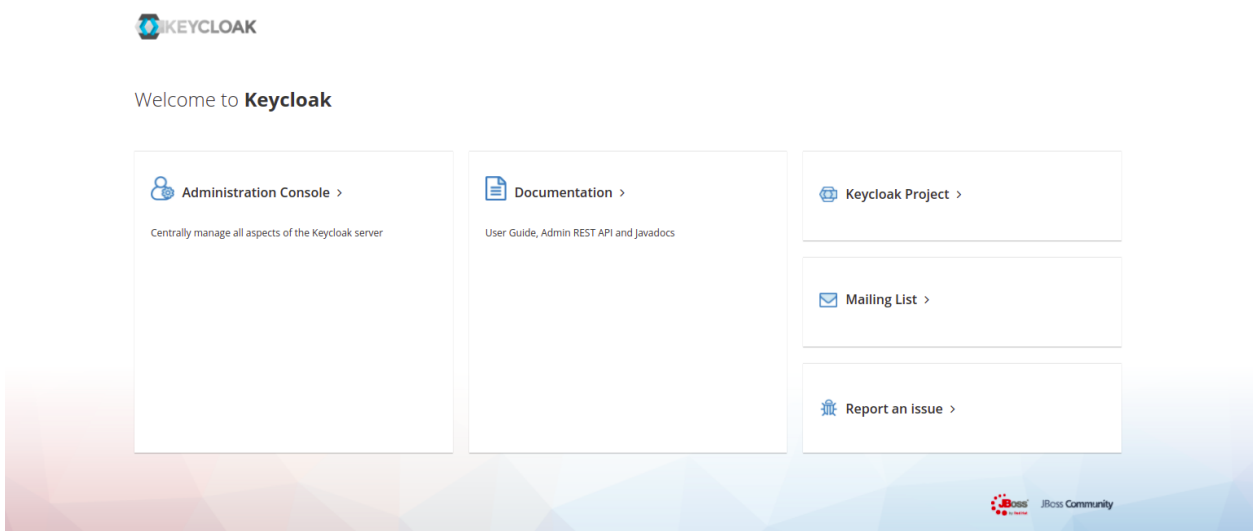
Now that you have installed satellite, to access it first setup an Satellite admin user with following procedure

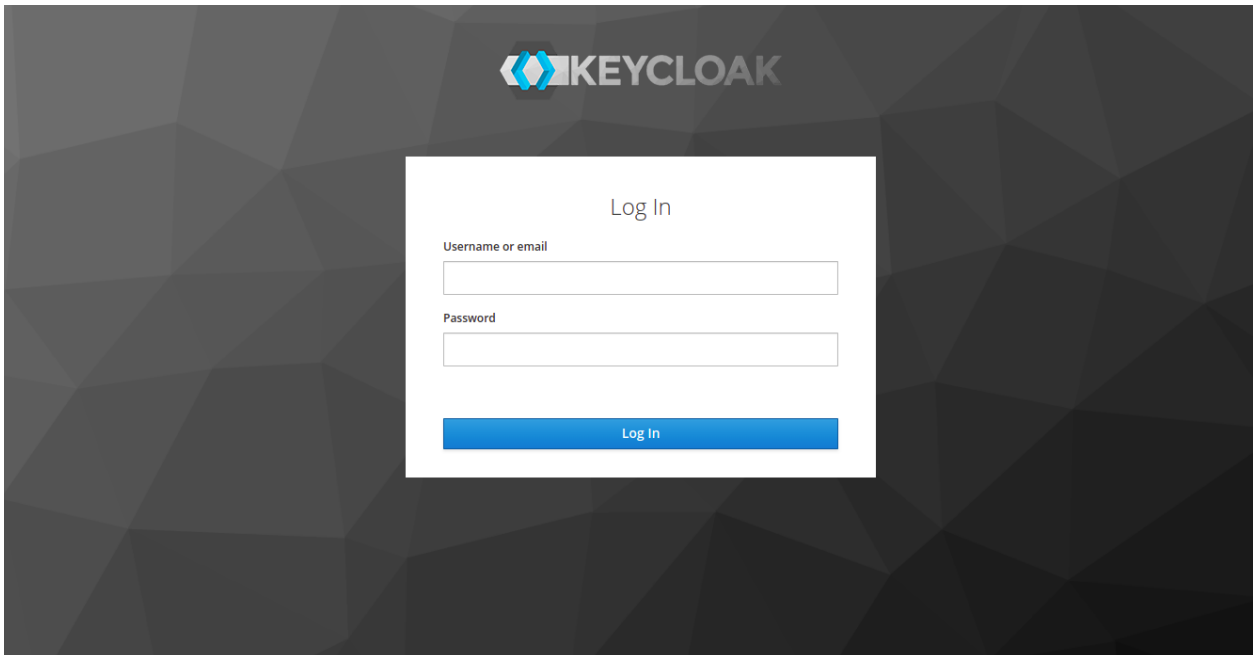
NOTE: Once you have setup initial user, you can add your colleagues via satellite UI which is very straight forward, so you do not need to follow same procedure for adding other users. Also, make sure to secure keycloak environment as per your organization policy so only limited users can login into keycloak admin console. Additionally, you can set more stricter password and user policies via keycloak admin console.

Keycloak URL is accessible over the browser example - <https://satellite.example.com:8443/auth>.

Steps for Keycloak user creation

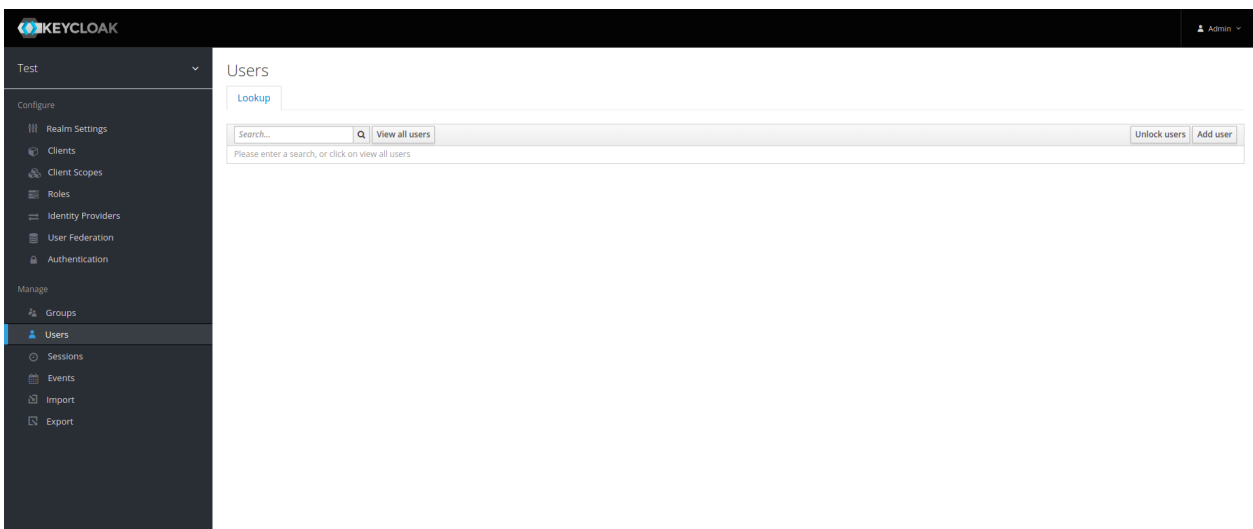
1. Click on administrator console and login with seeded admin user refer to below screenshots.



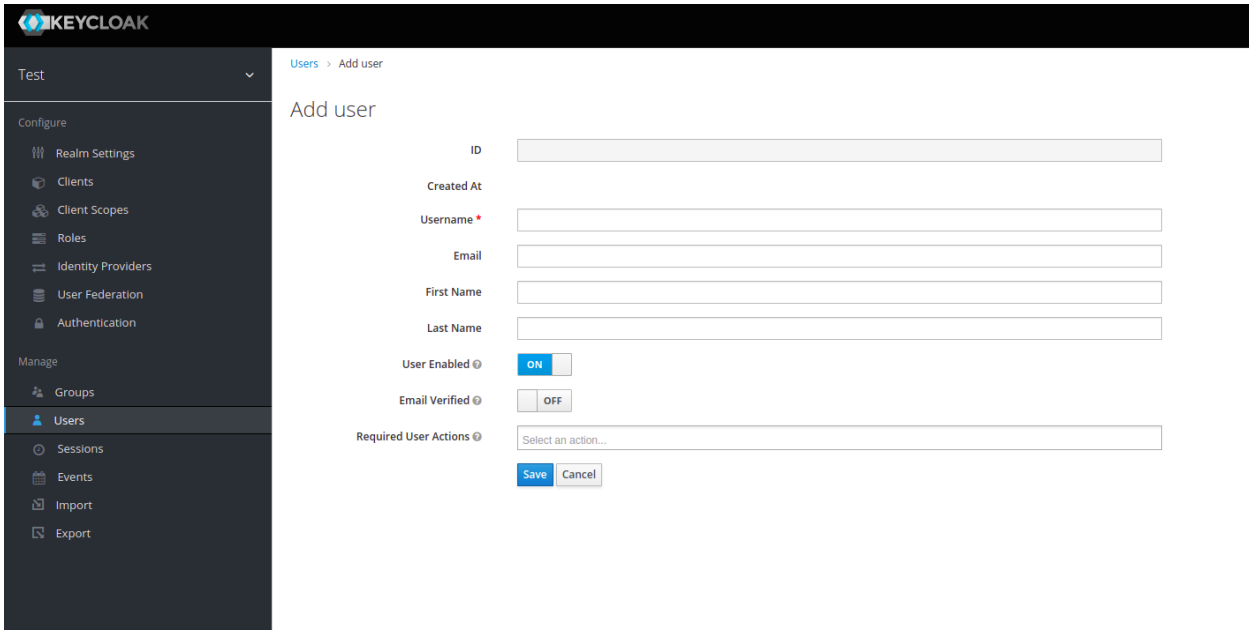


Note: username and password for admin could be found in keycloak-docker-compose.yaml inside keycloak directory for project folder.

2. Once logged in successfully, click on users in left menu



3. Click on Add User button on right side of it



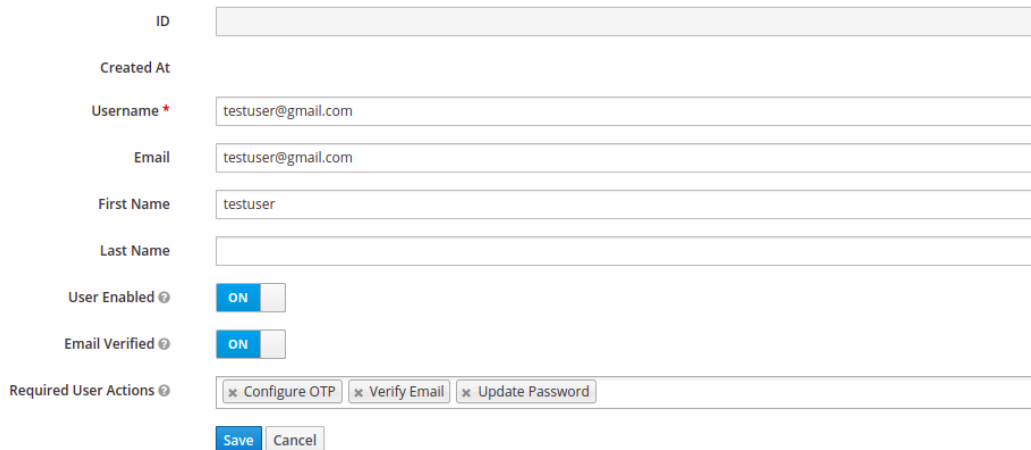
The image shows the Keycloak Admin Console 'Add user' form. The left sidebar contains navigation links for 'Test', 'Configure' (Realm Settings, Clients, Client Scopes, Roles, Identity Providers, User Federation, Authentication), and 'Manage' (Groups, Users, Sessions, Events, Import, Export). The 'Add user' form includes fields for ID, Created At, Username, Email, First Name, and Last Name. It also has toggle switches for 'User Enabled' (ON) and 'Email Verified' (OFF). A 'Required User Actions' dropdown is set to 'Select an action...'. At the bottom are 'Save' and 'Cancel' buttons.

4. Fill all the details in Add user form, refer below image and click save.

Note : username and email should be email id.

Users > Add user

Add user



The image shows the 'Add user' form with the following values filled in: ID (empty), Created At (empty), Username (testuser@gmail.com), Email (testuser@gmail.com), First Name (testuser), Last Name (empty), User Enabled (ON), Email Verified (ON), and Required User Actions (Configure OTP, Verify Email, Update Password). 'Save' and 'Cancel' buttons are at the bottom.

5. Click on Attribute tab, add the following attributes with values and save.

Note: Attributes are mandatory.

Attribute Name	Attribute Value
partyId	ID of the participant that this user belongs. Usually, the satellite ID
partyName	Name of the participant corresponding to the partyId



The image shows the 'Attributes' tab in the Keycloak Admin Console. It has tabs for 'Details', 'Attributes' (selected), 'Credentials', 'Role Mappings', 'Groups', 'Consents', and 'Sessions'. The table below lists attributes:

Key	Value	Actions
partyId	testparty001	Delete
partyName	testparty	Delete
		Add

'Save' and 'Cancel' buttons are at the bottom left.



- Click on credentials tab, enter the password for the user created as follows and set the password

Details Attributes **Credentials** Role Mappings Groups Consents Sessions

Manage Credentials

Position	Type	User Label	Data	Actions
----------	------	------------	------	---------

Set Password

Password:

Password Confirmation:

Temporary: ☒

- Click on Role Mapping tab, find Client Roles dropdown and select frontend

Details Attributes Credentials **Role Mappings** Groups Consents Sessions

Realm Roles

Available Roles: ADMIN, VISITOR

Assigned Roles: offline_access, uma_authorization

Effective Roles: offline_access, uma_authorization

Client Roles

Select a client... frontend

- Under Available Roles, select satelliteAdmin and click on Add Selected button

Details Attributes Credentials **Role Mappings** Groups Consents Sessions

Realm Roles

Available Roles: ADMIN, VISITOR

Assigned Roles: offline_access, uma_authorization

Effective Roles: offline_access, uma_authorization

Client Roles

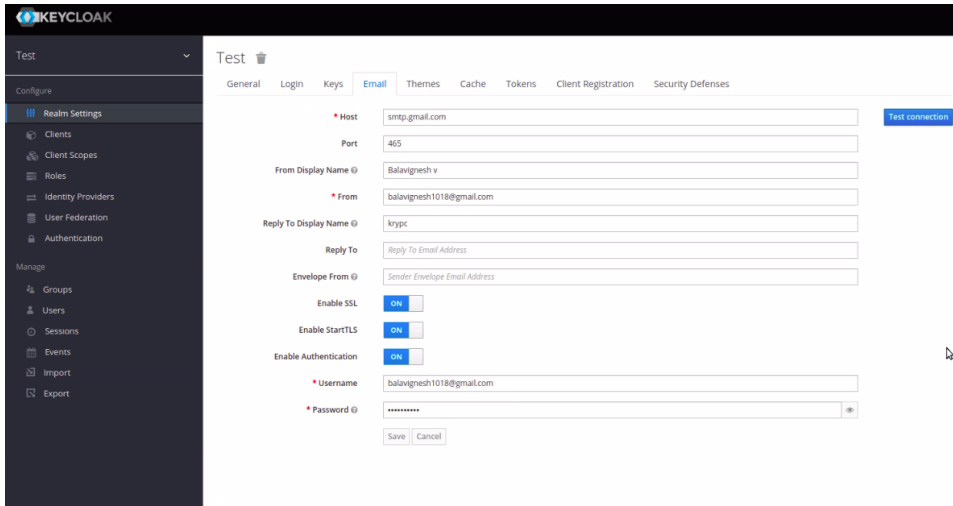
frontend

Available Roles: PartyAdmin, **SatelliteAdmin**, User

User is created, and you can proceed to use satellite UI with newly created User and start registering participants.

Steps for Email Notification under keycloak administrator login:

- Login as administrator user
- Click on Realm Settings under left menu bar

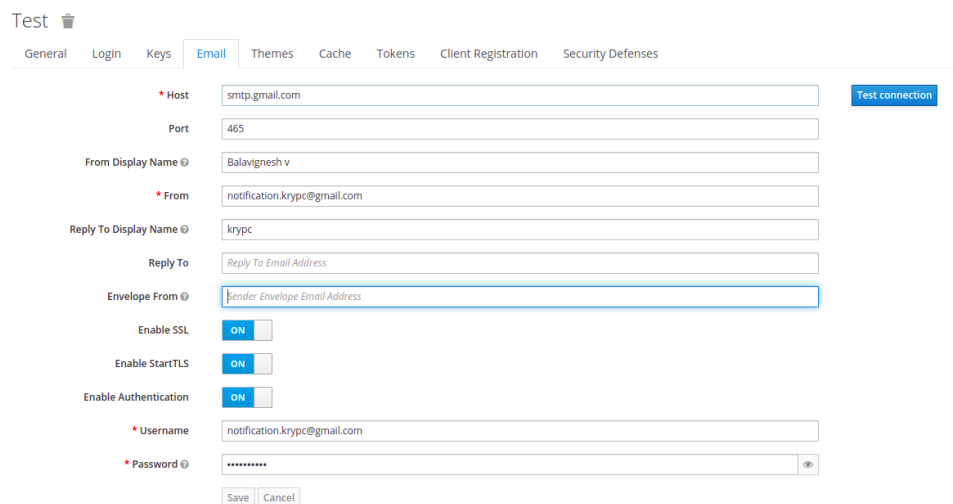


The image shows the Keycloak 'Email' configuration form. The left sidebar contains the 'Configure' menu with 'Email' selected. The form fields are as follows:

- Host: smtp.gmail.com
- Port: 465
- From Display Name: Balavignesh v
- From: balavignesh1018@gmail.com
- Reply To Display Name: krypc
- Reply To: Reply To Email Address
- Envelope From: Sender Envelope Email Address
- Enable SSL: ON
- Enable StartTLS: ON
- Enable Authentication: ON
- Username: balavignesh1018@gmail.com
- Password: (masked)

Buttons: Test connection, Save, Cancel.

- Click on Email tab, fill all the necessary details as below image and save it. Email notification will be enabled.

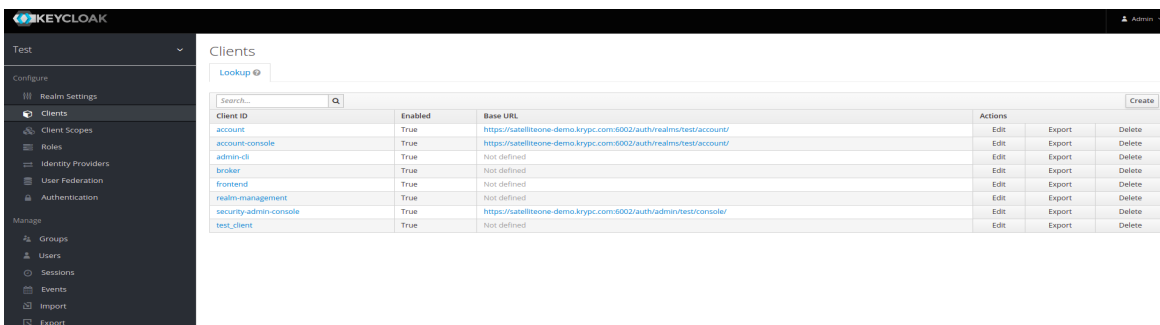


This is a duplicate of the Keycloak 'Email' configuration form shown above.

Note: Form inputs for *From* and *username* should be valid emailID. Password should be app password configured in the mail account.

Steps for RedirectURL configuration in keycloak

- Login as administrator user
- Click on Clients under left menu bar and select frontend as ClientID



The image shows the 'Clients' table in Keycloak. The table has columns: Client ID, Enabled, Base URL, and Actions. The 'Frontend' client is highlighted.

Client ID	Enabled	Base URL	Actions
account	True	https://satelliteone-demo.krypc.com:6002/auth/realms/test/accounts/	Edit Export Delete
account-console	True	https://satelliteone-demo.krypc.com:6002/auth/realms/test/accounts/	Edit Export Delete
admin-cli	True	Not defined	Edit Export Delete
broker	True	Not defined	Edit Export Delete
frontend	True	Not defined	Edit Export Delete
realm-management	True	Not defined	Edit Export Delete
security-admin-console	True	https://satelliteone-demo.krypc.com:6002/auth/admin/test/console/	Edit Export Delete
test_client	True	Not defined	Edit Export Delete

- In Frontends settings form, find *RootURL*, *valid RedirectURLs*, *Web Origins* and add the entries as follows:

RootURL – the values to this input should be UI application URL

example: <https://satelliteone-demo.example.com>

Valid Redirect URLs – it includes multiple values as URLs.

- UI application URL with * as route example - <https://satelliteone-demo.example.com/>*
- Keycloak URL with * as route example - <https://satelliteone-demo.exapmle.com:8443/>*


Web Origins - the values to this input should be keycloak URL with * as route example -

<https://satelliteone-demo.example.com:8443/>*

Find below Image as reference

Root URL	<input type="text" value="https://satelliteone-demo.krypc.com"/>
* Valid Redirect URIs	<input type="text" value="https://satelliteone-demo.krypc.com/*"/> - <input type="text" value="https://satelliteone-demo.krypc.com:8443/*"/> +
Base URL	<input type="text"/>
Admin URL	<input type="text"/>
Web Origins	<input type="text" value="https://satelliteone-demo.krypc.com:8443/*"/> - <input type="text"/> +

- Match all the settings as below image and save it. Redirection settings has be changed successfully.

Frontend 

Settings	Roles	Client Scopes	Mappers	Scope	Revocation	Sessions	Offline Access	Installation
Client ID	<input type="text" value="frontend"/>							
Name	<input type="text"/>							
Description	<input type="text"/>							
Enabled	<input checked="" type="checkbox"/>							
Consent Required	<input type="checkbox"/>							
Login Theme	<input type="text"/>							
Client Protocol	<input type="text" value="openid-connect"/>							
Access Type	<input type="text" value="public"/>							
Standard Flow Enabled	<input checked="" type="checkbox"/>							
Implicit Flow Enabled	<input type="checkbox"/>							
Direct Access Grants Enabled	<input type="checkbox"/>							
Root URL	<input type="text" value="https://satelliteone-demo.krypc.com"/>							
* Valid Redirect URIs	<input type="text" value="https://satelliteone-demo.krypc.com/*"/> - <input type="text" value="https://satelliteone-demo.krypc.com:8443/*"/> +							
Base URL	<input type="text"/>							
Admin URL	<input type="text"/>							
Web Origins	<input type="text" value="https://satelliteone-demo.krypc.com:8443/*"/> - <input type="text"/> +							

You are now ready with your Satellite.



6. Commands for managing Docker services

To check the status of docker containers:

```
docker-compose -f <docker-compose-file> ps
```

To stop the container

```
docker-compose -f <docker-compose-file> down
```

To restart containers

```
docker-compose -f <docker-compose-file> restart
```

To bring containers up and running

```
docker-compose -f <docker-compose-file> up -d
```

To get the logs of containers

```
docker-compose -f <docker-compose-file> logs <service-name-in-  
compose-file>
```