6 - Hyperledger Fabric Node Deployment

This page provides information on the steps and the proper sequence of execution that is required to deploy the HLF nodes. There are common and specific steps for each of the openIDL node types (carrier, analytics, ordering). The operator of the node has to execute only the common and specific step of the owned node type.

The steps are clustered in dedicated sections depending on their purpose.

Steps:

Step	Description	Node Type
Deploy Fabric Operator/Setup Environment Context	This step includes environment configuration actions (i.e. install tools and libraries) and the deployment of critical components used to bootstrap the HLF nodes (ingress, fabric operator, fabric operator console, vault, etc.). Those actions are common and should be performed by the operator of any node type.	Carrier Analytics Ordering
Deploy HLF Nodes with Operator Console		
• Deploy Ordering Service		Ordering
• Deploy analytics endorsing organization		Analytics
• Deploy a carrier endorsing organization		Carrier
Export, share and import the MSP / Ordering Service definitions		Carrier Analytics Ordering
Deploy the openIDL channels		Carrier Analytics Ordering
Deploy the openIDL chaincodes		Carrier Analytics

Prerequisites:

- AWX is configured (see the AWX Setup and Configuration chapter)
- Access to AWX with the organization user
- Configuration is done and available at a private git repository
- DNS entries are correct and maintained in Route53. The DNS node config ansible playbook expects to have an entry (hosted zone) in Route53. The hosted zone name should match the configured main_domain value in the node config file. The Name Servers should be correctly configured and maintained on the root domain level.
- The configured main domain DNS can be resolved on internet. The node communicates (using TLS) to other nodes on the network over internet.

Deploy Fabric Operator/Setup Environment Context

Run the following ansible jobs in the order below:

AWX Job Template	Notes
<env_id>-<org_id>-environment-setup	Installs the required software on the bastion host, and setups AWS CLI access.
<env_id>-<org_id>-deploy-fabric-ingress	Deploy k8s ingress controller for the HLF k8s cluster
<env_id>-<org_id>-dns-config-blk	After the ingress is deployed, DNS entries must be setup in order to route the traffic from the configured domain to the k8s cluster load balancers. Make sure the DNS entries are setup properly before proceeding with the configured domain in the configuration file
<env_id>-<org_id>-deploy-vault	Deploy vault cluster in HLF k8s cluster. The access credential to the vault instance are stored in AWS secrets manager
<env_id>-<org_id>-deploy-fabric-operator	Deploy fabric operator k8s controller
<env_id>-<org_id>-deploy-fabric-console	Deploy operator console. The access to the console is at the address. Note that the address is not configurable as it is assembled by convention. The user and password to access the console are those defined in the credential "fabric-console" `https://{{console_namespace}}-{{console_name}}-console.{{console_domain}}`

Deploy the HLF Nodes with Operator Console

The deployment of the HLF nodes is done using the fabric operator console web-based UI. The console provides an intuitive interface to perform a proper configuration and deployment of various HLF components. The fabric console is also used to operate and manage the HLF nodes post-deployment (i.e. join a new channel, deploy a new chaincode version, etc).

- **Deploy an Ordering Service**

The HLF Ordering Service is an essential part of the openIDL network. Those nodes are used to order the transactions into blocks and distribute them on the network. An HLF ordering service can be deployed and managed by anyone on the network. In order to streamline the network management the openIDL hosts and manages an Ordering Service that serves the transactions on the openIDL network. The carrier and analytics nodes that are part of the openIDL can join the channels served by the openIDL Ordering Service in order to become part of the network.

The creation of the ordering service and the ordering nodes (orderers) is an essential part of any HLF network deployment. The ordering nodes are used to form ordering clusters that serve the ordering of the blocks on the application channels. The ordering service on an application channel can be composed of multiple orderers operated and managed by different organizations on the network. The set of orderers that participate on a particular application channel may be updated anytime through the life-cycle of the application channel using channel update transactions.

The ordering service at openIDL is managed and operated by openIDL to serve the needs of the network members and their dedicated nodes (endorsing organizations).

ⓘ Note that the name(id) pattern of the identities below must be respected as those identities are also used in the application deployment. The variables below used in the naming convention of the resource names are as defined in the organization private config yaml file.

Steps:

Step	Details	Notes
Deploy Certificate Authority	Console Nodes Create CA Create new CA - CA name: <ordering_org_id> - CA admin enroll id: <ordering_org_id>-ca-admin - CA enrollment secret: anything that can be remembered (note it down)	The CA admin is used to register identities with the CA. That includes identities for the organization orderers/peers, organization admins, and the organization application users.

Accosiate CA admin user identity	Console Nodes Ordering Service CA Navigate to the details page of the ordering service CA created above. Make sure the CA is up and running (green light). Associate (enroll) the CA admin identity registered above during the CA deployment • Enroll id: <ordering_org_id>-ca-admin (display name <ordering_org_id>-ca-admin)	
Register the ordering service (MSP admin) admin identity	Console Nodes Ordering Service CA Navigate to the details page of the ordering service CA created above. Register the org admin user using the deployed CA above • Enroll id: <ordering_org_id>-msp-admin • Type: admin • Enroll secret: should be remembered (note it down)	The organization admin user is enrolled with the CA when the organization is created (next step).
Create the ordering service MSP definition	Console Nodes Organizations Create MSP Definition • MSP name: <ordering_org_id> • MSP id: <ordering_org_id> • Enroll ID: <ordering_org_id>-msp-admin • Identity Name: <ordering_org_id>-msp-admin	Use the enrollment secret as provided above. The enrolled admin PKI is stored in vault
Register the ordering node Identity with the ordering service CA	Console Nodes Ordering Service CA On the org CA node register the ordering node identity • Register User • Enroll Id: <ordering_org_id>-orderer • Enroll Secret: remember/note it down, it will be used to enrol the identity later • Type: orderer	
Enroll Ordering Service Admin TLS	Console Nodes Ordering Service CA Navigate to the details page of the ordering service CA created above. On the ordering service CA page enroll the identity <ordering_org_id>-msp-admin with the TLS Certificate Authority. • Choose enroll identity from the three-dot menu. • Choose TLS Certificate Authority in the CA dropdown • Use the same enrollment secret as entered during registration of the ordering service admin identity. • Store the identity in the wallet under the name <ordering_org_id>-msp-admin-tls.	The enrolment of the ordering service admin user with the ordering service TLS CA is essential. It allows you to administrate the ordering nodes in order to join/remove them on application channels.

| Create the ordering service | Console Nodes Ordering Service Create an ordering service

• Ordering Service name: <ordering_org_id>
• Without system channel
• Three ordering node
• Ordering Service enroll id: <ordering_org_id>-orderer
• Choose the ordering org MSP and ordering org CA
• Admin Identity: <ordering_org_id>-msp-admin | More ordering nodes may be added later to scale and distribute the ordering service nodes. |

- **Deploy analytics (carrier) endorsing organization**

The below steps are common for analytics and carrier node types.

The variables below used in the naming convention of the resource names are as defined in the organization's private config file.

Steps:

Step	Details	Notes
Deploy Certificate Authority	Console Nodes Add Certificate Authority Create new CA • CA name: <org_id> • CA admin enroll id: <org_id>-ca-admin • CA enrollment secret: anything that can be remembered (note it down) Associate (enroll) the CA admin identity registered above during the CA deployment • Enroll id: <org_id>-ca-admin (display name <org_id>-ca-admin)	
Accosiate CA admin user identity	Console Nodes Certificate Authorities Navigate to the details page of the endorsing org CA created above. Make sure the CA is up and running (green light). Associate (enroll) the CA admin identity registered above during the CA deployment • Enroll id: <org_id>-ca-admin (display name <org_id>-ca-admin)	
Register the organization admin identity	Console Nodes Certificate Authorities Register the org admin user using the deployed CA above • Enroll id: <org_id>-msp-admin • Type: admin • Enroll secret: should be remembered (note it down)	
Create the MSP definition for the organization	Console Organizations Create MSP Definition • MSP name: <org_id> • MSP id: <org_id> • Enroll ID: <org_id>-msp-admin • Identity Name: <org_id>-msp-admin • Generate Certificates • Next and deploy	The same step as for the ordering organization

Register the peer node identity	Console Nodes Certificate Authorities On the endorsing org CA node register the peer node identity • Enroll Id: <org_id>-peer1 • Enroll Secret: remember it and note it down • Type: peer	
Deploy the peer node	Console Nodes Add Peer • Peer enroll id (name): <org_id>-peer1 • Choose the endorsing org MSP and CA created above • Choose the peer-enrolled identity as registered above • Enter the enrollment secret of the <org_id>-peer1 as registered above • Accosiate the amdin Identity: <org_id>-msp-admin	More peer nodes can be added later to scale and distribute the peers of the endorsing organization

- **Deploy a carrier endorsing organization**

Follow the same steps as when deploying an analytics-endorsing organization.

ⓘ The openIDL network requires a minimum of carrier, analytics, and ordering service nodes in order to operate as designed. However, the network can be expanded by adding additional carriers, analytics, and ordering service nodes.

ⓘ In the real world, the carrier and analytics nodes are deployed on dedicated accounts operated by the respective business entities. It is possible though to operate the nodes of different endorsing organizations (carriers/analytics) on the same infrastructure using the same fabric operator console.

Export, share and import the MSP / Ordering Service definitions

In order to deploy application channels and connect the endorsing organizations on the openIDL network, the definition of each of the organization (MSP) should be exported, shared with the other organizations and respectively imported in their own fabric console. This enables the organizations to securely build the permissions on the application channels and assign the corresponding security policies.

ⓘ The import/export is required only if the organizations are deployed and managed by different fabric consoles. If the same console is used the organizations will be already available

Steps:

Step	Actor	Note
Export the MSP definitions to a file (json)	carrier, analytics, ordering	Console Organizations <org_id> export button in organization tile. The administrator has to export its own operated organization definition
Export the ordering service	ordering	Console Nodes Ordering Services <ordering_org_id> export button in ordering clusters tile
Share the definition of the ordering service	ordering	The downloaded file export of the ordering service can be shared with the rest of the organizations using a dedicated private git repository.
Share the MSP definition json file with the other organizations	carrier, analytics, ordering	The downloaded file above can be shared with the rest of the organizations using a dedicated private git repository.
Import the MSP definitions of the other organizations	carrier, analytics, ordering	Console Organizations Import MSP definition Every administrator has to import the MSP definitions of the rest of the network participating MSPs in their own fabric console. This is essential to operate the network like deploying/managing application channels.

Import the ordering service	carrier, analytics	Console Ordering Services Add ordering services import an existing ordering services
		Every administrator of an endorsing organization (analytics/carrier) has to import the ordering service that will be used to serve the application channels. Use the shared exported ordering service file provided by the administrator of the ordering service.

Deploy the openIDL channels

The openIDL HLF channels are used to perform transactions endorsed by the participating nodes. There is a public channel to record public data (i.e. data call) and private channels to manage the private transaction between a carrier and analytics node (i.e. securely share the carrier data call extraction with the analytics node).

> ⓘ The channel names in openIDL must follow a specific naming convention as specified in the table above. The channel names must be defined as per the above instructions.

By default, the channels have the following policies:

- Lifecycle endorsement policy (deploy chaincode on the channel): The majority must approve
- Smart contract endorsement policy: The majority must approve

By default, the ordering nodes of the ordering organization will be added to the ordering cluster that will be serving the channel.

> ⓘ openIDL network deployment doesn't depend on or require any custom definition of HLF access control list
>
> More details: https://hyperledger-fabric.readthedocs.io/en/latest/access_control.html

Steps:

Step	Actor	Notes
Create the openIDL default channel	ordering	Console Channels Create channel Channel Name: `defaultchannel` Organizations: • Add the MSPs of the carriers and the analytics organizations. Choose as an operator the analytics MSP. The carrier MSPs may have writer/reader permissions. Ordering organizations: • Add the MSP of the ordering service. Assign Administrator permission Create the channel genesis block Join the ordering service nodes to the channel

| Create the openIDL carrier analytics private channel

Repeat the step for every carrier in the network | ordering | Console Channels Create channel

Note that this is the private channel between a single carrier node and the analytics node. Therefore the step should be repeated in order to create specific channels for each pair of carrier/analytics nodes.

Channel Name:

`<analytics org id>-<carrier org id>`

Organizations:

- Add the MSPs of the carrier and the analytics organizations. Choose as an operator the analytics MSP. The carrier MSP may have writer/reader permissions.

Ordering organizations:

- Add the MSP of the ordering service. Assign Administrator permission

Create the channel genesis block

Join the orderering service nodes to the channel |
|---|---|---|
| Join peers on public /common/ default channel | Analytics, Carrier | Console Nodes Peer Join channel

The administrators of the analytics and carriers nodes have to join their own peers on the defaultchannel as created above

Select the ordering service

Enter the channel name:

`defaultchannel`

Select the peers to join the channel and mark it as anchor

Every MSP must have an anchor peer on the channel in order to enable the private communication capability of the channel.

Anchor peers can be updated through channel configuration update transactions. |
| Join peers on the private channels | Analytics | Console Nodes Peer Join channel

The administrator of the anaylitics node must join the analytics node peer(s) to all the channel created to serve the private communication between the analytics node and the carriers.

Enter the channel name:

`<analytics org id>-<carrier org id>`

Select the peers to join the channel and mark it as anchor

Every MSP must have an anchor peer on the channel in order to enable the private communication capability of the channel.

Anchor peers can be updated through channel configuration update transactions. |
| Join peers on the private channel | Carrier | Console Nodes Peer Join channel

Every administrator of the carrier nodes must join their own carrier peer(s) the private channel created above to serve the private communication between the analytics node and the carrier node.

Enter the channel name:

`<analytics org id>-<carrier org id>`

Select the peers to join the channel and mark it as anchor

Every MSP must have an anchor peer on the channel in order to enable the private communication capability of the channel.

Anchor peers can be updated through channel configuration update transactions. |

Deploy the openIDL chaincodes

The openIDL chaincode implements the data call business logic that is endorsed by the peers on the network.

Steps:

Step	Actor	Details

Propose openIDL default chaincode definition	Analytics	Console Channels **defaultchannel** Propose smart contract definition
		
		Chaincode:
		
		`openidl-cc-default`
		
		- Organization: the <org_id>; Organization msp admin: <org_id>-msp-admin
		- Install the smart contract by using the package file (add file):
		
		`https://github.com/orgs/openidl-org/packages?repo_name=openidl-main`
		`The latest version of (*.tgz): openidl-chaincode.openidl-cc-default`
		
		- Smart Contract version: v1 (version increases with every chaincode upgrade)
		- Use the default values for the rest of the steps
		
		The default chaincode is deployed on the defaultchannel and is used to record the data calls issued by the analytics node.

Approve the proposed chaincode definition	Carrier	Console Notifications
		
		Chaincode:
		
		`openidl-cc-default`
		
		- Organization: the <org_id>; Organization msp admin: <org_id>-msp-admin
		- Install the smart contract by using the package file (add file):
		
		`https://github.com/orgs/openidl-org/packages?repo_name=openidl-main`
		`The latest version of (*.tgz): openidl-chaincode.openidl-cc-default`
		
		- Smart Contract version: v1 (version increases with every chaincode upgrade)
		- Use the default values for the rest of the steps
		
		The default chaincode is deployed on the defaultchannel and is used to record the data calls issued by the analytics node.

Commit the chaincode proposal	Analytics	Console Notifications Trigger commit of the approved chaincode definition. After a successful commit the chaincode deployment is done. Chaincode: `openidl-cc-default`

Propose openIDL analytics-carrier private chaincode definition	Analytics	Console Channels **<analytics org id>-<carrier org id>** Propose smart contract definition
		
		Chaincode:
		
		`openidl-cc-analytics-carrier`
		
		• Organization: the <org_id>; Organization msp admin: <org_id>-msp-admin
		• Install the smart contract by using the package file (add file):
		
		`https://github.com/orgs/openidl-org/packages?repo_name=openidl-main`
		`The latest version of (*.tgz): openidl-cc-analytics-carrier`
		
		• Smart Contract version: v1 (version increases with every chaincode upgrade)
		• If required select the chaincode reqruies init option
		
		   ⓘ In case the used public channel has name other than the default one (defaultchannel), the chaincode should be initialized. Therefore it is required to select the chaincode requires init option.
		
		      During initialization (see below) the name of the public channel is passed to the chaincode and stored on the ledger (the carrier/analytics channel).
		
		Use the following template to create a private data collection (PDC)) definition file on your local file system (replace the values with the analytics and carrier specifics.
		
		`[`
		`    {`
		`        "name":"<analytics org id>_<carrier org id>_pdc",`
		`        "policy": "OR('<analytics org id>.member', '<carrier org id>.member')",`
		`        "requiredPeerCount":0,`
		`        "maxPeerCount":0,`
		`        "blockToLive":0`
		`    }`
		`]`
		
		• Choose the templated local PDC file to add as PDC definition of the deployment
		• Use the default values for the rest of the steps
		
		Repeat the above step for each analytics-carrier channel
		
		The analytics-carrier chaincode is deployed on each of the analytics-carrier channels. It is used to record the extraction of carrier data on the private data collection shared between the carrier and the analytics nodes.

Approve openIDL analytics-carrier private chaincode definition	Carrier	Console Notifications
		
		Chaincode:
		
		`openidl-cc-analytics-carrier`
		
		• Organization: the <org_id>; Organization msp admin: <org_id>-msp-admin
		• Install the smart contract by using the package file (add file):
		
		`https://github.com/orgs/openidl-org/packages?repo_name=openidl-main`
		`The latest version of (*.tgz): openidl-cc-analytics-carrier`
		
		• Smart Contract version: v1 (version increases with every chaincode upgrade)
		• Use the default values for the rest of the steps
		
		Repeat the above step for each analytics-carrier channel
		
		The analytics-carrier chaincode is deployed on each of the analytics-carrier channels. It is used to record the extraction of carrier data on the private data collection shared between the carrier and the analytics nodes.

Commit the chaincode proposal	Analytics	Console Notifications Trigger commit of the approved chaincode definition. After a successful commit the chaincode deployment is done. Chaincode: `openidl-cc-analytics-carrier`

| Initialize the chaincode | Carrier

or

Analytics | In case the public channel name is other than "defaultchannel", the private carrier/analytics chaincode must be initialized with the name of the public channel.

This step should be performed by the carrier node admin or the analytics node admin.

@Carrier admin: The admin of the carrier can login to the AWX instance and launch the template "<env_id>-<org_id>-chaincode-init".

@Analytics admin: The analytics node admin can login to the AWX instance and navigate to the template "<env_id>-<org_id>-chaincode-init". The admin should launch the template with additional variable "init_on_channel_id".

The variable should define the name of the private (carrier/analytics) channel where the chaincode is deployed and should be initialized. The admin user can repeat that step for every specific carrier/analytics channel. |