

Blueventure:
Blockchain LabTrack-and-Trace Blockchain
Workshop for Hyperledger
Fabric 2.2 (BETA)▼ Create a Hyperledger Fabric
Network

- ▶ Create Network & Member
- ▶ Accept invite and create
Supplier member
- Congratulations

▶ Setup Development
Environment

▶ Set up a Fabric client

▶ Write and deploy chaincode

▼ Invoking chaincode via API

Deploy CDK application

Store user secrets

Create Coanito users

▼ AWS account access

[Open AWS console](#)

(us-east-1)

[Get AWS CLI credentials](#)

Exit event

[Event dashboard](#) > [Invoking chaincode via API](#) > Deploy CDK application

Deploy CDK application

Each consortium member should execute the following commands in their Cloud9 terminal.

You will deploy the components highlighted in the [architecture](#) using [AWS Cloud Development Kit \(AWS CDK\)](#). The AWS CDK lets you build reliable, scalable, cost-effective applications in the cloud with the considerable expressive power of a programming language.

Install NodeJS v16 and set as the default version

```
1 nvm install lts/gallium
2 nvm use lts/gallium
3 nvm alias default lts/gallium
```



The AWS CDK includes the CDK Toolkit (also called the CLI), a command line tool for working with your AWS CDK apps and stacks. Install the CDK toolkit

```
1 npm install -g aws-cdk@2.55.1
2 cdk --version
```



Deploying stacks with the AWS CDK requires dedicated Amazon S3 buckets and other containers to be available to AWS CloudFormation during deployment. Creating these is called bootstrapping. To bootstrap, issue

```
1 cdk bootstrap aws://$MEMBER_AWS_ID/$AWS_DEFAULT_REGION
```



Now that you have installed and bootstrapped CDK, clone the supply chain CDK application code repository. The CDK application code is written in [TypeScript](#). The components are defined in `lib/stack.ts` file

```
1 cd $HOME/environment
2 git clone --depth=1 https://github.com/aws-samples/amb-hf-workshop-supplychain-app
```



Install the CDK application dependencies

```
1 cd $HOME/environment/amb-hf-workshop-supplychain-app
2 npm ci
```



Blueventure: Blockchain Lab

Track-and-Trace Blockchain Workshop for Hyperledger Fabric 2.2 (BETA)

▼ Create a Hyperledger Fabric Network

► Create Network & Member

► Accept invite and create Supplier member

Congratulations

► Setup Development Environment

► Set up a Fabric client

► Write and deploy chaincode

▼ Invoking chaincode via API

[Deploy CDK application](#)

Store user secrets

Create Coanito users

▼ AWS account access

[Open AWS console](#)

(us-east-1) 

[Get AWS CLI credentials](#)

Exit event

Lambda layers are a powerful way of bundling Lambda dependencies in a way that makes them more easily reused. They can also improve the performance of Lambda functions by making it easier for dependencies to be pre-loaded prior to Lambda invocation. Finally, by putting all your dependencies in a layer, your actual Lambda code can be kept lean, which makes it a lot easier to edit and maintain, even in the AWS Management Console if you prefer. Install the dependencies for Lambda Layer

```
1 cd $HOME/environment/amb-hf-workshop-supplychain-app
2 npm ci --omit=dev --prefix lib/lambda-layer/nodejs
```

The `lib/lambda` folder consists of files used by the Lambda function. The code in the file `lib/lambda/index.js` retrieves the necessary arguments from the AppSync caller, uses them to retrieve the user's credentials from AWS Secrets Manager, and finally submits a query or transaction to the Fabric peer nodes, returning the result to AppSync.

Generate [connection profile](#) JSON file from the Fabric environment settings. This file is used by the Lambda function to determine how to connect to the Hyperledger Fabric blockchain network. Observe that `connection-profile.json` and `managedblockchain-tls-chain.pem` files are created in `lib/lambda` folder.

```
1 cd $HOME/environment/amb-hf-workshop-supplychain-app
2 ./scripts/setupConnectionProfile.sh
```

Retrieve environment variables needed to deploy the CDK application

```
1 export INTERFACE=$(curl --silent http://169.254.169.254/latest/meta-data/network/interfaces/mac/)
2 export SUBNETID=$(curl --silent http://169.254.169.254/latest/meta-data/network/interfaces/mac/${INTERFACE}/subnet-id)
3 export VPCID=$(curl --silent http://169.254.169.254/latest/meta-data/network/interfaces/mac/${INTERFACE}/vpc-id)
4 export SECURITY_GROUPS=$(curl --silent http://169.254.169.254/latest/meta-data/network/interfaces/mac/${INTERFACE}/security-group-ids)
5 export GROUPID=$(aws ec2 describe-security-groups --group-ids $SECURITY_GROUPS --filter "Name=group-name, Values=HFClientAndEndpoint" --
6 export DEFAULT_GROUP_ID=$(aws ec2 describe-security-groups --filter "Name=group-name, Values=default" --query "SecurityGroups[?VpcId=]"
```

Deploy the application. The file `deploy-out.json` contains the stack output that will be needed in subsequent steps

```
1 cd $HOME/environment/amb-hf-workshop-supplychain-app
2 cdk deploy --json --outputs-file deploy-output.json
```

A successful output of the deployment should look similar to this

```
1 Outputs:
2 SupplyChainApp.CognitoPoolID = us-east-1_JM1yeHYpg
3 SupplyChainApp.GraphQLURL = https://mt1tpauhj5hnzo27jw6schw574.appsync-api.us-east-1.amazonaws.com/graphql
4 SupplyChainApp.Worker1PKSecret = arn:aws:secretsmanager:us-east-1:123456789012:secret:amb/supplychain/rtworker/pk-VJzFGf
5 SupplyChainApp.Worker1SignCertSecret = arn:aws:secretsmanager:us-east-1:123456789012:secret:amb/supplychain/rtworker/signcert-dwbXi8
6
7
```

**Blueventure:
Blockchain Lab**

Track-and-Trace Blockchain
Workshop for Hyperledger
Fabric 2.2 (BETA)

▼ Create a Hyperledger Fabric
Network

- ▶ Create Network & Member
- ▶ Accept invite and create
Supplier member

Congratulations

- ▶ Setup Development
Environment
- ▶ Set up a Fabric client
- ▶ Write and deploy chaincode
- ▼ Invoking chaincode via API

Deploy CDK application

Store user secrets

Create Coanito users

▼ **AWS account access**

[Open AWS console](#)
(us-east-1)

[Get AWS CLI credentials](#)

Exit event

```
SupplyChainApp.Worker2PKSecret = arn:aws:secretsmanager:us-east-1:123456789012:secret:amb/supplychain/rtseller/pk-DFkgmJ
SupplyChainApp.Worker2SignCertSecret = arn:aws:secretsmanager:us-east-1:123456789012:secret:amb/supplychain/rtseller/signcert-0EsJLh
```

You should also observe the same output in JSON format by issuing the following

```
1 cd $HOME/environment/amb-hf-workshop-supplychain-app
2 cat deploy-output.json | jq .
```



Previous

Next

