10/19/23. 8:48 AM Blueventure: Blockchain Lab

Blueventure: **Blockchain Lab** 

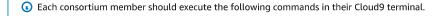


Track-and-Trace Blockchain Workshop for Hyperledger Fabric 2.2 (BETA)

X

## **Deploy CDK application**

▼ Create a Hyperledger Fabric



Network

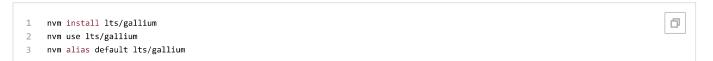
You will deploy the components highlighted in the architecture using AWS Cloud Development Kit (AWS CDK) [2]. The AWS CDK lets you build reliable, scalable, cost-effective applications in the cloud with the considerable expressive power of a programming language.

Create Network & Member

Install NodeJS v16 and set as the default version

Accept invite and create Supplier member

Congratulations



▶ Setup Development

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

Environment ▶ Set up a Fabric client

CDK toolkit

▶ Write and deploy chaincode ▼ Invoking chaincode via API



## **Deploy CDK application**

Deploying stacks with the AWS CDK requires dedicated Amazon S3 buckets and other containers to be available to AWS CloudFormation

Store user secrets Create Cognito users

during deployment. Creating these is called bootstrapping. To bootstrap, issue

▼ AWS account access

git clone --depth=1 https://github.com/aws-samples/amb-hf-workshop-supplychain-app

cdk bootstrap aws://\$MEMBER\_AWS\_ID/\$AWS\_DEFAULT\_REGION

Open AWS console

Now that you have installed and bootstrapped CDK, clone the supply chain CDK application code repository. The CDK application code is

(us-east-1) 🗇

written in TypeScript 2. The components are defined in lib/stack.ts file

Get AWS CLI credentials

Install the CDK application dependencies

cd \$HOME/environment

Exit event

ð cd \$HOME/environment/amb-hf-workshop-supplychain-app npm ci

o

ð

**1** 

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 Cognito 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

```
cd $HOME/environment/amb-hf-workshop-supplychain-app
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 I 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

```
export INTERFACE=$(curl --silent http://169.254.169.254/latest/meta-data/network/interfaces/macs/)

export SUBNETID=$(curl --silent http://169.254.169.254/latest/meta-data/network/interfaces/macs/${INTERFACE}/subnet-id)

export VPCID=$(curl --silent http://169.254.169.254/latest/meta-data/network/interfaces/macs/${INTERFACE}/vpc-id)

export SECURITY_GROUPS=$(curl --silent http://169.254.169.254/latest/meta-data/network/interfaces/macs/${INTERFACE}/security-group-ids)

export GROUPID=$(aws ec2 describe-security-groups --group-ids $SECURITY_GROUPS --filter "Name=group-name, Values=HFClientAndEndpoint" --

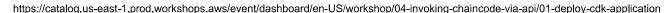
export DEFAULT_GROUP_ID=$(aws ec2 describe-security-groups --filter "Name=group-name, Values=default" --query "SecurityGroups[?VpcId=='"
```

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

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

A successful output of the deployment should look similar to this

```
Outputs:
SupplyChainApp.CognitoPoolID = us-east-1_JM1yeHYpg
SupplyChainApp.GraphQLURL = https://mtltpauhj5hnzo27jw6schw574.appsync-api.us-east-1.amazonaws.com/graphql
SupplyChainApp.Worker1PKSecret = arn:aws:secretsmanager:us-east-1:123456789012:secret:amb/supplychain/rtworker/pk-VJzFGf
SupplyChainApp.Worker1SignCertSecret = arn:aws:secretsmanager:us-east-1:123456789012:secret:amb/supplychain/rtworker/signcert-dwbXi8
```





**1** 

