

## Amazon Managed Blockchain (Hyper Ledger Fabric) ワークショップ

2021/04/07

シニアエバンジェリスト

亀田 治伸

## 【環境構築】

1. マネージメントコンソールで Cloud9 にアクセスします。

AWS Cloud9 > Your environments

Your environments (1)

Open IDE

View details

Edit

Delete

Create environment

2. [Create environment]を押します
3. 適当な名前を付け、[Next step]を押します

## Name environment

### Environment name and description

**Name**  
The name needs to be unique per user. You can update it at any time in your environment settings.

Limit: 60 characters

**Description - Optional**  
This will appear on your environment's card in your dashboard. You can update it at any time in your environment settings.

*Write a short description for your environment*

Limit: 200 characters

CancelNext step

4. Instance Type を[Other...]から[t2.medium]にします

#### Instance type

- ☐ t2.micro (1 GiB RAM + 1 vCPU)  
Free-tier eligible. Ideal for educational users and exploration.
- ☐ t3.small (2 GiB RAM + 2 vCPU)  
Recommended for small-sized web projects.
- ☐ m5.large (8 GiB RAM + 2 vCPU)  
Recommended for production and general-purpose development.
- ☒ Other instance type  
Select an instance type.

t2.medium ▼

5. [Network settings (advanced)]を開き、VPC と Public subnet を選びます。(Cloud9 は通常開発環境なので、Private に構築しますが、今日の手順では Cloud9 のブロックチェーンのメンバークライアントとして設定するため、Public に設定します) (VPC はなんでもいいですが、Public Subnet があること。インターネットに出れる環境であること、が必要です)

#### ▼ Network settings (advanced)

##### Network (VPC)

Launch your EC2 instance into an existing Amazon Virtual Private Cloud (VPC) or create a new one. To allow the AWS Cloud9 environment to connect to its EC2 instance, attach an internet gateway (IGW) to your new VPC.

HandsOn | vpc-aa1644cf ▼



Create new VPC

##### Subnet

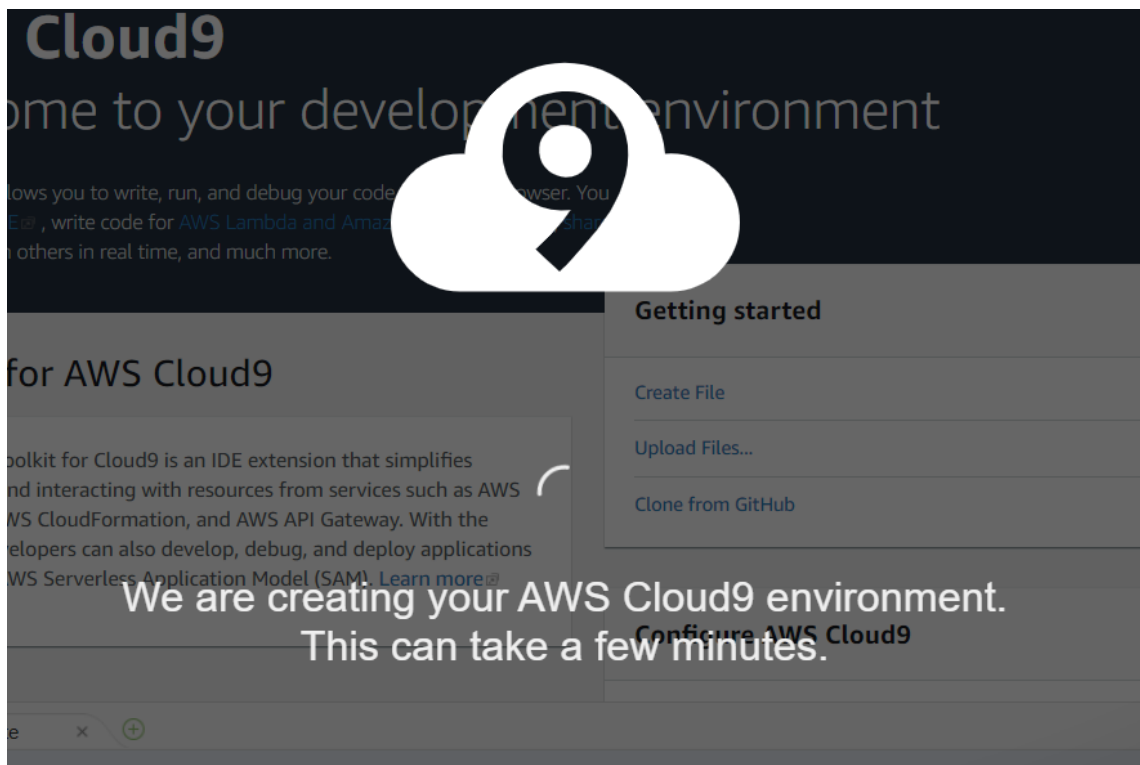
Select a public subnet in which the EC2 instance is created. (For a private subnet, you must create an environment that connects to its instance via Systems Manager.)

パブリックサブネット | subnet-1e2d7d47 | Non-default... ▼



Create new subnet

6. [Next step]を押します。次の画面ではそのまま[Create environment]を押します。
7. Cloud9 が構築中になりますので、コンソールにアクセスできるまでしばらく待ちます



8. ブラウザ別タブで IAM ロールの画面にいきます

## Identity and Access Management (IAM)

### ダッシュボード

#### ▼ アクセス管理

グループ

ユーザー

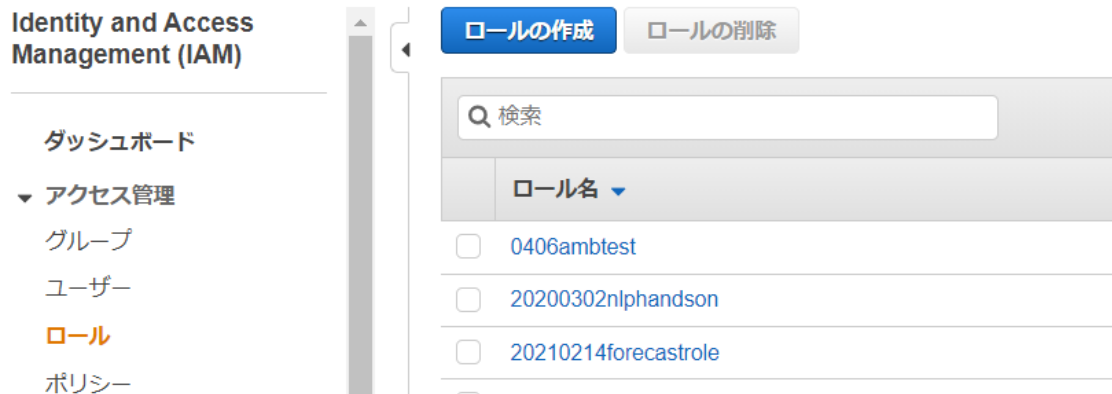
ロール

ポリシー

ID プロバイダー

アカウント設定

9. [ロール]をクリックし、[ロールの作成]を押します



10. [一般的なユースケース]から EC2 を選び、[次のステップ:アクセス権限]を押します

信頼されたエンティティの種類を選択



AWS のサービスによるアクションの代行を許可します。 [詳細はこちら](#)

ユースケースの選択

一般的なユースケース

**EC2**

Allows EC2 instances to call AWS services on your behalf.

**Lambda**

Allows Lambda functions to call AWS services on your behalf.

または、サービスを選択してユースケースを表示します

API Gateway	CloudWatch Events	EKS	IoT SiteWise	RDS
AWS Backup	CodeBuild	EMR	IoT Things Graph	Redshift
AWS Chatbot	CodeDeploy	ElastiCache	KMS	Rekognition
AWS Marketplace	CodeGuru	Elastic Beanstalk	Kinesis	RoboMaker
AWS Support	CodeStar Notifications	Elastic Container Registry	Lake Formation	S3
Amplify	Comprehend	Elastic Container Service	Lambda	SMS
AppStream 2.0	Config	Elastic Transcoder	Lex	SNS

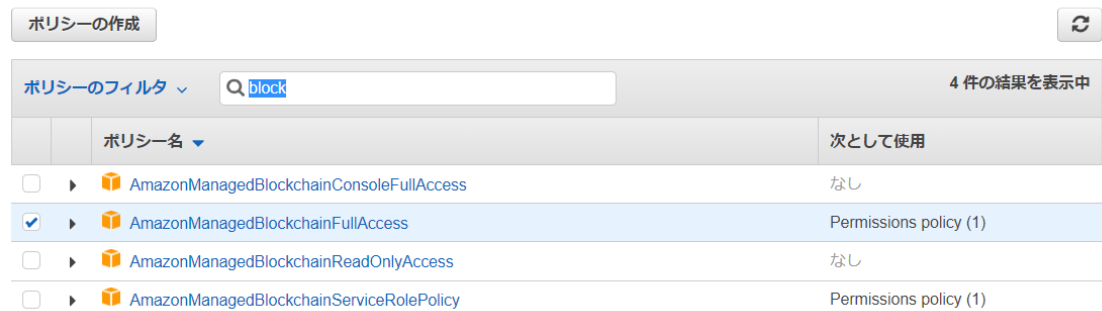
\* 必須

キャンセル

次のステップ: アクセス権限

11. 以下 2 つのロールをアタッチします

新しいロールにアタッチするポリシーを 1 つ以上選択します。



ポリシーの作成		15 件の結果を表示中
ポリシーのフィルタ		Q s3
	ポリシー名	次として使用
<input type="checkbox"/>	AmazonDMSRedshiftS3Role	なし
<input checked="" type="checkbox"/>	AmazonS3FullAccess	Permissions policy (6)
<input type="checkbox"/>	AmazonS3OutpostsFullAccess	なし
<input type="checkbox"/>	AmazonS3OutpostsReadOnlyAccess	なし
<input type="checkbox"/>	AmazonS3ReadOnlyAccess	なし

12. [次のステップ：タグ]を押します。その次のページではそのまま[次のステップ：確認]を押します
13. 適当な名前を付け、[ロールの作成]を押します。ポリシーが正しく 2 つ設定されていることを確認してください

## 確認

以下に必要な情報を指定してこのロールを見直してから、作成してください。



ロール名\* 20210407ambrole

英数字と「+,.@\_」を使用します。最大 64 文字。

ロールの説明 Allows EC2 instances to call AWS services on your behalf.

最大 1000 文字。英数字と「+,.@\_」を使用します。

信頼されたエンティティ AWS のサービス: ec2.amazonaws.com

ポリシー  AmazonManagedBlockchainFullAccess [🔗](#)  
 AmazonS3FullAccess [🔗](#)

アクセス権限の境界 アクセス権限の境界が設定されていません

追加されたタグはありません。

\* 必須

キャンセル

戻る

ロールの作成

14. 以下が表示されれば完了です



ロール **20210407ambrole** が作成されました。

ロールの作成

ロールの削除

15. ブラウザの別タブを開き EC2 の画面にいきます。画面左ペインのインスタンスをクリックし、Cloud9 用 EC2 インスタンスを特定します。Name に[cloud9]の文字列が自動で設定されているものがそれです。
16. インスタンスのチェックボックスを押し、パブリック IP が正しく設定されているか確認します



(設定されていない場合、左ペインの Elastic IP から設定してください)

## ▼ ネットワーク & セキュリティ

セキュリティグループ

New

Elastic IP New

プレースメントグループ

キーペア

ネットワークインターフ

エイス New

17. [アクション]→[セキュリティ]→[IAM ロールの変更]を選びます



18. 今日作成したロールを選択し、[保存]を押します

**IAM ロールを変更** 情報

IAM ロールをインスタンスにアタッチします。


インスタンス ID


 **i-0221620b9dda33041** (aws-cloud9-20210407-008436a4d2c4478fab18ab71f863f39)

IAM ロール

インスタンスにアタッチする IAM ロールを選択するか、まだ作成していない場合は新しいロールを作成します。選択したロールによって、現在インスタンスにアタッチされているロールが置き換えられます。

20210407ambrole ▼



[新しい IAM ロールを作成](#) 

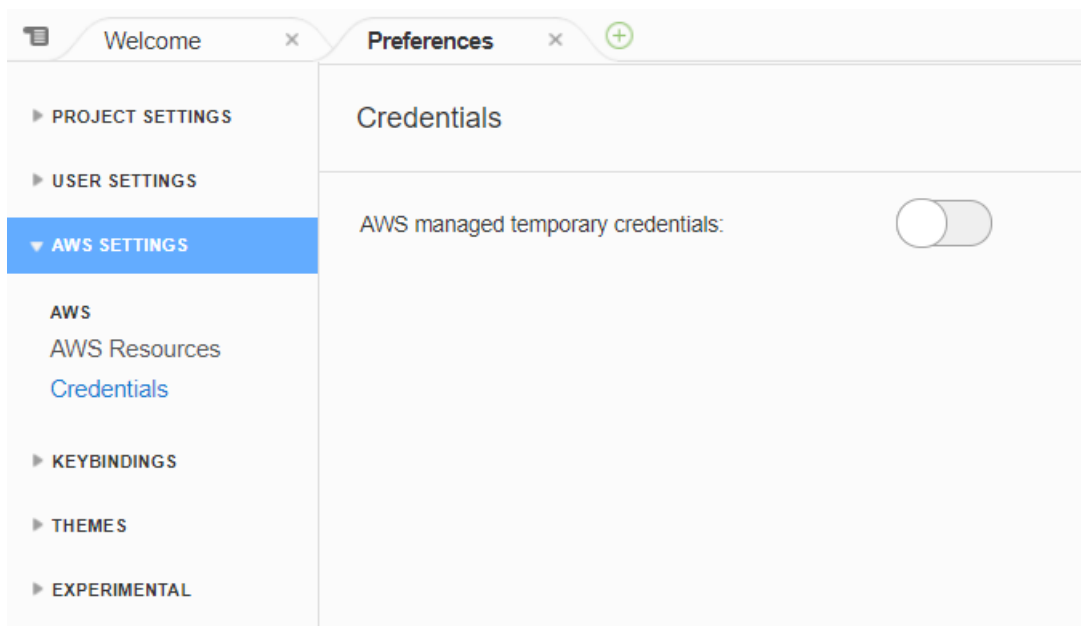
キャンセル

保存

19. Cloud9 画面右上の歯車マークを押します



20. 左ペインから[AWS SETTINGS]→[Credentials]を選び、[AWS managed temporary credentials]をオフにします



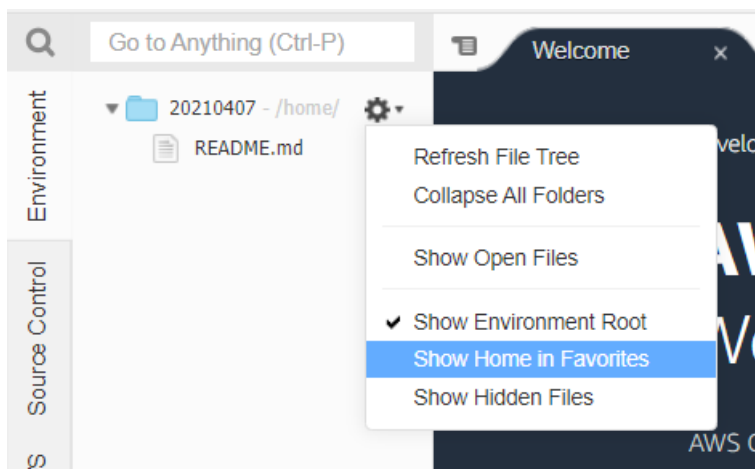
21. タブ右上の x マークでタブを閉じます
22. `aws configure` をターミナルで入力し実行します

```
ec2-user:~/environment $ aws configure
AWS Access Key ID [None]:
```

23. [Default region name]に[ap-northeast-1]と設定し、それ以外は空欄のまま、エンターをおします

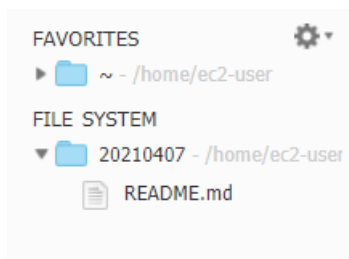
```
ec2-user:~/environment $ aws configure
AWS Access Key ID [None]:
AWS Secret Access Key [None]:
Default region name [None]: ap-northeast-1
```

24. 左ペイン、[Environment]から歯車を押し、[Show Home in Favorites]を押します



25. 以下のように EC2 のホームディレクトリが表示されます





## [Managed Blockchain]の初期設定

26. マネージメントコンソールで managed Blockchain を開きます



27. [ネットワークを作成]をおし、すべての設定はデフォルトのまま、適当な名前をつけ、[次へ]をおします

## プライベートネットワークを作成



28. メンバー名に適当な名前をつけます

## メンバーを作成 情報

### メンバー設定

Amazon Managed Blockchain ネットワークで最初のメンバーを作成します。メンバーはネットワーク内の個別のアイデンティティであり、各ネットワークには少なくとも 1 つが必要です。メンバーを作成した後、メンバーに属するピアノードを追加できます。

**メンバー名**  
ネットワーク上でこのメンバーを識別する名前を入力します。各メンバーの名前はすべてのメンバーに表示され、ネットワーク内で一意である必要があります。

メンバー名の最大長は 64 文字であり、英数字とハイフンを使用できます。先頭を数字にしたり、先頭と末尾をハイフン (-) にしたり、2 つの連続したハイフンを含めることはできません。メンバー名もネットワーク全体で一意である必要があります。

**説明 (オプション)**

説明の最大長は 128 文字です。

29. 認証機関の名前とパスワードを付けます。この 2 つは後で使うので必ずメモをお願いします。パスワードは大文字小文字数字の組み合わせが必要です

## Hyperledger Fabric 認証機関 (CA) の設定 情報

**管理者ユーザー名**  
Fabric CA 管理者ユーザーのログイン ID を定義する英数字の文字列を指定します。

管理者ユーザー名は最大 16 文字です。文字で始まる必要があり、英数字を使用できます。

**管理者パスワード**  
Fabric CA 管理者ユーザーのパスワードを定義する英数字の文字列を指定します。

☐ パスワードを表示

管理者パスワードは 8 文字以上で、少なくとも 1 つの大文字、1 つの小文字、および 1 つの数字を含める必要があります。一重引用符 (')、二重引用符 (")、スラッシュ (/)、バックスラッシュ (\)、@、またはスペースを含めることはできません。管理者パスワードの最大長は 32 文字です。

30. [次へ]をおし、次の画面で[ネットワークとメンバーの作成]を押します

ネットワークを作成しています

マネージド型ブロックチェーン > ネットワーク > 20210407network

20210407network

30 分ほど待ち時間が発生しますので、その間 Cloud9 に戻り作業を続行します  
[Cloud9 で HyperLedger Fabric クライアントの設置]

31. EC2 マネージメントコンソールで Cloud9 インスタンスを特定し、チェックボックスにチェックを押し、画面下のセキュリティグループをクリックします。(セキュリティグループ ID は後で使うのでメモをとっておきます)

インスタンス (1/1) 情報

インスタンスをフィルタリング

Name	インスタンス ID	インスタンス...	インスタ...	ステータスチェ...	アラーム
aws-cloud9-2...	i-0221620b9dda33041	実行中	t2.medium	2/2 のチェックに...	アラーム

インスタンス: i-0221620b9dda33041 (aws-cloud9-20210407-008436a4d2c4478fab71f863f39)

詳細 | **セキュリティ** | ネットワーキング | ストレージ | ステータスチェック | モニタリング | タグ

▼ セキュリティの詳細

IAM ロール

20210407ambrole

所有者 ID

294963776963

セキュリティグループ

sg-02664a9e9a92201f8 (aws-cloud9-20210407-008436a4d2c4478fab71f863f39-InstanceSecurityGroup-17XB1TV86MNUC)

32. [インバウンドルールの編集]ボタンをおします。(設定済ルールは人により異なりますので、気にする必要はありません)

インバウンドルール | アウトバウンドルール | タグ

インバウンドルール (2) インバウンドルールを編集

タイプ	プロトコル	ポート範囲	ソース	説明 - オプション
SSH	TCP	22	18.179.48.96/27	-
SSH	TCP	22	18.179.48.128/27	-

33. [ルールの追加]ボタンをおします

インバウンドルールを編集

インバウンドルールは、インスタンスに到達できる通信トラフィックをコントロールします。

インバウンドルール

タイプ: SSH | プロトコル: TCP | ポート範囲: 22 | ソース: カスタム | 説明 - オプション:

18.179.48.96/27

18.179.48.128/27

ルールを追加

34. タイプにすべての TCP、ソースに先ほどメモをしたセキュリティグループ ID (つま

り自分自身です) を選びます



The screenshot shows the AWS IAM console interface for adding a rule to a security group. A dropdown menu is set to 'すべてのTCP' (All TCP). Below it, there are tabs for 'TCP', 'UDP', and 'ICMP'. A search bar contains the text '0-65535'. To the right, there is a 'カスタム' (Custom) button and a search bar with 'sg-02664a9e9a92201f8' entered. A '削除' (Delete) button is also visible. At the bottom, there is a 'ルールを追加' (Add rule) button.

これにより、同じセキュリティグループを持つインターフェースからの通信が許可されるようになります

35. [ルールを保存]をおします



36. Cloud9 コンソールで以下を実行します

```
sudo yum update -y
```

37. 以下を実行します

```
sudo yum install jq telnet emacs docker libtool libtool-ltdl-devel git -y
```

38. 以下を実行します

```
sudo service docker start
```

39. 以下を実行します

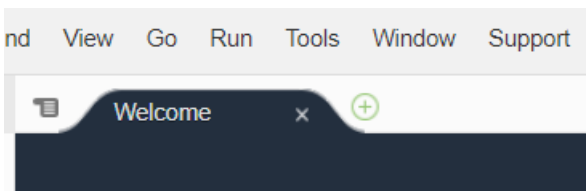
```
sudo usermod -a -G docker ec2-user
```

40. df -h を実行すると、ディスク残り容量が少ないことがわかります。

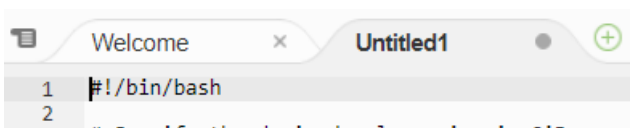
```
ec2-user:~/environment $ df -h
Filesystem      Size  Used Avail Use% Mounted on
devtmpfs        2.0G   0    2.0G   0% /dev
tmpfs           2.0G   0    2.0G   0% /dev/shm
tmpfs           2.0G  520K    2.0G   1% /run
tmpfs           2.0G   0    2.0G   0% /sys/fs/cgroup
/dev/xvda1      10G   8.3G   1.8G  83% /
tmpfs           395M   0    395M   0% /run/user/1000
tmpfs           395M   0    395M   0% /run/user/0
```

41. [resize.sh]を今日の資料フォルダや git から開き内容をコピーします。

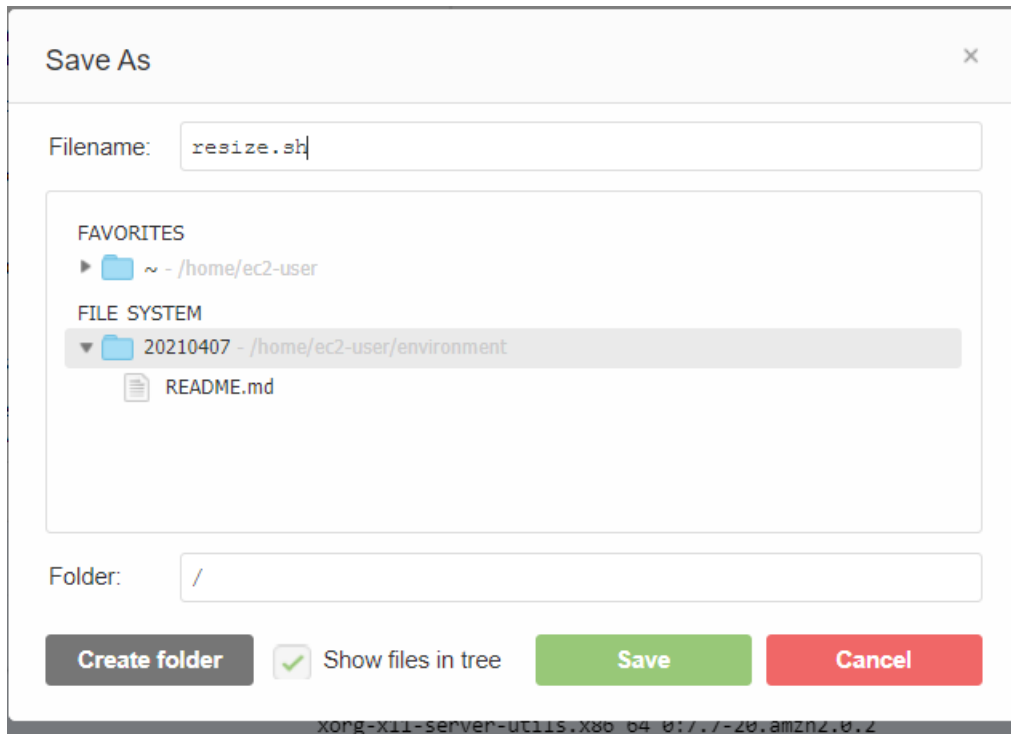
42. Cloud9 上部画面のプラスボタンをおして [New File] を選びます



43. 先ほどの内容を貼り付け、タブの黒丸を押すと保存画面がでできます。



44. [resize.sh]と名前を付けて [Save]をおします



45. 先ほど作成した IAM ロールに以下 2 つの権限を追加で付与します

[AmazonEC2FullAccess](#)

[AWSCloud9Administrator](#)



46. 2-3 分待って、以下を実行します

```
sh resize.sh 30
```

47. df -h の出力が以下のようなになれば成功です

```

ec2-user:~/environment $ df -h
Filesystem      Size  Used Avail Use% Mounted on
devtmpfs        2.0G   0    2.0G   0% /dev
tmpfs           2.0G   0    2.0G   0% /dev/shm
tmpfs           2.0G 464K    2.0G   1% /run
tmpfs           2.0G   0    2.0G   0% /sys/fs/cgroup
/dev/xvda1      30G   8.3G    22G  28% /
tmpfs           395M   0    395M   0% /run/user/1000
ec2-user:~/environment $

```

48. EC2 の画面から、一度 Cloud9 インスタンスを再起動し、再度ターミナル画面に繋がります
49. 以下を実行します
 

```

sudo curl -L ¥
https://github.com/docker/compose/releases/download/1.20.0/docker-compose-
`uname ¥
-s` -`uname -m` -o /usr/local/bin/docker-compose

```
50. 以下を実行します
 

```

sudo chmod a+x /usr/local/bin/docker-compose

```
51. 以下を実行します
 

```

wget https://dl.google.com/go/go1.14.4.linux-amd64.tar.gz

```
52. 以下を実行します
 

```

tar -xzf go1.14.4.linux-amd64.tar.gz

```
53. 以下を実行します
 

```

sudo mv go /usr/local

```
54. 以下を実行します
 

```

sudo yum install git -y

```

(Nothing to do と表示される場合がありますが、気にせずすすめます)
55. シナリオアセットの[bash\_profile]を開きます。
56. そろそろ blockchain の構築が終わっているので、ブラウザでステータスを確認します。利用可能となっていれば構築完了です。まだの方は利用可能となるまで待ちます。定期的にブラウザをリロードしてみてください

マネージドブロックチェーン > ネットワーク > 20210407network

## 20210407network

[詳細](#) | [メンバー](#) | [提案](#)

**詳細**

ネットワークID n-K5X6D4O4GRFDDG535UV3VNJ3TM	説明 -	ARN arn:aws:managedblockchain:ap-northeast-1::networks/n-K5X6D4O4GRFDDG535UV3VNJ3TM
投票ポリシー Greater than 50%	ステータス 🟢 利用可能	作成済み 2021年4月7日(水)
提案期間 24 時間	VPC エンドポイントサービス名 <a href="#">情報</a> com.amazonaws.ap-northeast-1.managedblockchain.n-k5x6d4o4grfddg535uv3vnj3tm	フレームワーク Hyperledger Fabric 1.4
サービスエンドポイントの注文 <a href="#">情報</a> orderer.n-k5x6d4o4grfddg535uv3vnj3tm.managedblockchain.ap-northeast-1.amazonaws.com:30001	ネットワークエディション スターター	アクティブな提案 0
メンバー 1		

VPC エンドポイントを作成

57. [bash\_profile]の[MyMemberCaEndpoint]をメンバー詳細画面で表示される[Fabric 認証機関エンドポイント]の値に置換します。同様に[MyNetworkOrdererEndpoint]を[サービスエンドポイントの注文]の値に置換します
58. Cloud9 で以下を実行します  

```
vi ~/.bash_profile
```
59. 小文字の[d]を押して、全部消します。その後カーソルが一番左上にあることを確認し、[a]をおし（画面下に INSERT と表示されます）、修正した bash\_profile の中身をコピペします。
60. esc キーを押した後、[:wq]と入力します
61. 以下を実行します  

```
source ~/.bash_profile
```
62. 以下を実行します  

```
sudo docker version
```
63. 以下を実行します  

```
sudo /usr/local/bin/docker-compose version
```
64. 以下を実行します  

```
go version
```
- [Managed Blockchain の VPC エンドポイント作成]
65. Managed blockchain のマネージメントコンソールから、ネットワークの詳細画面を開き、[VPC エンドポイントを作成]ボタンをおします

マネージドブロックチェーン > ネットワーク > 20210407network

## 20210407network

詳細 | メンバー | 提案

詳細		
ネットワーク ID n-K5X6D4O4GRFDDG535UV3VNJ3TM	説明 -	ARN arn:aws:managedblockchain:ap-northeast-1::networks/n-K5X6D4O4GRFDDG535UV3VNJ3TM
投票ポリシー Greater than 50%	ステータス 🟢 利用可能	作成済み 2021年4月7日(水)

VPC エンドポイントを作成

66. Cloud9 を構築した、VPC,サブネットを指定し、先ほどメモした同じセキュリティグループを指定し、[作成]を押します。

### VPC エンドポイントを作成

VPC エンドポイントサービス名  
com.amazonaws.ap-northeast-1.managedblockchain.n-k5x6d4o4grfddg535uv3vnj3tm

VPC  
vpc-aa1644cf

サブネット  
サブネットを選択

subnet-1e2d7d47  
パブリックサブネット ap-northeast-1c

セキュリティグループ  
セキュリティグループを選択

sg-02664a9e9a92201f8  
aws-cloud9-20210407-008436a4d2c4478fab71f863f39-InstanceSecurityGroup-17XB1TV86MNUC

プライベート DNS 名  
☒ このエンドポイントに対して有効にする

キャンセル 作成

これにより、Cloud9 と VPC 内部で通信が行えるようになりました

67. アセットから[get\_member]を開き、[network-id],[member-id]をそれぞれ、マネージメントコンソールで表示されている ID に置き換えます。以下のようなになるはずで



す。なお、この2つのIDもどこかにメモしておいた方が便利です

```
get_member.txt - メモ帳
ファイル(E) 編集(E) 書式(O) 表示(V) ヘルプ(H)

aws managedblockchain get-member ¥
--network-id n-K5X6D404GRFDDG535UV3VNJ3TM ¥
--member-id m-76SGD6FQMVDKJP5YSXOPBPXH2I|
```

68. 全体をコピーし、Cloud9 で実行します。以下のような表示になれば成功です

```
{
  "Status": "AVAILABLE",
  "Name": "hkameda",
  "Tags": {},
  "FrameworkAttributes": {
    "Fabric": {
      "AdminUsername": "hkameda",
      "CaEndpoint": "ca.m-76sgd6fqmvdckjp5ysxopbpxh2i.n-k5x6d4o4grfddg535uv3vnj3tm.managedblockchain.ap-northeast-1"
    }
  },
  "LogPublishingConfiguration": {
    "Fabric": {
      "CaLogs": {
        "Cloudwatch": {
          "Enabled": false
        }
      }
    }
  },
  "CreationDate": "2021-04-07T01:22:58.480Z",
  "Id": "m-76SGD6FQMVDKJP5YSXOPBPXH2I",
  "Arn": "arn:aws:managedblockchain:ap-northeast-1:294963776963:members/m-76SGD6FQMVDKJP5YSXOPBPXH2I"
}
```

ec2-user:~/environment \$

69. 以下を実行します

```
curl https://$CASERVICEENDPOINT/cainfo -k
```

出力の最後に[true]と表示されれば成功です

70. 以下を実行します

```
mkdir -p /home/ec2-user/go/src/github.com/hyperledger/fabric-ca
```

71. 以下を実行します

```
cd /home/ec2-user/go/src/github.com/hyperledger/fabric-ca
```

72. 以下を実行します

```
wget https://github.com/hyperledger/fabric-ca/releases/download/v1.4.7/hyperledger-fabric-ca-linux-amd64-1.4.7.tar.gz
```

73. 以下を実行します

```
tar -xzf hyperledger-fabric-ca-linux-amd64-1.4.7.tar.gz
```

74. 以下を実行します

```
cd /home/ec2-user
```

75. 以下を実行します

```
git clone --branch v1.4.7 https://github.com/hyperledger/fabric-samples.git
```

## [Managed Blockchain で Peer Note を作成]

76. マネージメントコンソールの Blockchain で、メンバーの詳細画面にいき[ピアノードを作成]ボタンをおします



77. 全てデフォルトのまま[ピアノードを作成]ボタンをおします



78. 作成中となります。作成中でもノード ID をクリックすると詳細が出てきますので [ピアエンドポイント] の値をメモします



79. アセットから [docker-compose-cli.yaml] を開きます

80. [MyMemberID], [MyPeerNodeEndpoint] の値をそれぞれ置き換えます。以下のように  
なります

```

image: hyperledger/fabric-tools:1.4
tty: true
environment:
  - GOPATH=/opt/gopath
  - CORE_VM_ENDPOINT=unix:///host/var/run/docker.sock
  - FABRIC_LOGGING_SPEC=info # Set logging level to debug for more verbose logging
  - CORE_PEER_ID=cli
  - CORE_CHAINCODE_KEEPALIVE=10
  - CORE_PEER_TLS_ENABLED=true
  - CORE_PEER_TLS_ROOTCERT_FILE=/opt/home/managedblockchain-tls-chain.pem
  - CORE_PEER_LOCALMSPID=m-76SGD6FQMVDKJP5YSXOPBPXH2I
  - CORE_PEER_MSPCONFIGPATH=/opt/home/admin-msp
  - CORE_PEER_ADDRESS=nd-eyt7l3b56zh7lcuehykf6g3l2u.m-76sgd6fqmvdjkp5sysxopbp2i.n-k5x6d4o4grfddg535uv3vnj3tm.managedblockchain.ap-northeast-1.amazonaws.com:30003
working_dir: /opt/gopath/src/github.com/hyperledger/fabric/peer
command: /bin/bash

```

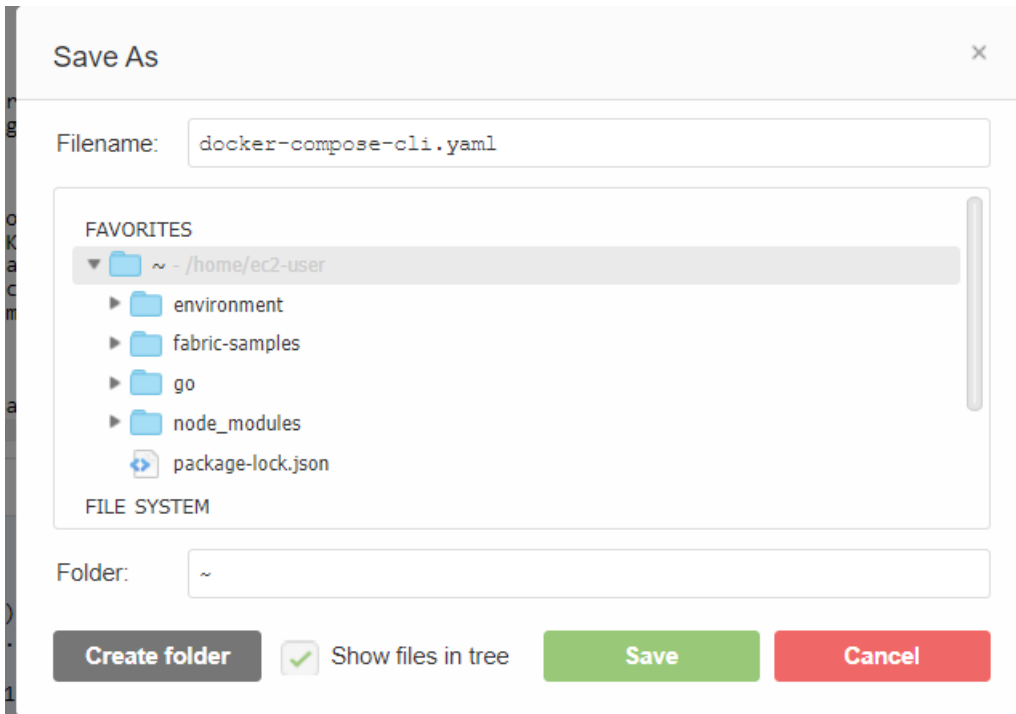
81. 今日すでに作業をしたように、Cloud9 で New File を作成し、上記で修正した内容をペーストします。

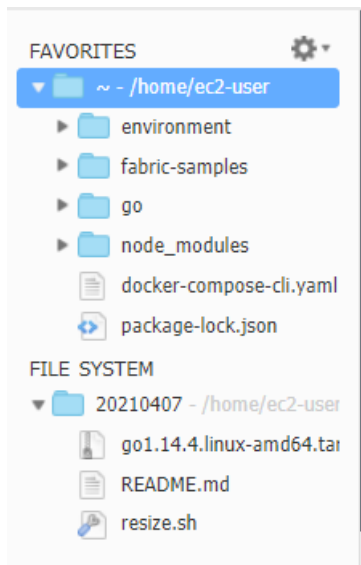
```

1 version: '2'
2 services:
3   cli:
4     container_name: cli
5     image: hyperledger/fabric-tools:1.4
6     tty: true
7     environment:
8       - GOPATH=/opt/gopath
9       - CORE_VM_ENDPOINT=unix:///host/var/run/docker.sock
10      - FABRIC_LOGGING_SPEC=info # Set logging level to debug for more verbose logging
11      - CORE_PEER_ID=cli
12      - CORE_CHAINCODE_KEEPALIVE=10
13      - CORE_PEER_TLS_ENABLED=true
14      - CORE_PEER_TLS_ROOTCERT_FILE=/opt/home/managedblockchain-tls-chain.pem
15      - CORE_PEER_LOCALMSPID=m-76SGD6FQMVDKJP5YSXOPBPXH2I
16      - CORE_PEER_MSPCONFIGPATH=/opt/home/admin-msp
17      - CORE_PEER_ADDRESS=nd-eyt7l3b56zh7lcuehykf6g3l2u.m-76sgd6fqmvdjkp5sysxopbp2i.n-k5x6d4o4grfddg535uv3vnj3tm.managedblockchain.ap-northeast-1.amazonaws.com:30003
18     working_dir: /opt/gopath/src/github.com/hyperledger/fabric/peer
19     command: /bin/bash
20   volumes:
21     - /var/run/:/host/var/run/
22     - /home/ec2-user/fabric-samples/chaincode:/opt/gopath/src/github.com/
23     - /home/ec2-user:/opt/home

```

82. [docker-compose-cli.yaml] のファイル名で保存します。保存先は[ec2-user]の直下であることを注意してください





83. 以下を実行します

```
docker-compose -f docker-compose-cli.yaml up -d
```

正しく設定されていると以下のような出力となります

```
ec2-user:~ $ docker-compose -f docker-compose-cli.yaml up -d
Creating network "ec2user_default" with the default driver
Pulling cli (hyperledger/fabric-tools:1.4)...
1.4: Pulling from hyperledger/fabric-tools
7ddbc47eeb70: Pull complete
c1bbdc448b72: Pull complete
8c3b70e39044: Pull complete
45d437916d57: Pull complete
b5035666b1cd: Pull complete
94c898b5fdef: Pull complete
260e3fcb1baf: Pull complete
554fde936dba: Pull complete
139c82a7cff9: Pull complete
190894457765: Pull complete
5b3ca7222c1b: Pull complete
Digest: sha256:c9d93b746dfcfbf9755a28d4073588288d4c89ad20bf7a473747efad3e252cc8
Status: Downloaded newer image for hyperledger/fabric-tools:1.4
Creating cli ... done
```

84. 以下を実行します

```
sudo /usr/local/bin/docker-compose -f docker-compose-cli.yaml up -d
```

#### [証明書の設定]

85. 今までの手順で Cloud9 がネットワークに参加しているメンバーのクライアントとして設定され、ブロックチェーン処理を行うためのコンテナが作成されました。ただしネットワークに処理を流すためには、証明書が正しく設定されなければ認証されないため、設定を行う必要があります。以下を実行します

```
aws s3 cp s3://ap-northeast-1.managedblockchain/etc/managedblockchain-tls-chain.pem /home/ec2-user/managedblockchain-tls-chain.pem
```

(別リージョンで作業している場合は、ap-northeast-1 を書き換えてください)

86. 以下を実行します

```
openssl x509 -noout -text -in /home/ec2-user/managedblockchain-tls-chain.pem
```

87. アセットから[fabric-ca-client]を開き、[AdminUsername],[AdminPassword]を今日設定した ID とパスワードに置換し、さらに、[\$CASERVICEENDPOINT]を[Fabric 認証機関エンドポイント]の値に置き換えます。(ドルマーク毎置き換えてください)

88. 置き換えが完了したら全体をコピーし、Cloud9 で実行します

```
ec2-user~$ fabric-ca-client enroll \
> -u 'https://kameda:Harukame0707@ca.m-76sgd6fqmvdjkjp5ysx0pbpxh2i.n-k5xd404grfdg535uv3vnj3tm.managedblockchain.ap-northeast-1.amazonaws.com:30002' \
> --tls.certfiles /home/ec2-user/managedblockchain-tls-chain.pem -M /home/ec2-user/admin-msp
2021/04/07 02:44:43 [INFO] Created a default configuration file at /home/ec2-user/.fabric-ca-client/fabric-ca-client-config.yaml
2021/04/07 02:44:43 [INFO] TLS Enabled
2021/04/07 02:44:43 [INFO] generating key: &[A:ecdsa S:256]
2021/04/07 02:44:43 [INFO] encoded CSR
2021/04/07 02:44:44 [INFO] Stored client certificate at /home/ec2-user/admin-msp/signcerts/cert.pem
2021/04/07 02:44:44 [INFO] Stored root CA certificate at /home/ec2-user/admin-msp/cacerts/ca-m-76sgd6fqmvdjkjp5ysx0pbpxh2i.n-k5xd404grfdg535uv3vnj3tm.managedblockchain.ap-northeast-1.amazonaws.com-30002.pem
2021/04/07 02:44:44 [INFO] Stored Issuer public key at /home/ec2-user/admin-msp/IssuerPublicKey
2021/04/07 02:44:44 [INFO] Stored Issuer revocation public key at /home/ec2-user/admin-msp/IssuerRevocationPublicKey
```

89. 以下を実行します

```
cp -r /home/ec2-user/admin-msp/signcerts admin-msp/admincerts
```

#### [チャネルの作成]

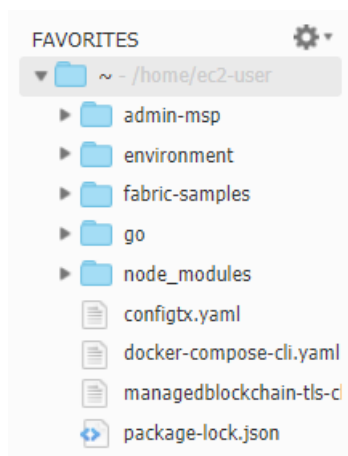
Hyperledger Fabric では、元帳はチャネルで管理されます。すべてのメンバーが共通のチャネルで操作する場合、元帳はネットワーク全体で共有できます。

90. アセットから[configtx.yaml]を開きます

91. 2 か所の[MemberID]を置換します

```
Organizations:
- &Org1
  # member id defines the organization
  Name: m-76SGD6FQMVDKJP5YSX0BPBXH2I
  # ID to load the MSP definition as
  ID: m-76SGD6FQMVDKJP5YSX0BPBXH2I
  #msp.dir.of.org1 in the docker container
```

92. Cloud9 で New File を作成し中身を全部コピーし以下のようにファイルを作成します



93. 以下のコマンドを実行します

```
docker exec cli configtxgen ¥
```

```
-outputCreateChannelTx /opt/home/mychannel.pb ¥
-profile OneOrgChannel -channelID mychannel ¥
--configPath /opt/home/
```

以下のように表示がされれば成功です。(Warning は新しいチャネル作成をしていない、というただの表示なので気にせず進めます)

```
ec2-user:~$ docker exec cli configtxgen \
> -outputCreateChannelTx /opt/home/mychannel.pb \
> -profile OneOrgChannel -channelID mychannel \
> --configPath /opt/home/
2021-04-07 02:54:07.879 UTC [common.tools.configtxgen] main -> INFO 001 Loading configuration
2021-04-07 02:54:07.881 UTC [common.tools.configtxgen.localconfig] Load -> INFO 002 Loaded configuration: /opt/home/configtx.yaml
2021-04-07 02:54:07.883 UTC [common.tools.configtxgen.localconfig] LoadProfile -> INFO 003 Loaded configuration: /opt/home/configtx.yaml
2021-04-07 02:54:07.883 UTC [common.tools.configtxgen] doOutputChannelCreateTx -> INFO 004 Generating new channel configtx
2021-04-07 02:54:07.883 UTC [common.tools.configtxgen.encoder] NewChannelGroup -> WARN 005 Default policy emission is deprecated, please include policy specifications for the channel group in configtx.yaml
2021-04-07 02:54:07.883 UTC [common.tools.configtxgen.encoder] NewApplicationGroup -> WARN 006 Default policy emission is deprecated, please include policy specifications for the application group in configtx.yaml
2021-04-07 02:54:07.884 UTC [common.tools.configtxgen.encoder] NewChannelGroup -> WARN 007 Default policy emission is deprecated, please include policy specifications for the channel group in configtx.yaml
2021-04-07 02:54:07.884 UTC [common.tools.configtxgen.encoder] NewApplicationGroup -> WARN 008 Default policy emission is deprecated, please include policy specifications for the application group in configtx.yaml
2021-04-07 02:54:07.884 UTC [common.tools.configtxgen.encoder] NewApplicationGroup -> WARN 009 Default policy emission is deprecated, please include policy specifications for the application group in configtx.yaml
2021-04-07 02:54:07.884 UTC [common.tools.configtxgen.encoder] NewApplicationGroup -> WARN 00a Default policy emission is deprecated, please include policy specifications for the application group in configtx.yaml
2021-04-07 02:54:07.884 UTC [common.tools.configtxgen] doOutputChannelCreateTx -> INFO 00b Writing new channel tx
```

94. 以下の部分の置き換え、Cloud9 で実行します

export ORDERER=<サービスエンドポイントの注文、の値>

95. 以下を実行します

```
docker exec cli peer channel create -c mychannel ¥
-f /opt/home/mychannel.pb -o $ORDERER ¥
--cafile /opt/home/managedblockchain-tls-chain.pem -tls
```

96. 以下を実行します

```
docker exec cli peer channel join -b mychannel.block ¥
-o $ORDERER --cafile /opt/home/managedblockchain-tls-chain.pem -tls
```

### [Chaincode]の実行

chaincode は、Hyperledger Fabric 上におけるいわゆる SmartContract とされる部分のことで、台帳への読み書き操作をつかさどるモジュールです

97. 以下を実行します

```
docker exec cli peer chaincode install -n mycc -v v0 -p
github.com/chaincode_example02/go
```

ここまでの手順が正しければ、以下のような表示になります。

```
ec2-user:~$ docker exec cli peer chaincode install -n mycc -v v0 -p github.com/chaincode_example02/go
2021-04-07 03:00:14.462 UTC [chaincodeCmd] checkChaincodeCmdParams -> INFO 001 Using default escc
2021-04-07 03:00:14.462 UTC [chaincodeCmd] checkChaincodeCmdParams -> INFO 002 Using default vscc
2021-04-07 03:00:14.641 UTC [chaincodeCmd] install -> INFO 003 Installed remotely response:<status:200 payload:"OK" >
```

98. 以下を実行します

```
docker exec cli peer chaincode instantiate ¥
-o $ORDERER -C mychannel -n mycc -v v0 ¥
-c '{"Args":["init","a","100","b","200"]}' ¥
--cafile /opt/home/managedblockchain-tls-chain.pem -tls
(実行完了には数分待ちます)
```

99. 以下を実行します

```
docker exec cli peer chaincode list --instantiated ¥
```

```
-o $ORDERER -C mychannel ¥
```

```
--cafile /opt/home/managedblockchain-tls-chain.pem --tls
```

100. いよいよトランザクションをテストで流してみます。以下を実行します

```
docker exec cli peer chaincode query -C mychannel ¥
```

```
-n mycc -c '{"Args":["query","a"]}'
```

```
ec2-user:~ $ docker exec cli peer chaincode list --instantiated \  
> -o $ORDERER -C mychannel \  
> --cafile /opt/home/managedblockchain-tls-chain.pem --tls  
Get instantiated chaincodes on channel mychannel:  
Name: mycc, Version: v0, Path: github.com/chaincode_example02/go, Escc: escc, Vscc: vscc  
ec2-user:~ $ docker exec cli peer chaincode query -C mychannel \  
> -n mycc -c '{"Args":["query","a"]}'  
100
```

101. 今度は 100 という値から 10 を引くコマンドを実行します

```
docker exec cli peer chaincode invoke -C mychannel ¥
```

```
-n mycc -c '{"Args":["invoke","a","b","10"]}' ¥
```

```
-o $ORDERER --cafile /opt/home/managedblockchain-tls-chain.pem --tls
```

102. 以下を実行します

```
docker exec cli peer chaincode query -C mychannel ¥
```

```
-n mycc -c '{"Args":["query","a"]}'
```

```
2021-04-07 03:04:56.297 UTC [chaincodeCmd] chaincodeInvokeOrQuery -> INFO 001 Chaincode invoke successful. result: status:200  
ec2-user:~ $ docker exec cli peer chaincode query -C mychannel \  
> -n mycc -c '{"Args":["query","a"]}'  
90
```

### [Next Step]

現在はメンバーが 1 人しかいませんが、異なる AWS アカウントで同じ手順を行うことで、メンバーを増やすことが可能です。環境の準備が可能な方は、こちらを参考に挑戦してみてください。英語ですが、実際の物流を想定したシナリオになっています。

<https://track-and-trace-blockchain.workshop.aws/>

おつかれさまでした！

後片付けですが、以下の削除をお願いします

- Cloud9 の画面から Cloud9 インスタンス
- Managed Blockchain のピアノード削除
- Managed Blockchain のメンバー削除
- Managed Blockchain のネットワーク削除