【DH-1】

ブロックチェーン・アプリ開発入門

日本アイ・ビー・エム (株) 東京基礎研究所
FSS & ブロックチェーン・ソリューションズ
齋藤 新

アンケート

「ブロックチェーンのアプリを
書いたことがありますか？」

アンケート

ブロックチェーンのアプリを
書いたことがありますか？

- ある (Hyperledger Fabric/Composer)
- ある (その他)
- ないが、ブロックチェーンには詳しい
- ないが、ブロックチェーンは少し知っている
- なくて、ブロックチェーンもほぼ知らない

このセッションについて

目的

- ブロックチェーン技術およびHyperledger Fabricの基礎知識の習得
- Hyperledger Composerアプリ開発の基礎知識の習得

齋藤 新 (Shin Saito)

IBM東京基礎研究所 研究員

- 研究分野: プログラム言語理論・形式検証
- 分散台帳技術・スマートコントラクトの開発・検証手法の研究に従事
- 各種ブロックチェーンプロジェクトのテクニカル・リード/アーキテクト

GitHub/Qiita: shinsa82

日本百名山 大菩薩嶺

2018-06-13 発売

ブロックチェーンの革新技術:
Hyperledger Fabricによる
アプリケーション開発
(リックテレコム)

http://www.ric.co.jp/book/
contents/book_1146.html

Hyperledger Fabricの設定・スマートコントラクト
開発など詳細に解説

think Japan

このセッションについて

こんな感じの

Hyperledger Composerで遊んでみた

このセッションについて

記事を

Hyperledger Composerアプリを
インストール不要で開発

このセッションについて

書けるように

爆速Hyperledger Composer開発ガイド

ブロックチェーンとは？

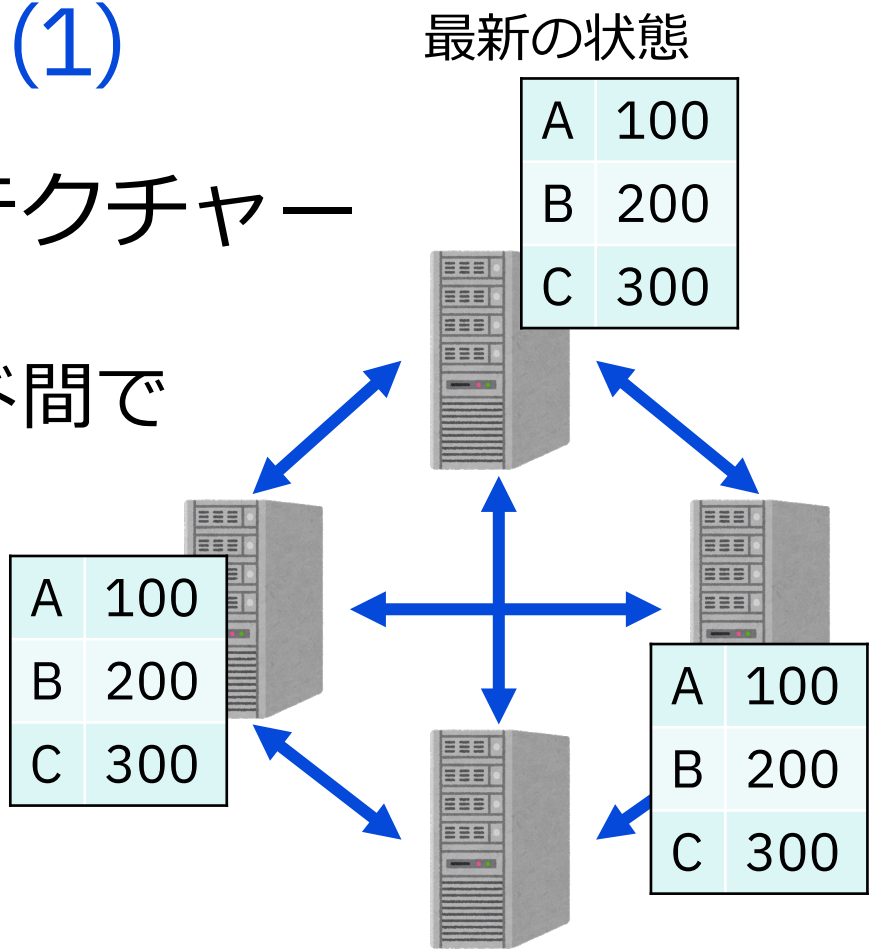
(分散台帳技術)

ブロックチェーンとは (1)

分散台帳技術の1アーキテクチャー

- **「台帳」**のコピーをノード間で共有・同期する技術
- **非集中型**の共有

- **耐障害性・耐改竄性**に優れる

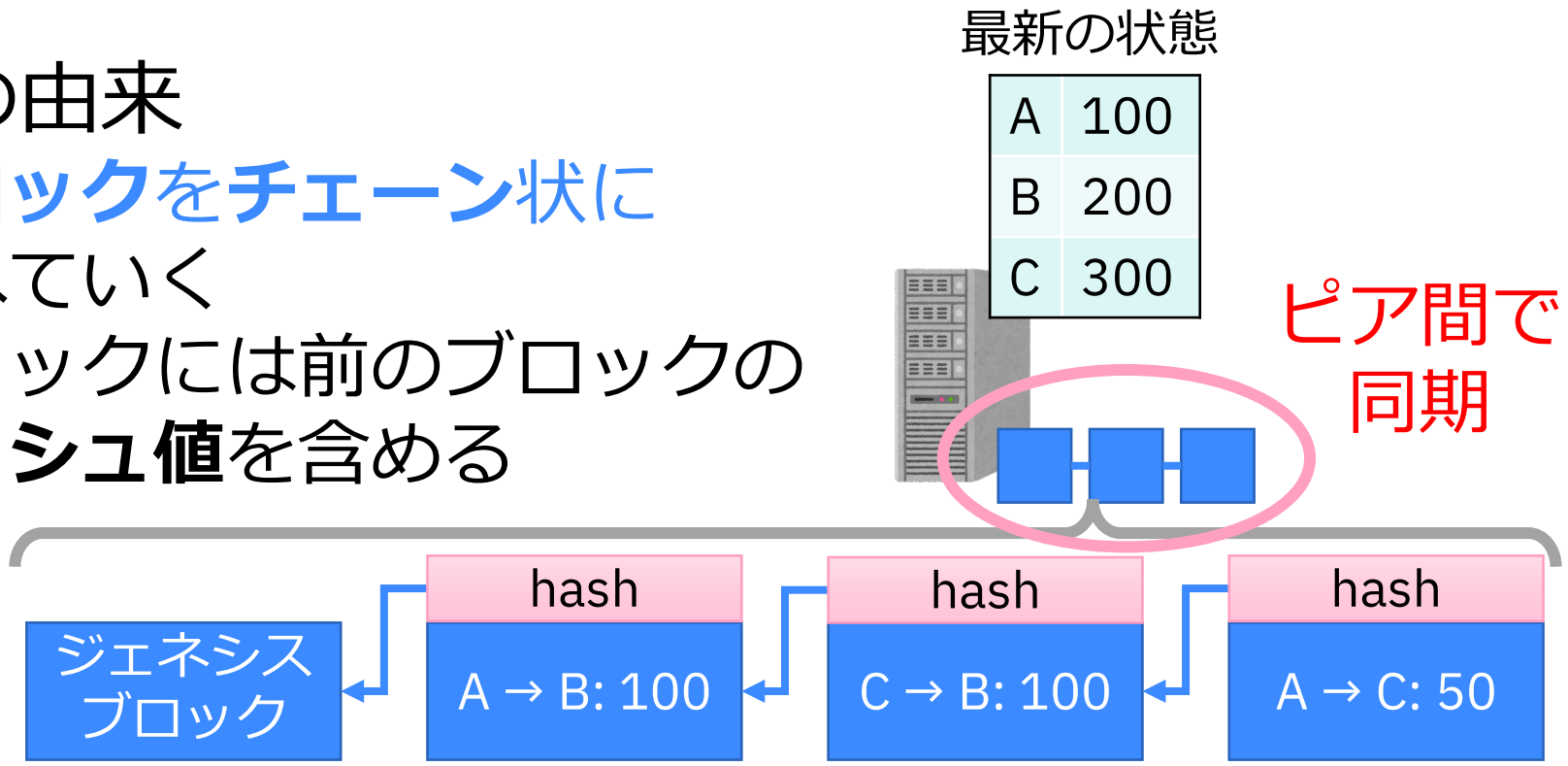


画像：いらすとや

ブロックチェーンとは (2)

名前の由来

- **ブロック**を**チェーン**状に並べていく
- ブロックには前のブロックの**ハッシュ値**を含める



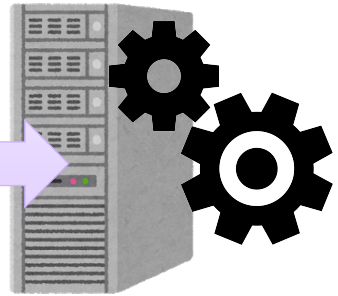
ブロックチェーンとは (3)

スマートコントラクト

- ビジネスロジックの実行

融資
(A ← B, 100, 10%/10日)

(1) トランザクション発行



(2) スマート
コントラクト
実行

A	0
B	200
C	300

A	**90**
B	**110**
C	300

(3) 状態変化

画像：無料のシルエットイラスト アイコン素材

ブロックチェーンの2タイプ (1)

	パブリック型	コンソーシアム型 プライベート型
代表例	ビットコイン イーサリアム	**Hyperledger Fabric** Corda
参加者	不特定	特定
非集中性	高い	やや低い

ブロックチェーンの2タイプ (2)

	パブリック型	コンソーシアム型 プライベート型
代表例	ビットコイン イーサリアム	**Hyperledger Fabric** Corda
コンセンサス (分散合意)	計算量	多数決
速度	遅い	速い
ファイナリティ	ない	ある

Hyperledger Fabric

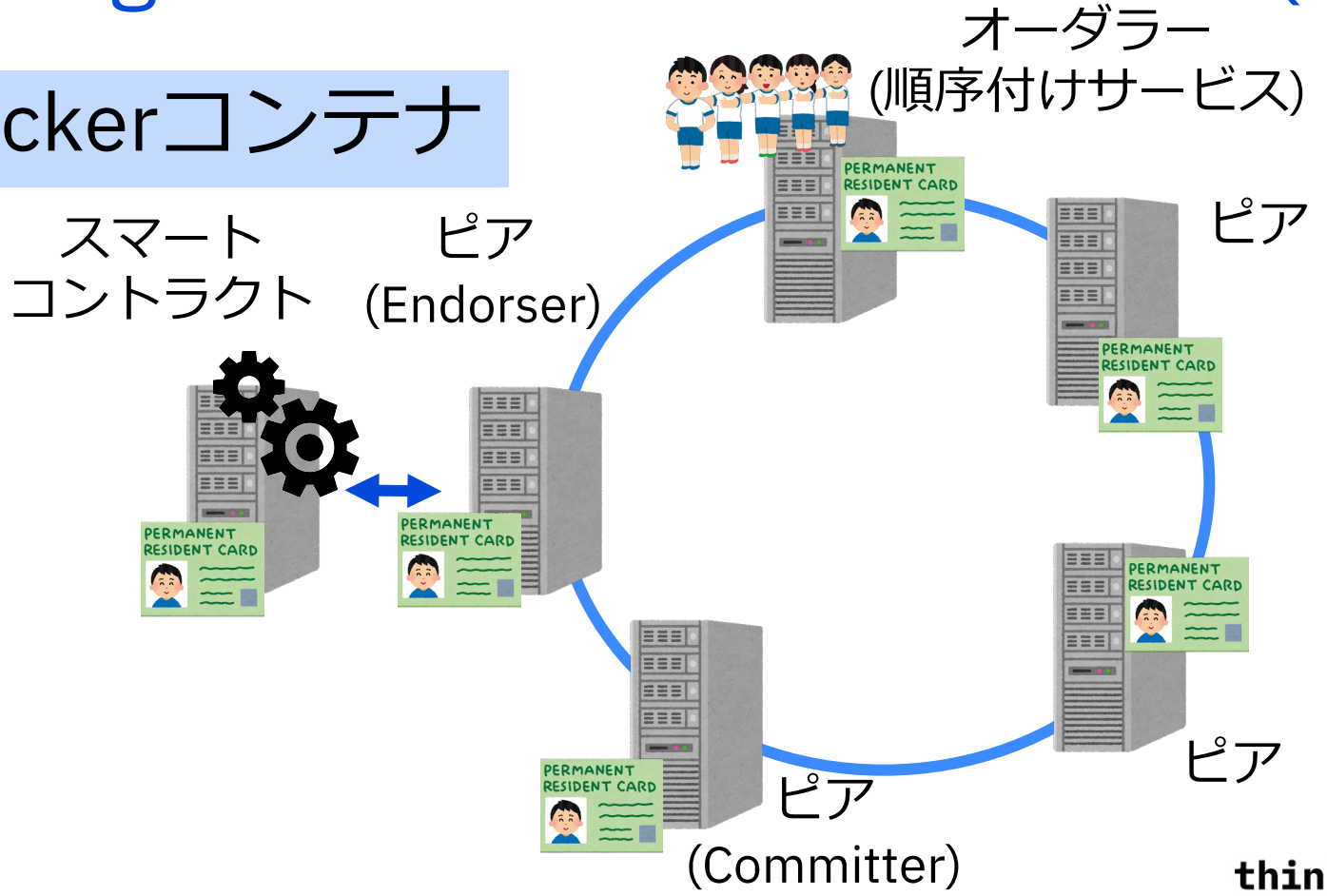
Hyperledgerコンソーシアムの1プロジェクト

ビジネス・アプリケーションに特化

- コンソーシアム型実装
- メンバーシップの柔軟な管理
- 柔軟な設定・プラグイン構造
- CouchDBを用いたリッチな問い合わせ機能
- 並列化による処理速度向上

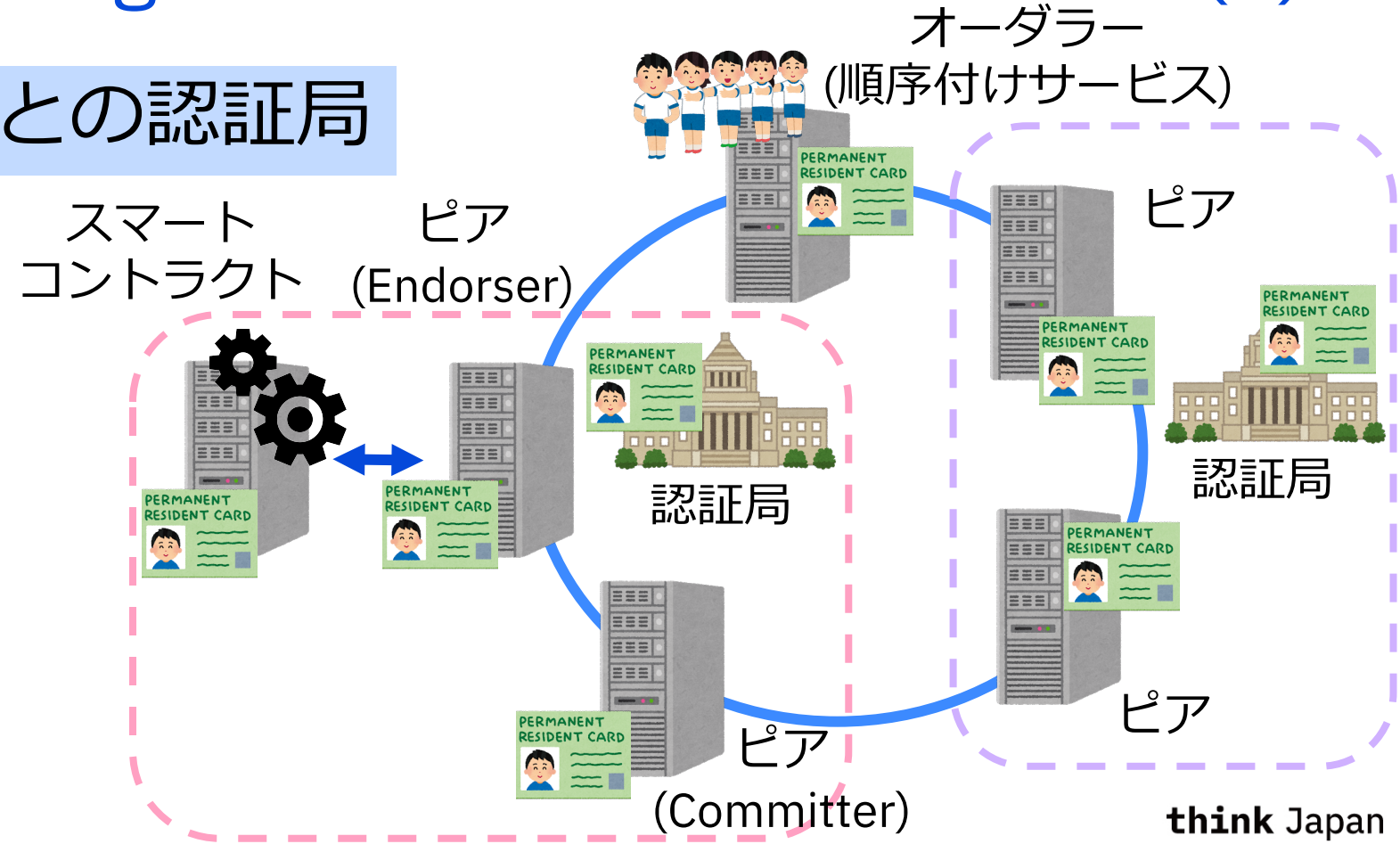
Hyperledger Fabricアーキテクチャー (1)

全部Dockerコンテナ

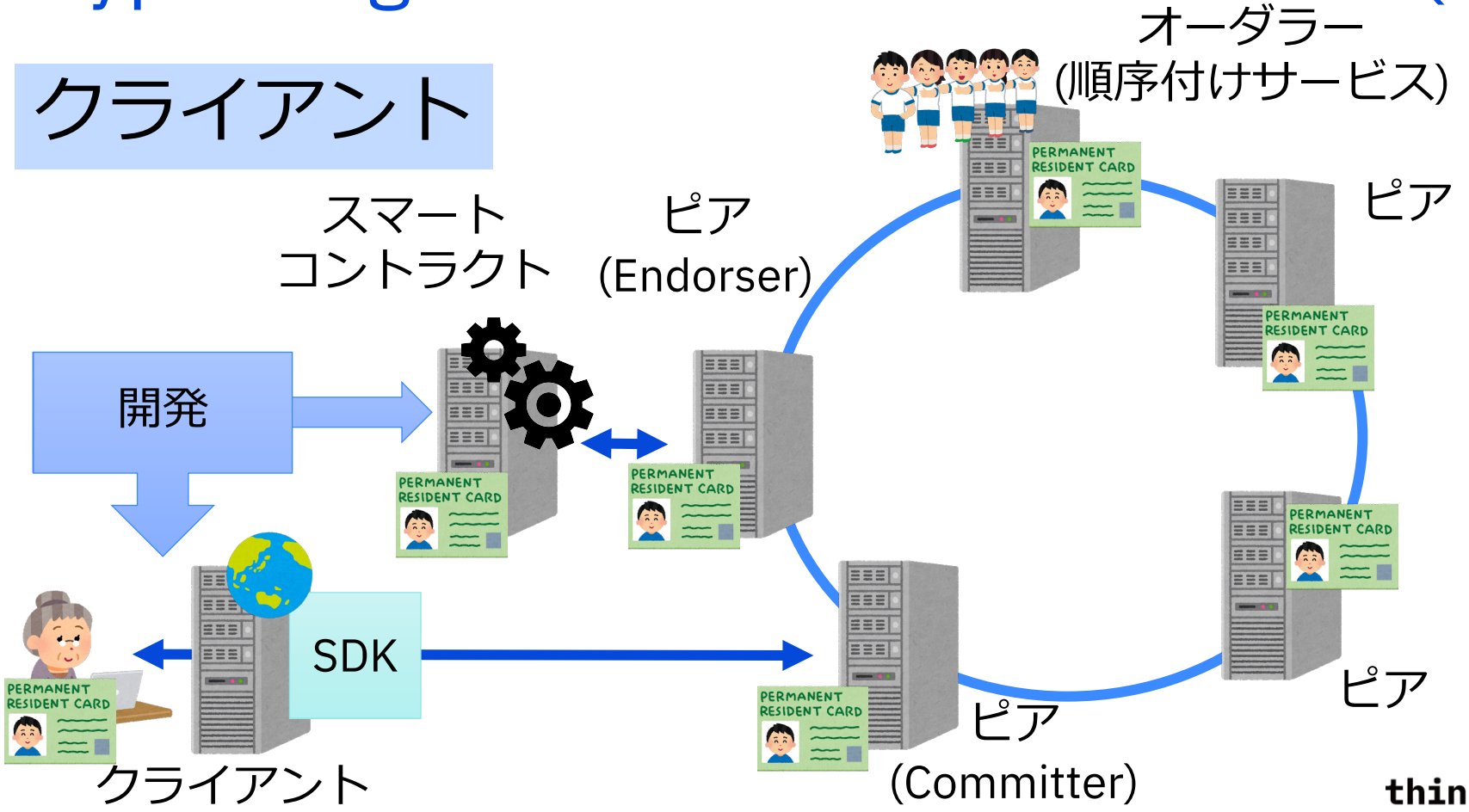


Hyperledger Fabricアーキテクチャー (2)

組織ごとの認証局



Hyperledger Fabricアーキテクチャー (3)



アプリケーション開発

ブロックチェーンアプリの開発プロセス

ノード
ネットワーク
設定

スマート
コントラクト
コーディング

単体テスト
(モック使用)

デプロイ

フロントエンド
開発

統合テスト

ブロックチェーンアプリの開発プロセス

ノード
ネットワーク
設定

スマート
コントラクト
[illegible]

単体テスト
(モック使用)

つらい

デプロイ

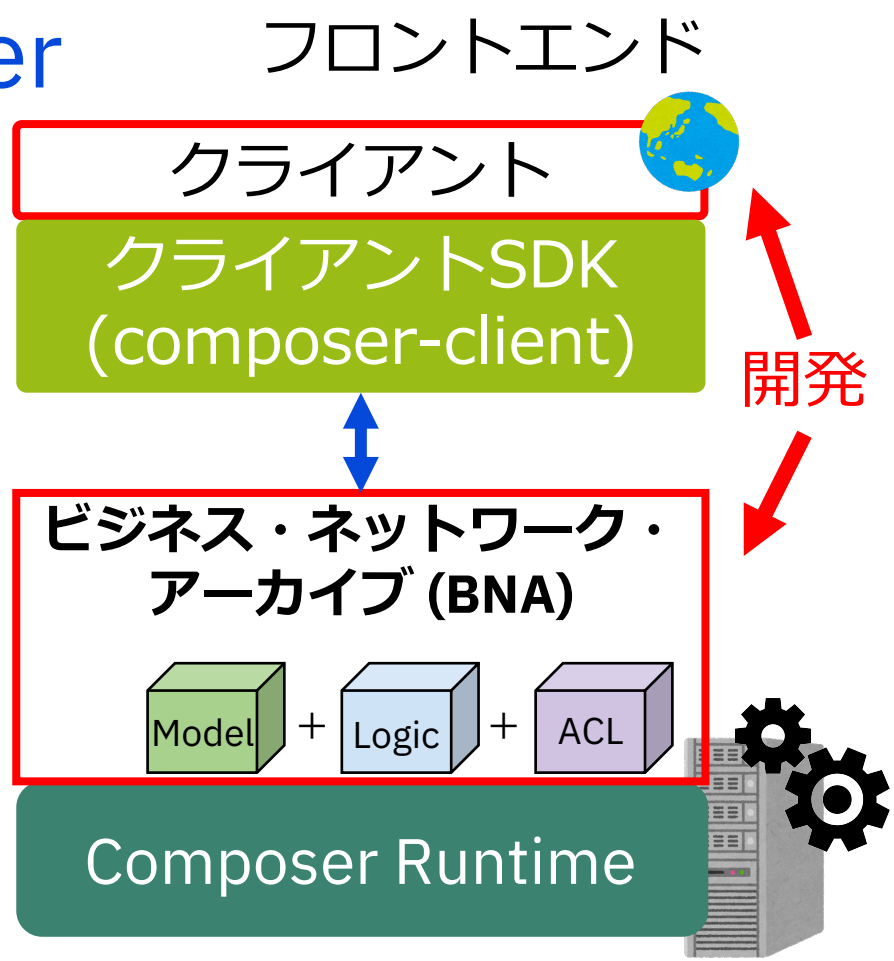
フロントエンド
開発

統合テスト

Hyperledger Composer

Hyperledger Composer

- Hyperledger Fabric上のフレームワーク
- 基本ロジックの自動生成
- テスト支援ツール群
- **PlayGround付属**



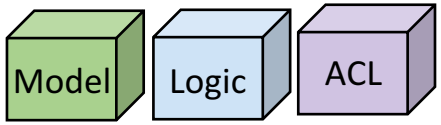
今日のデモ

Hyperledger Composerのコーディング例を
PlayGround (https://ibm.biz/playgnd)
で紹介

ブラウザ上で試せる
開発環境・インスタンス不要！
※もちろんローカル環境で開発もできます
※もちろん実際のネットワークにデプロイ可

Hyperledger Composerの開発パターン

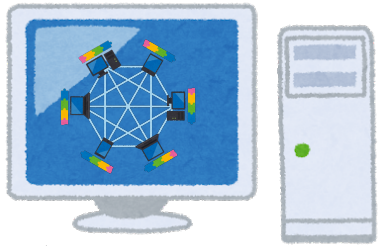
PlayGround



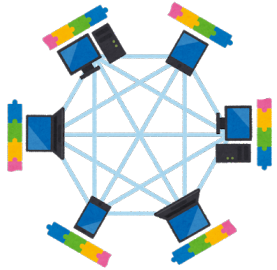
開発

BNAファイル

デプロイ



テスト



ローカル環境

Business Network

Participant

- アプリを使うユーザ
- Hyperledger Fabricのユーザ・証明書と対応

Asset

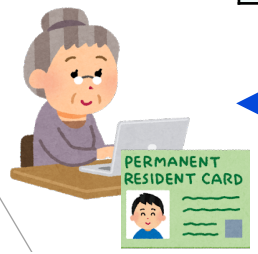
- 何らかの「資産」

Transaction

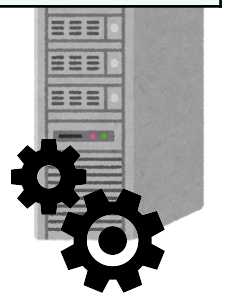
- 「資産」の問い合わせ・変更

Asset

111	{vin: 111, type: TA-GGA, owner: Alice }
112	…



Transaction



Participant

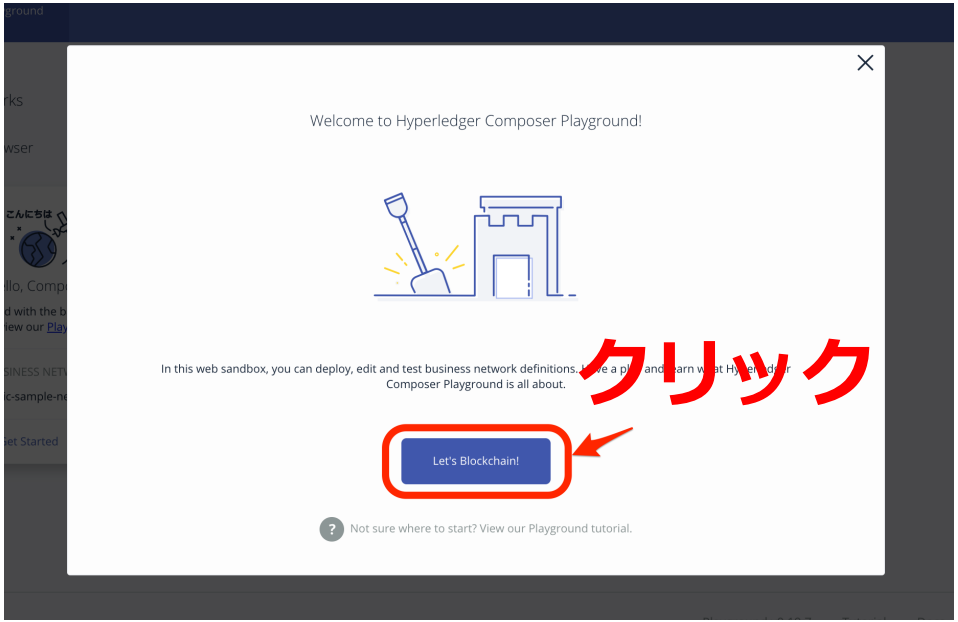
簡単なプログラムを作ってみる

題材: 自動車

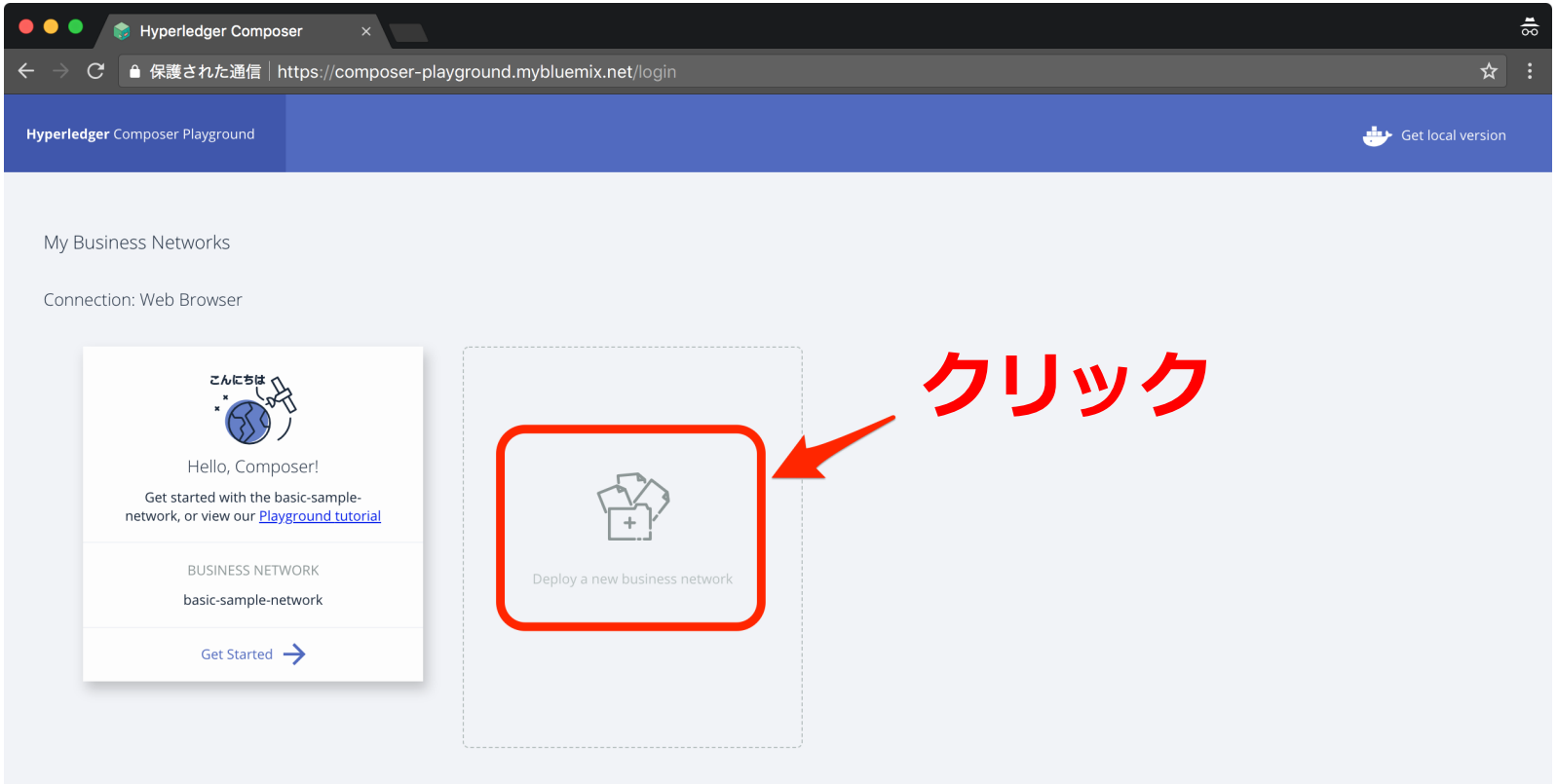
- 車 (Car) という資産を扱う
- 車のメーカー (Maker) と、
  車のオーナー (Owner) がいる
- オーナー間で車の移転 (Transfer) ができる

PlayGroundを開く

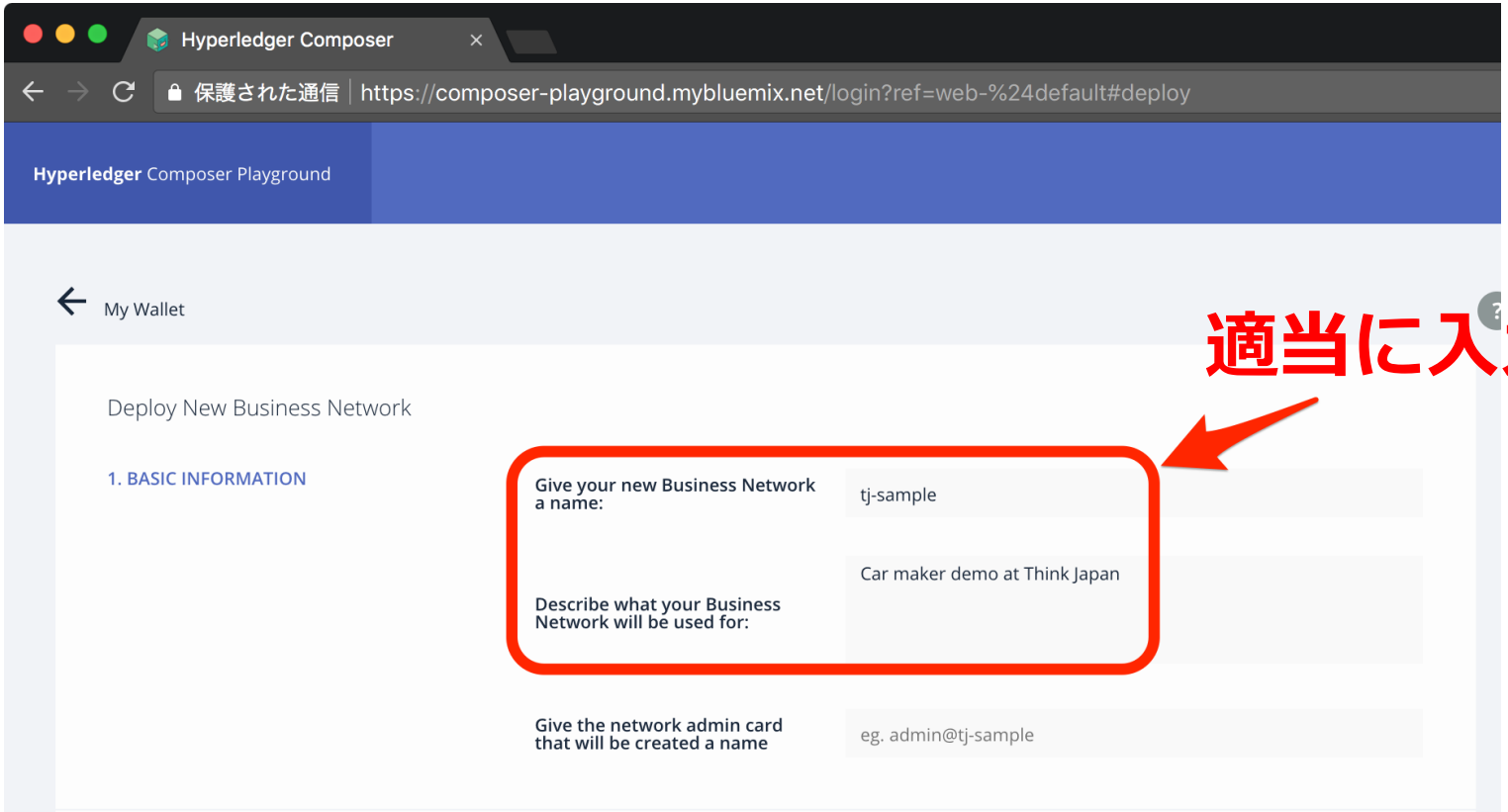
https://composer-playground.mybluemix.net
https://ibm.biz/playgnd



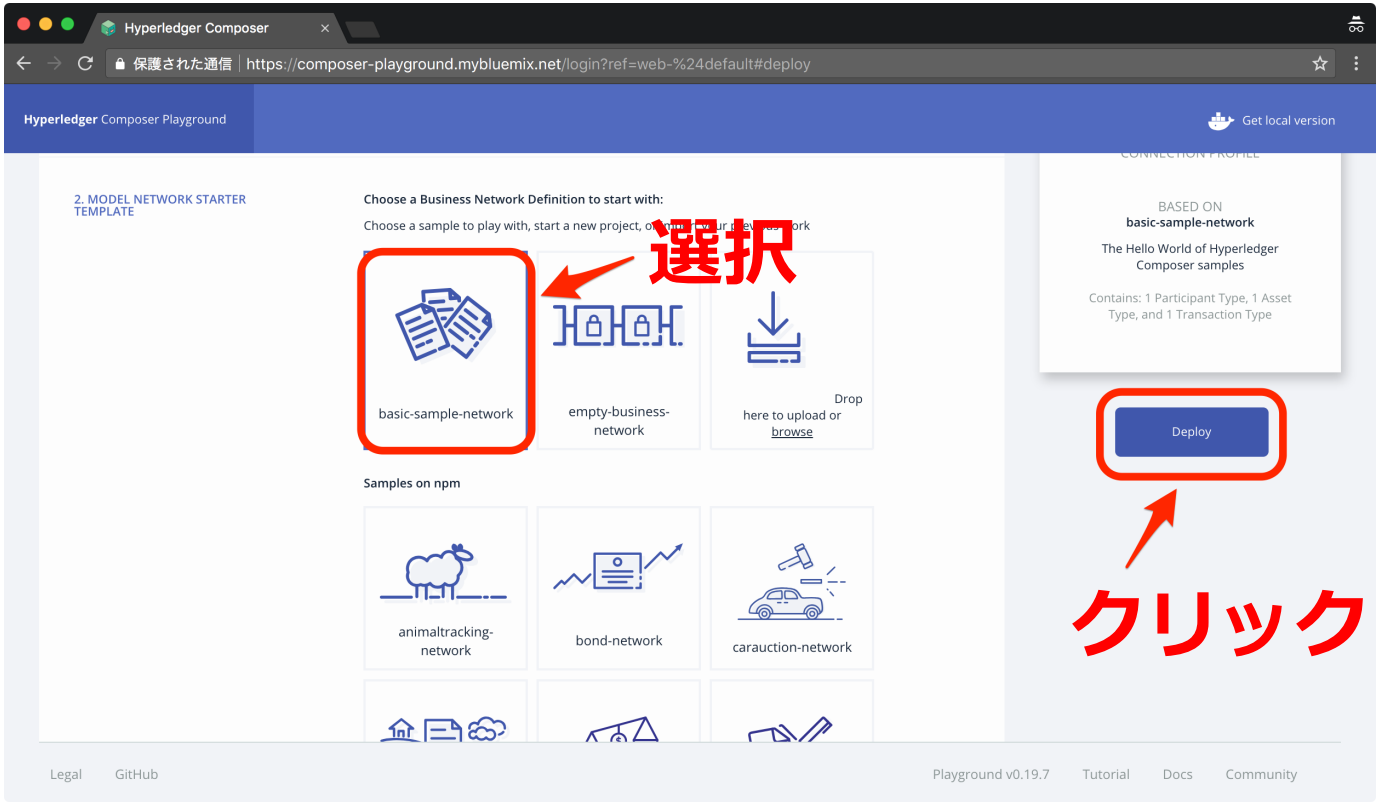
新規Business Network作成



Basic Information入力

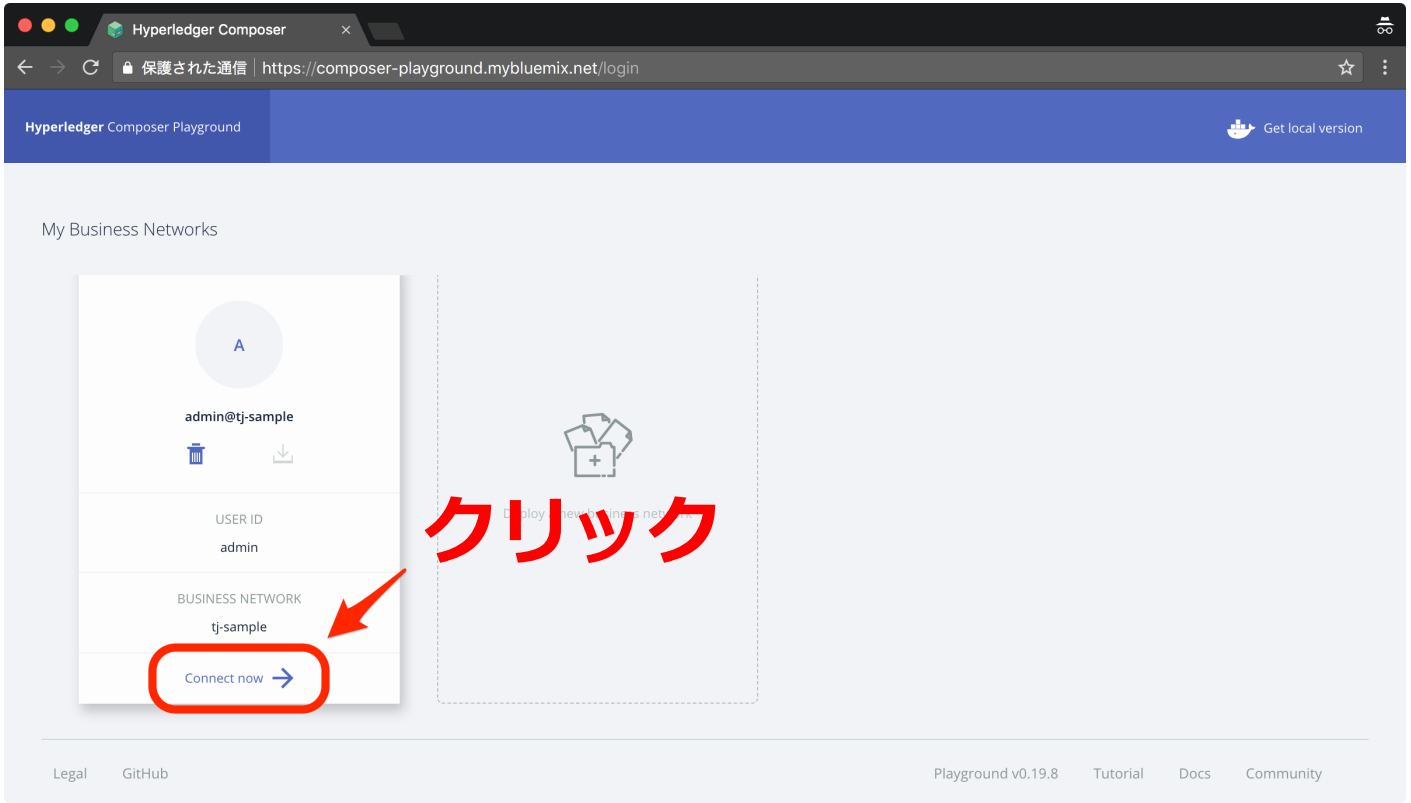


Template選択

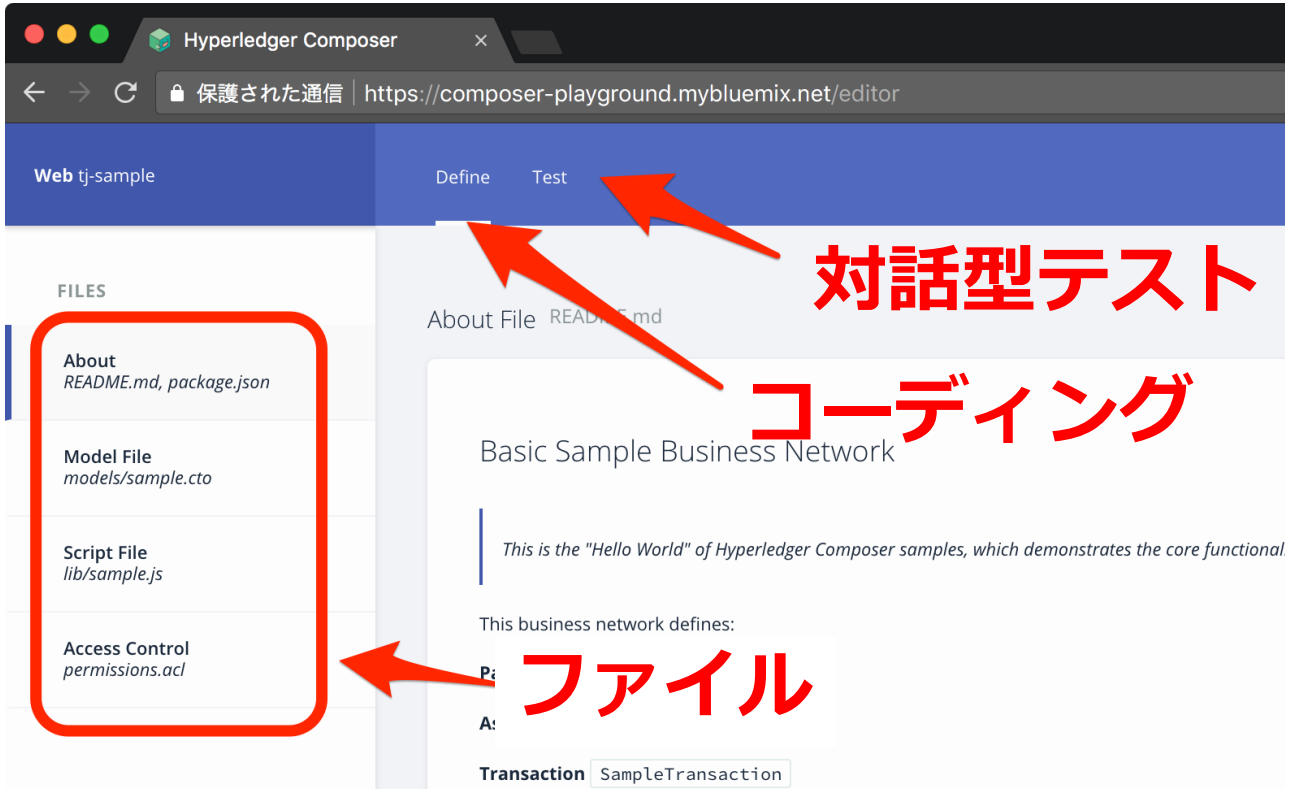


think Japan

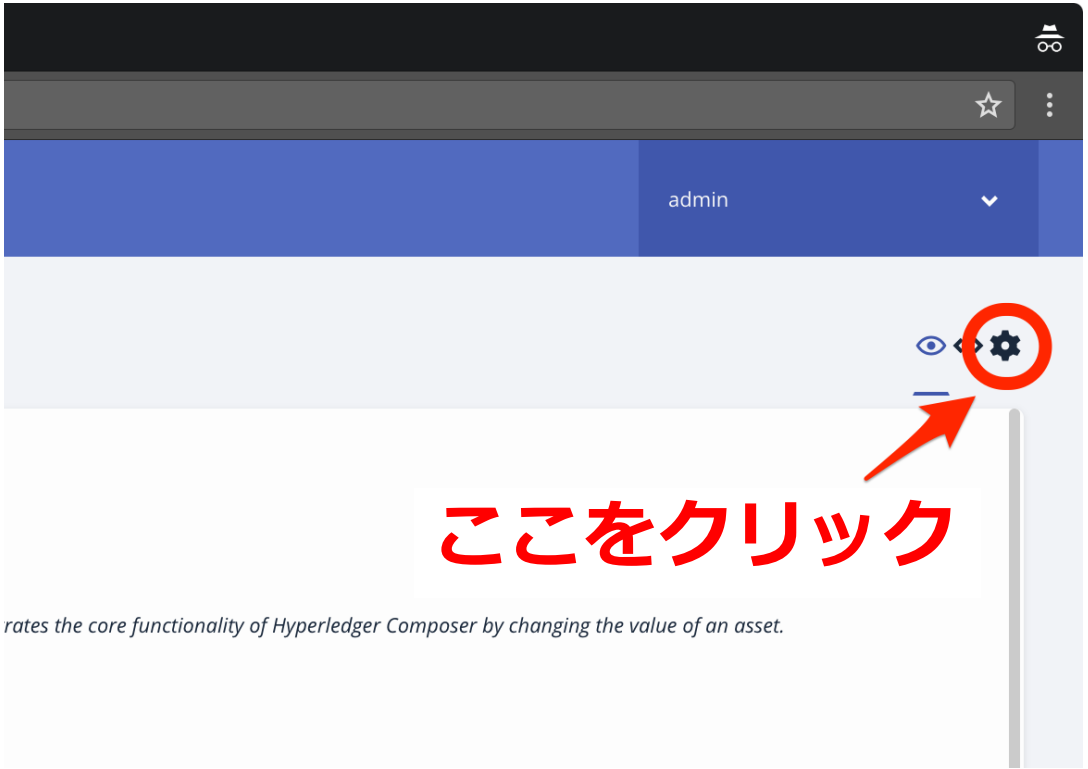
Business Network選択



ダッシュボード


Hyperledger Composer
保護された通信 | https://composer-playground.mybluemix.net/editor
Web tj-sample
Define
Test
FILES
About
README.md, package.json
Model File
models/sample.cto
Script File
lib/sample.js
Access Control
permissions.acl
About File
Basic Sample Business Network
This is the "Hello World" of Hyperledger Composer samples, which demonstrates the core functional
This business network defines:
Transaction
SampleTransaction
対話型テスト
コーディング
ファイル

参考: package.jsonを見るには



モデル (models/sample.cto)

クラス定義

- Asset　車 (Car)
- Participant　メーカー (Maker),
  オーナー (Owner)
- Transaction　移転 (Transfer)

文法

- JSに似ている独自の記法

Namespace

このままで

```
namespace org.example.basic
```

Participant

Participant定義

自動車メーカとオーナーを作成

KVSに入れるときのキー

```
participant Owner identified by id {
  o String id
  o String firstName
  o String lastName
}
```

小文字のオー

Participant定義

自動車メーカとオーナーを作成

```
participant Maker identified by id {
  o String id
  o String name
}
```

Asset定義

自動車asset

- 今回、データ正規化は考えない

```
asset Car identified by vin {
  o String vin
  o String type
  o String name
  --> Maker maker
  --> Owner owner
}
```

他エンティティへの参照

Transaction定義

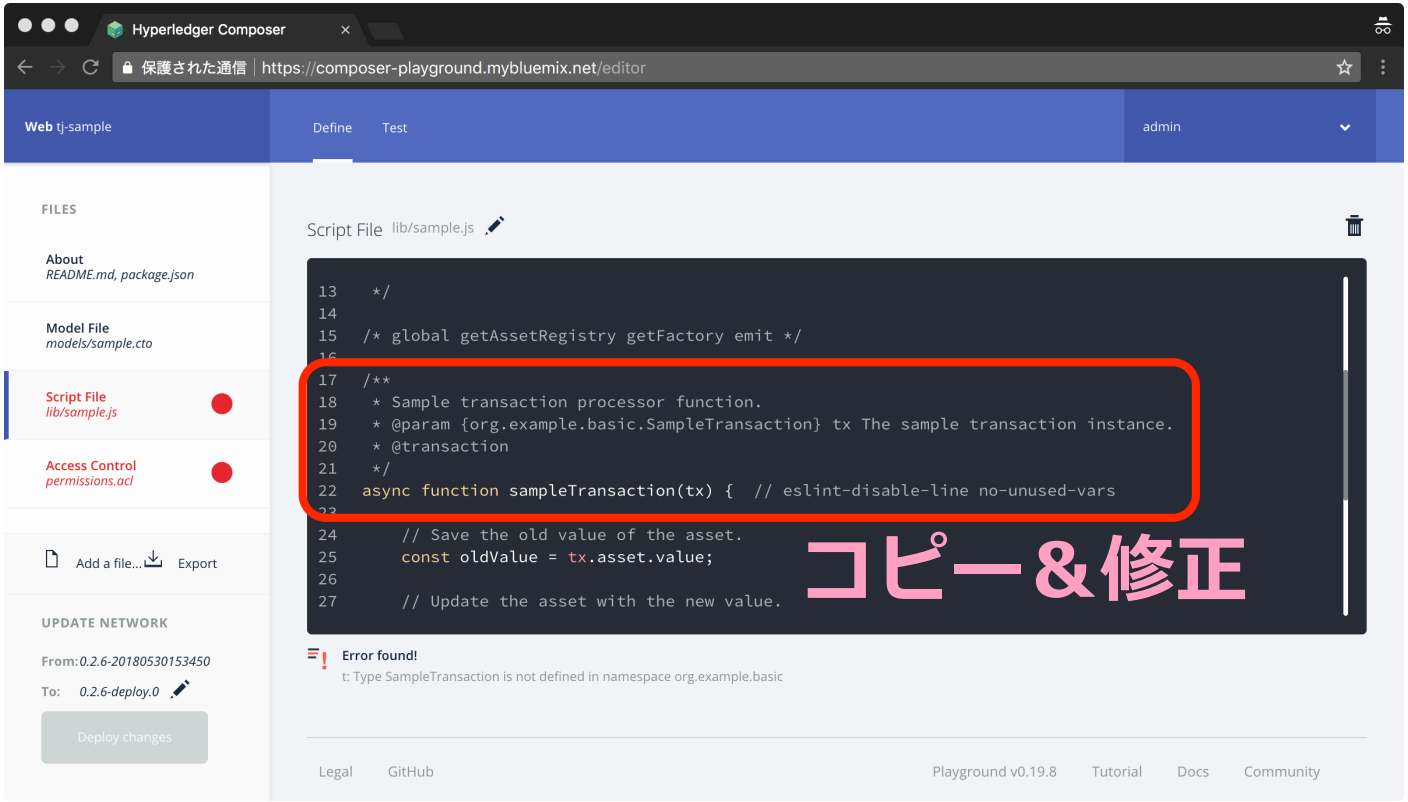
Transaction
パラメータの
データ型を定義

- ロジックは
  別ファイル
  (lib/sample.js)

```
transaction Transfer {
  --> Car car
  --> Owner newOwner
}
```

車の移転

Transactionロジック (lib/sample.js)



Transactionロジック (lib/sample.js)

コメント内の記法も決まっているので注意

引数の型と名前

```
/**
 * Sample transaction processor function.
 * @param {org.example.basic.Transfer} tx ...
 * @transaction
 */
async function transfer (tx) {  // eslint...
```

この関数がTXであること

ES2017非同期関数

関数名

Transactionロジック (lib/sample.js)

```
async function transfer(tx) {  // eslint...
    // Get the asset registry for the asset.
    const assetRegistry =
            await getAssetRegistry(
              'org.example.basic.Car'
            );
    (後半へ続く)
}
```

レジストリ取得

ES2017非同期関数

Transactionロジック (lib/sample.js)

```
async function transfer(tx) {  // eslint...
    (続き)

    // Update the asset with the new value.
    tx.car.owner = tx.newOwner;
                                  TX引数取得・更新
    // Update the asset in the asset registry.
    await assetRegistry.update(tx.car);
}                                 Asset更新
```

アクセス制御 (permissions.acl) [1]

(Participant, Resource,
Operation[, Condition]) → ALLOW | DENY

カテゴリ	例
Participant (p)	org.example.User | <システムadmin>
Resource (r)	org.example.* | ** | <システムリソース>

Asset または Transaction

アクセス制御 (permissions.acl) [2]

(Participant, Resource,
Operation[, Condition]) → ALLOW | DENY

カテゴリ	例
Operation	ALL | READ | UPDATE
Condition (オプション)	p.getIdentifier() == r.owner.getIdentifier()

「アセットのオーナーのみ」○○できる

アクセス制御 (permissions.acl)

ルール1: 誰でも自動車の情報を読める

```
rule EverybodyCanReadEverything {
    description: "Allow all ..."
    participant: "org.example.basic.*"
    operation: READ
    resource: "org.example.basic.Car"
    action: ALLOW
}
```

読む

誰でも

自動車

アクセス制御 (permissions.acl)

[2] : オーナーのみTransferトランザクションを呼べる

```
rule OwnerCanSubmitTransfer {
    description: "Allow ..."
    participant: "org.example.basic.Owner"
    operation: CREATE
    resource: "org.example.basic.Transfer"
    action: ALLOW
}
```

オーナー

(TXを) 呼ぶ

Transferトランザクション

アクセス制御 (permissions.acl)

[3]: オーナーは自分の自動車のみ移転できる

```
rule OwnerCanTransferMyCars {
    description: "Allow all ..."
    participant(p): "org.example.basic.Owner"
    operation: UPDATE
    resource(r): "org.example.basic.Car"
    condition: (r.owner.getIdentifier()
                  === p.getIdentifier())
    action: ALLOW
}
```

オーナー

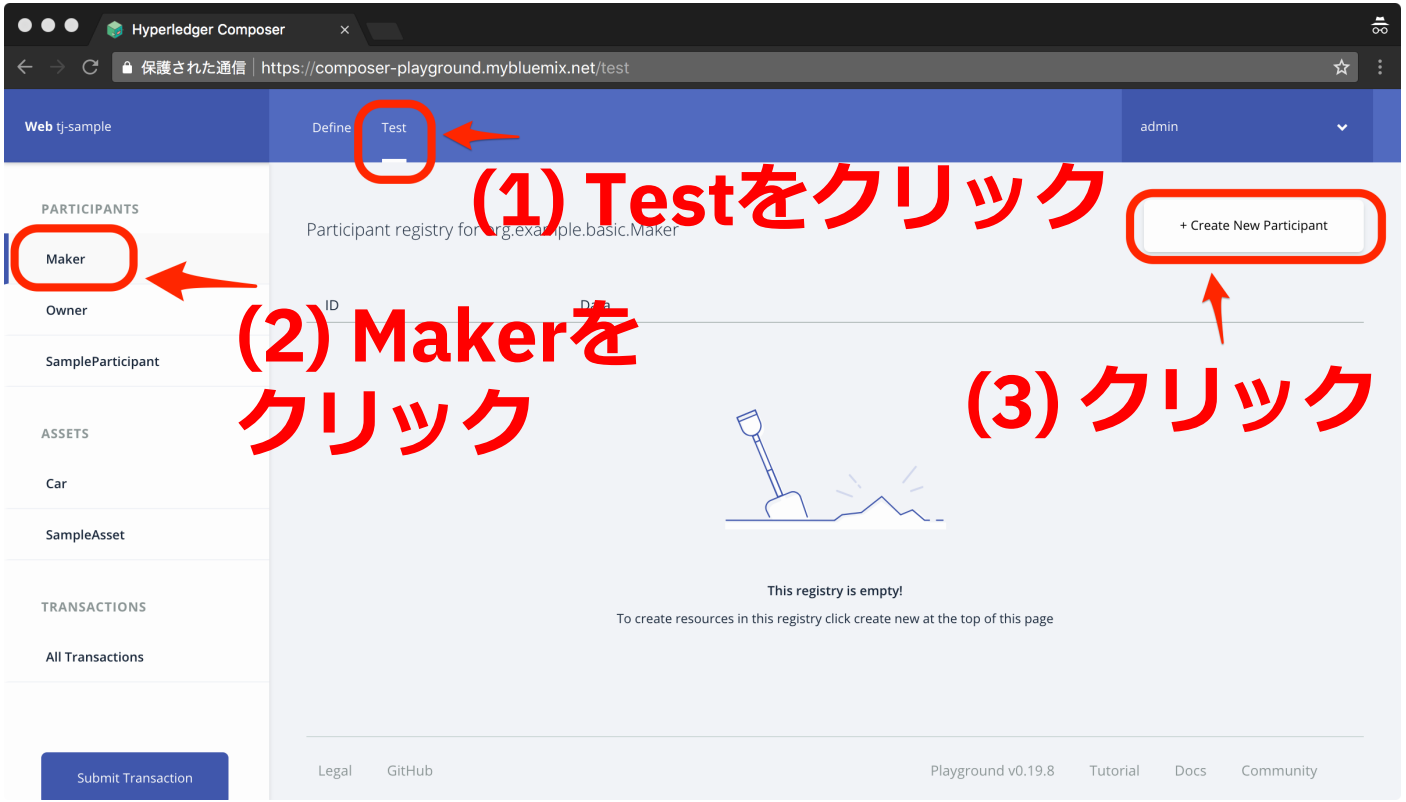
更新

自分のAsset

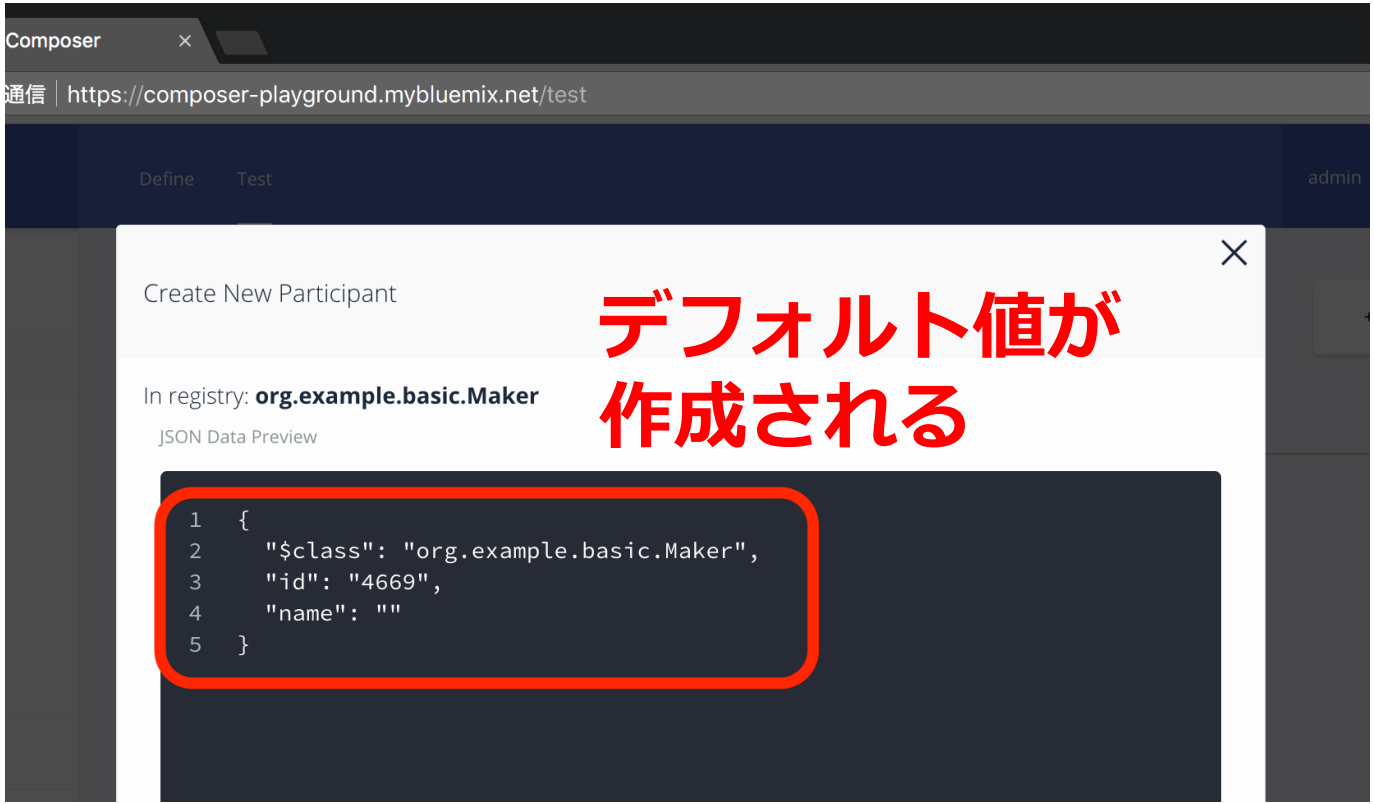
デプロイ


Script File
lib/sample.js
Access Control
permissions.acl
Add a file...
Export
UPDATE NETWORK
From: 0.2.6-deploy.0
To: 0.2.6-deploy.1
Deploy changes
6 * http://www.apache.org/license
7 *
8 * Unless required by applicable
9 * distributed under the License
10 * WITHOUT WARRANTIES OR CONDITI
11 * See the License for the speci
12 * limitations under the License
13 */
14
15 /**
Everything looks good!
Any problems detected in your code would be reported
クリック
Legal
GitHub

Testタブで対話的テスト



Maker作成

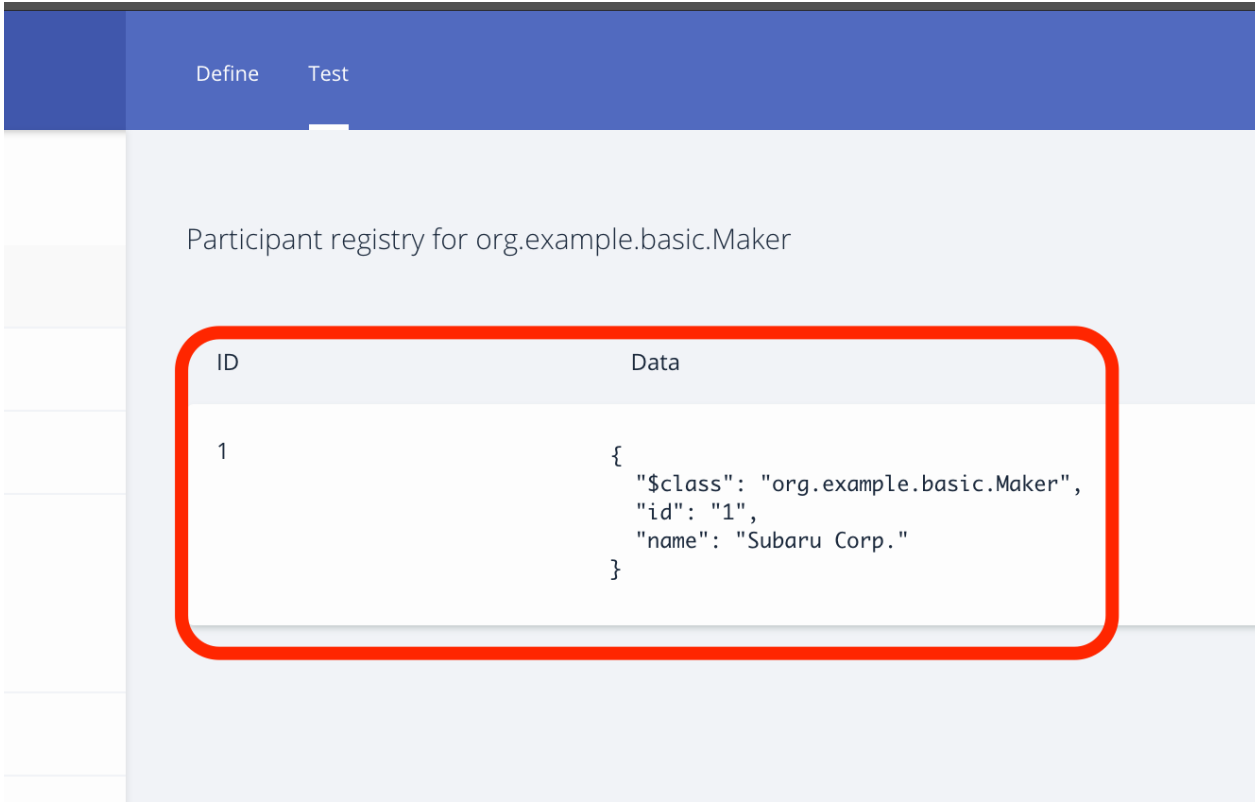

Composer
通信 | https://composer-playground.mybluemix.net/test
Define
Test
admin
Create New Participant
In registry: org.example.basic.Maker
JSON Data Preview
1 {
2 "$class": "org.example.basic.Maker",
3 "id": "4669",
4 "name": ""
5 }
デフォルト値が
作成される

Maker作成

クラス

```
{
  "$class": "org.example.basic.Maker",
  "id": "1",
  "name": "Subaru Corp.s"
}
```

Maker作成


Define
Test
Participant registry for org.example.basic.Maker
ID
Data
1
{
"$class": "org.example.basic.Maker",
"id": "1",
"name": "Subaru Corp."
}

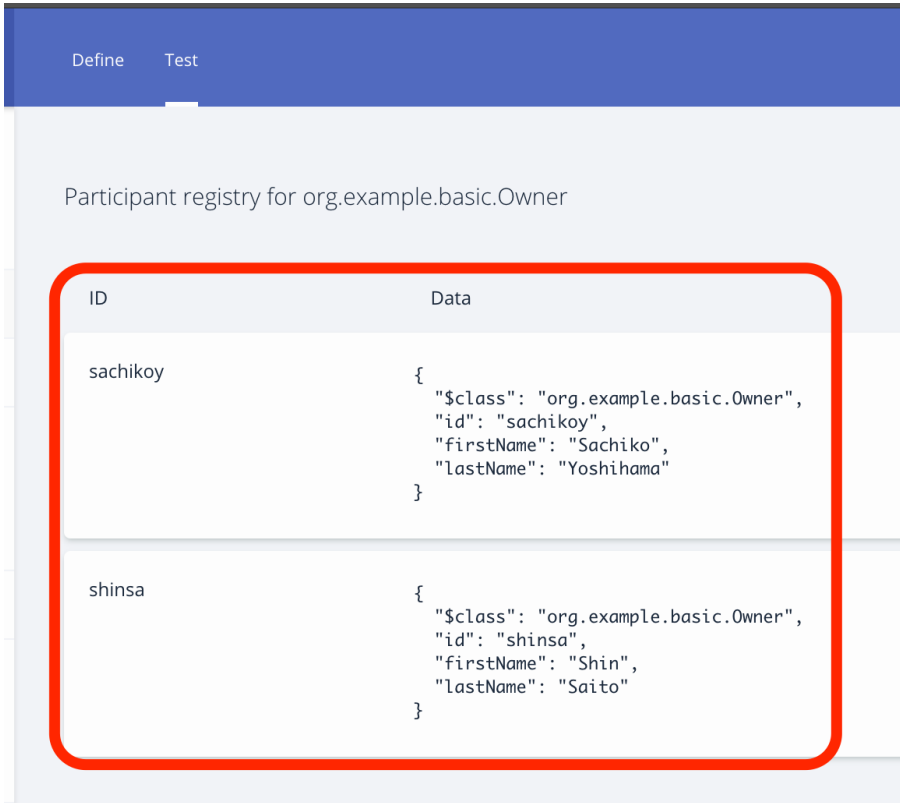
Owner作成

```
{
  "$class": "org.example.basic.Owner",
  "id": "shinsa",
  "firstName": "Shin",
  "lastName": "Saito"
}
```

Owner作成

```
{
  "$class": "org.example.basic.Owner",
  "id": "sachikoy",
  "firstName": "Sachiko",
  "lastName": "Yoshihama"
}
```

Owner作成


Define
Test
Participant registry for org.example.basic.Owner
ID
Data
sachikoy
{
"$class": "org.example.basic.Owner",
"id": "sachikoy",
"firstName": "Sachiko",
"lastName": "Yoshihama"
}
shinsa
{
"$class": "org.example.basic.Owner",
"id": "shinsa",
"firstName": "Shin",
"lastName": "Saito"
}

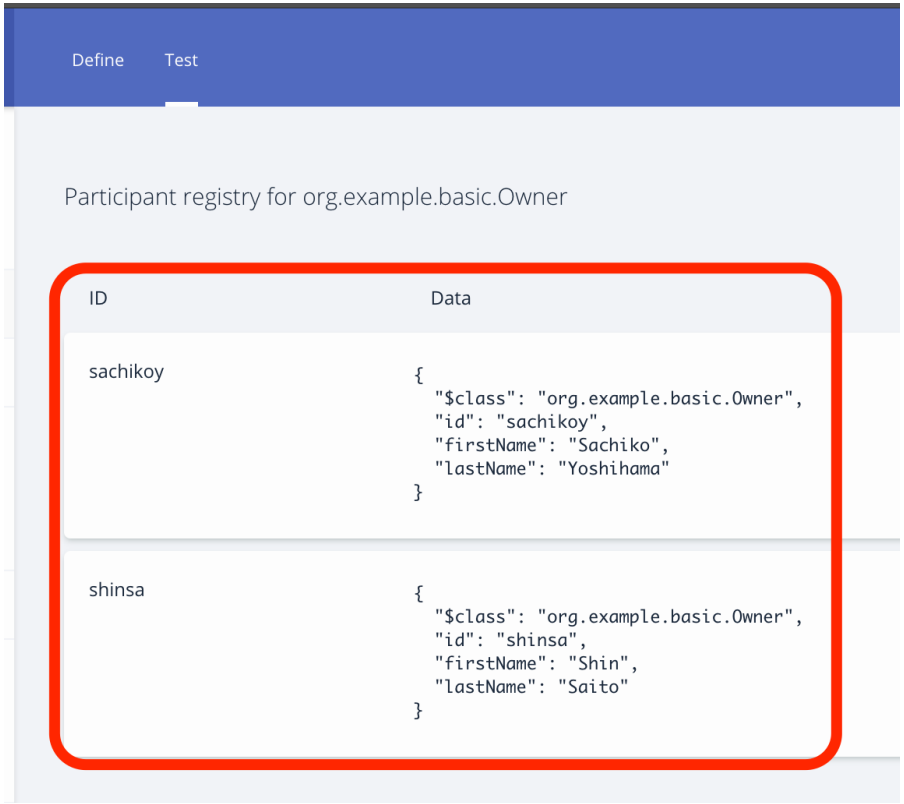
Asset作成

Carを作成

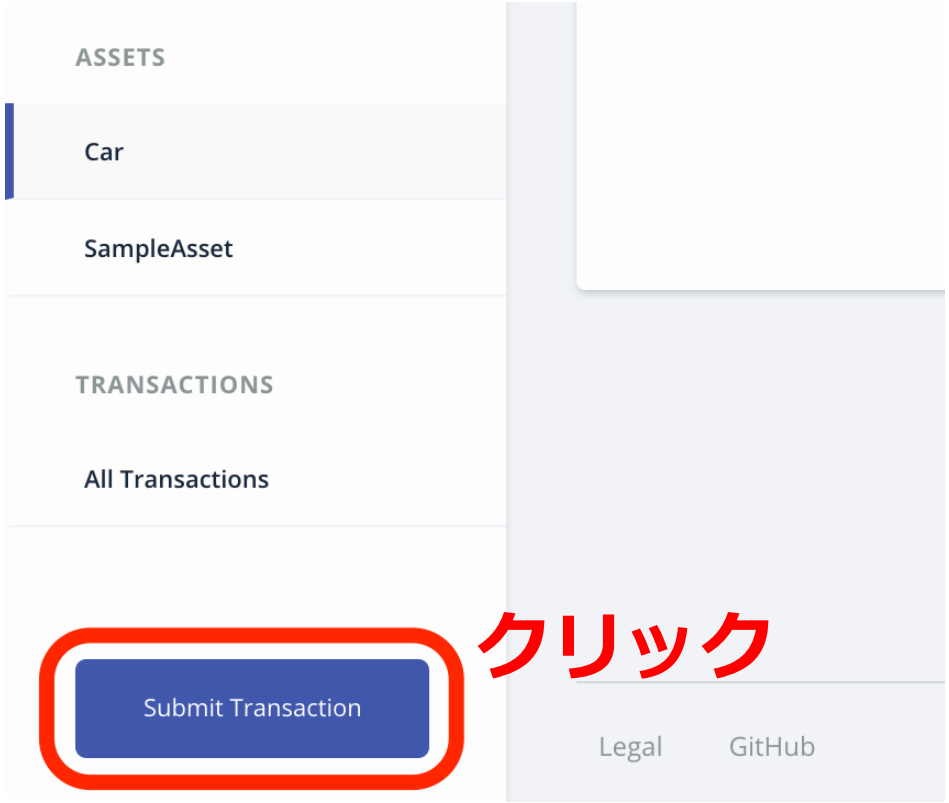
```
{
  "$class": "org.example.basic.Car",
  "vin": "1",
  "type": "CBA-VAB",
  "name": "WRX STI",
  "maker": "resource:org.example.basic.Maker#1",
  "owner": "resource:org.example.basic.Owner#shinsa"
}
```

resource:<クラス>#<アセットID>

Owner作成


Define
Test
Participant registry for org.example.basic.Owner
ID
Data
sachikoy
{
"$class": "org.example.basic.Owner",
"id": "sachikoy",
"firstName": "Sachiko",
"lastName": "Yoshihama"
}
shinsa
{
"$class": "org.example.basic.Owner",
"id": "shinsa",
"firstName": "Shin",
"lastName": "Saito"
}

Transaction作成



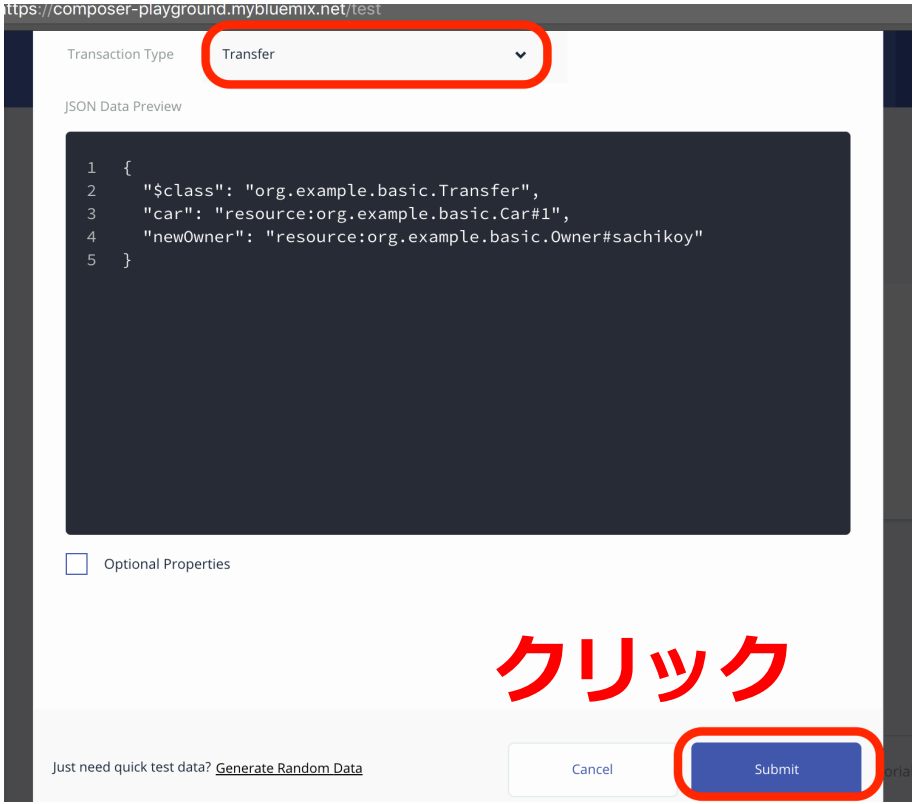
Transaction作成

Transferを呼ぶ

```
{s
  "$class": "org.example.basic.Transfer",
  "car": "resource:org.example.basic.Car#1",
  "newOwner":
    "resource:org.example.basic.Owner#sachikoy"
}
```

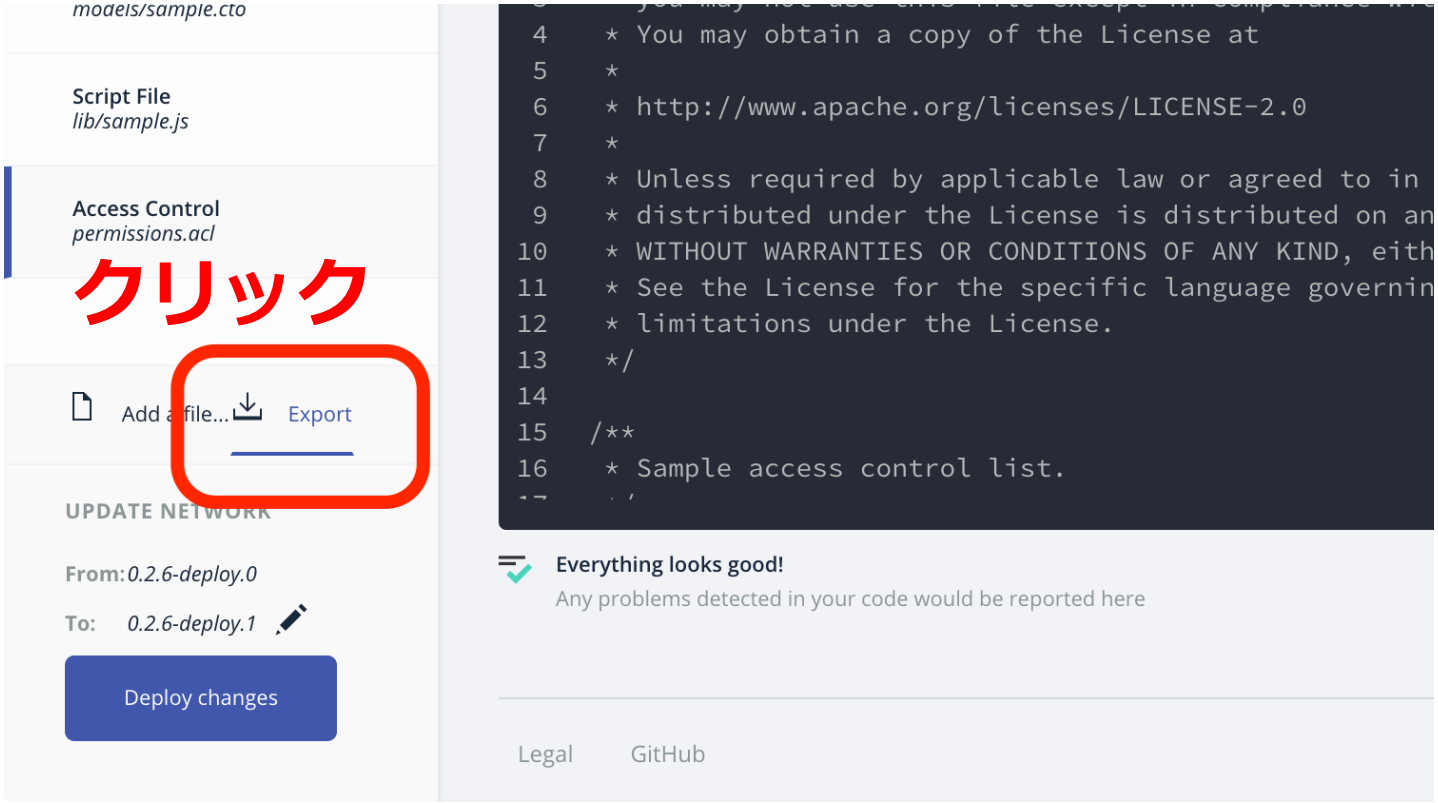
Transaction作成

Transferを選択



Transaction作成

BNA作成


models/sample.cto
Script File
lib/sample.js
Access Control
permissions.acl
クリック
Add a file...
Export
UPDATE NETWORK
From: 0.2.6-deploy.0
To: 0.2.6-deploy.1
Deploy changes
4 * You may obtain a copy of the License at
5 *
6 * http://www.apache.org/licenses/LICENSE-2.0
7 *
8 * Unless required by applicable law or agreed to in
9 * distributed under the License is distributed on an
10 * WITHOUT WARRANTIES OR CONDITIONS OF ANY KIND, eith
11 * See the License for the specific language governin
12 * limitations under the License.
13 */
14
15 /**
16 * Sample access control list.
Everything looks good!
Any problems detected in your code would be reported here
Legal
GitHub

ちゃんとしたテストもできます

Mocha
- 有名なNode.jsテストフレームワーク

Cucumber
- BDD (Behavior Driven Dev.) フレームワーク

どちらもデプロイは不要でテスト可能

フロントエンドについて

Composer-client
- Node.jsライブラリ

REST Server
- Webサーバを自動生成
- Curlなどで気軽にアクセスが可能

Angularコンポーネント作成

説明しなかったこと

クエリ (.qry ファイル)

- リレーショナルな問い合わせを実装
- 例:
  「オーナーの年収が1000万円以上のCar」

イベント

- クライアントへの非同期通信

開発アプローチの使い分け

フレームワーク開発 (Hyperledger Composer)

- アプリケーションを迅速に開発したい時に。
  支援ツール・テスト環境なども充実

ネイティブ開発

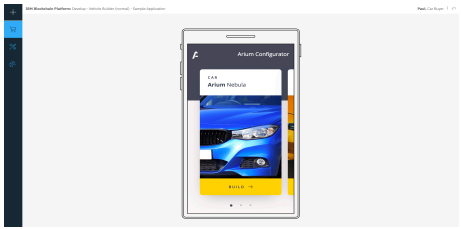
- とにかく速度を追求したい、複雑なリレーションを扱いたい、ネットワーク構成が複雑、
  Go言語が大好き、などといった時に。

IBM Blockchain Platform (IBP)

IBM Blockchain Platform Starter Plan

https://console.bluemix.net

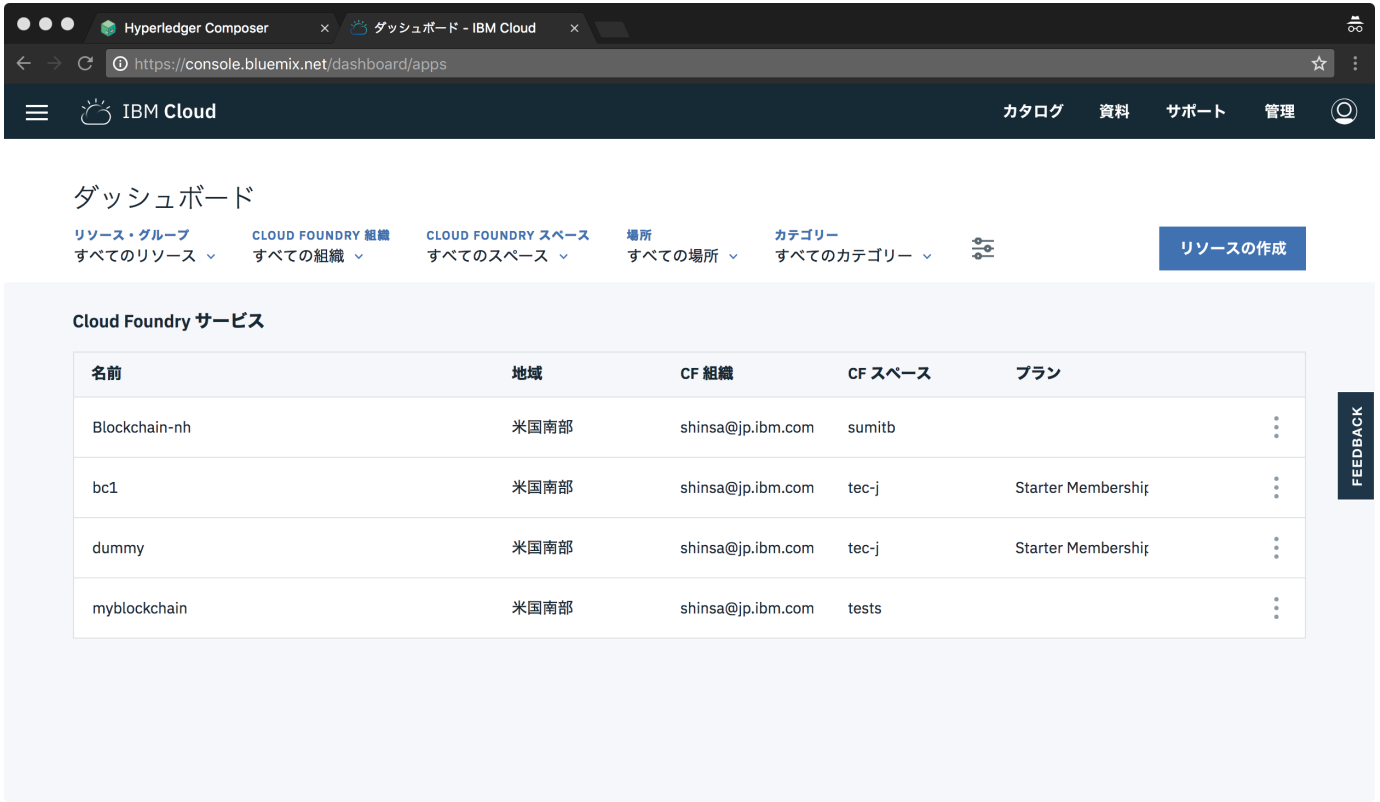
- ネットワークが直ちに立ち上がる
- 簡単にネットワーク管理
- サンプル・アプリケーションも充実



サンプル・アプリ

※Beta期間中は無料、まもなく正式公開
※正式公開後、30日無料トライアルを予定

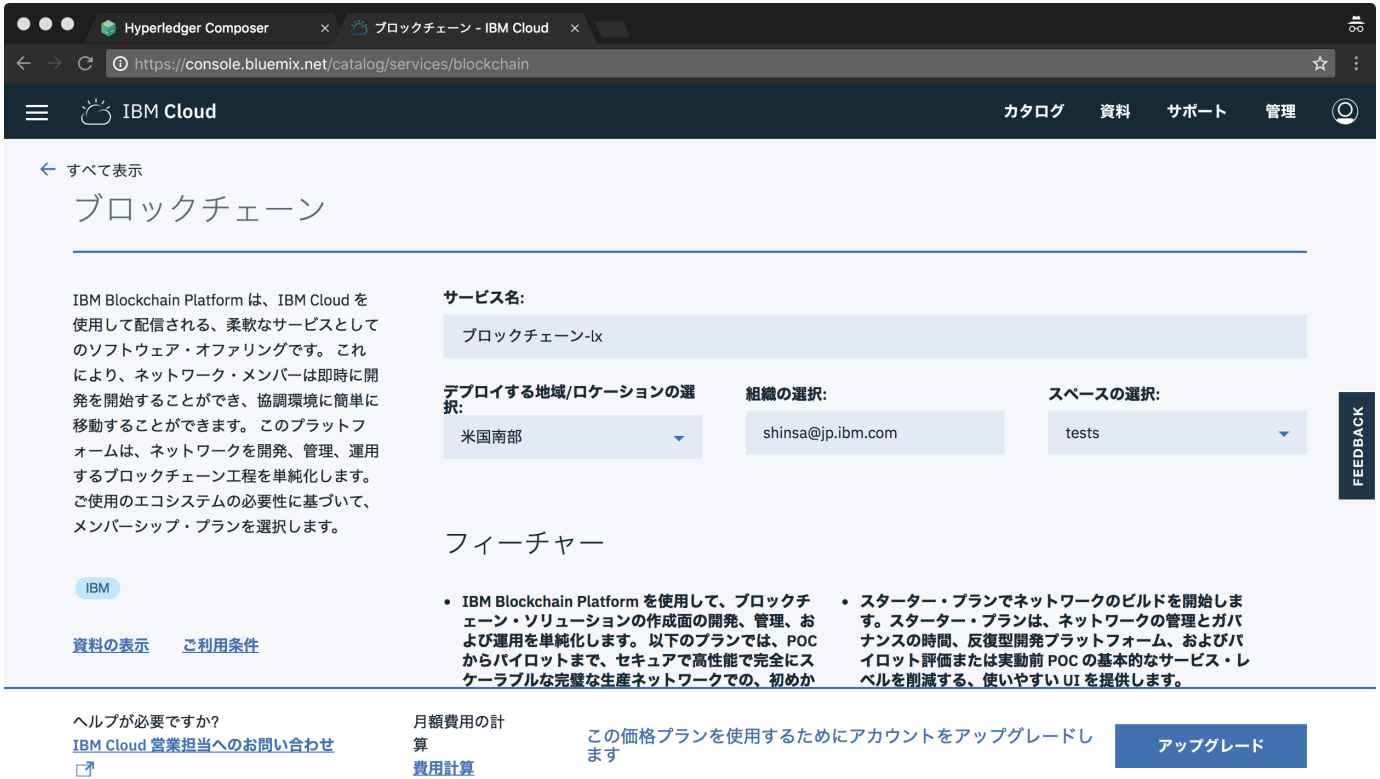
リソースの作成


Hyperledger Composer
ダッシュボード - IBM Cloud
https://console.bluemix.net/dashboard/apps
IBM Cloud
カタログ
資料
サポート
管理
ダッシュボード
リソース・グループ
すべてのリソース
CLOUD FOUNDRY 組織
すべての組織
CLOUD FOUNDRY スペース
すべてのスペース
場所
すべての場所
カテゴリー
すべてのカテゴリー
リソースの作成
Cloud Foundry サービス
名前
地域
CF 組織
CF スペース
プラン
Blockchain-nh
米国南部
shinsa@jp.ibm.com
sumitb
bc1
米国南部
shinsa@jp.ibm.com
tec-j
Starter Membershi
dummy
米国南部
shinsa@jp.ibm.com
tec-j
Starter Membershi
myblockchain
米国南部
shinsa@jp.ibm.com
tests
FEEDBACK

ブロックチェーン選択

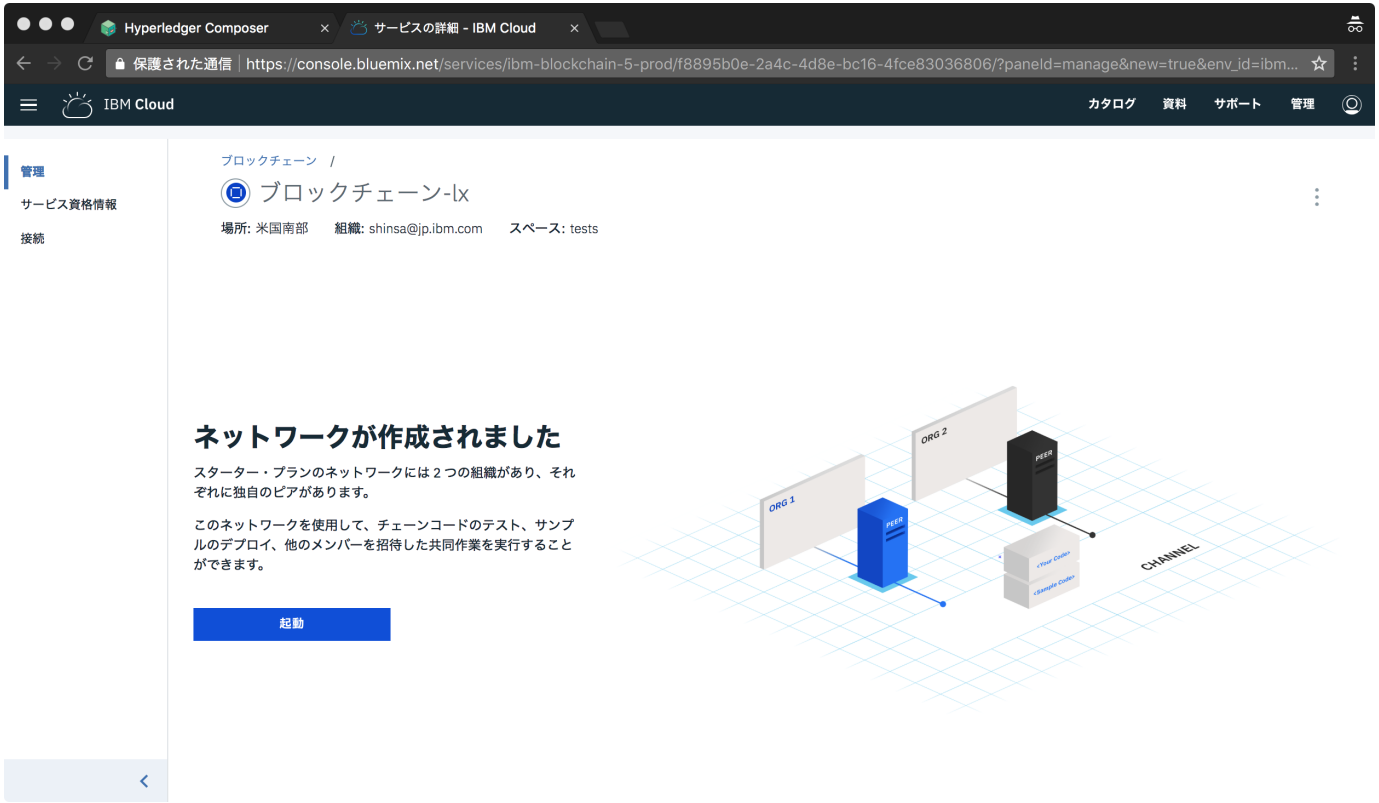

Hyperledger Composer
カタログ - IBM Cloud
保護された通信 | https://console.bluemix.net/catalog/?category=blockchain
IBM Cloud
カタログ
資料
サポート
管理
すべてのカテゴリー
インフラストラクチャー
コンピュート
ストレージ
ネットワーク
セキュリティー
コンテナー
VMware
プラットフォーム
ボイラープレート
API
アプリケーション・サービス
Blockchain
Cloud Foundry アプリ
データ & 分析
DevOps
金融
Functions
インテグレーション
IoT
検索
フィルター
このサービスは、所有権および制御がさまざまな組織に分散している Blockchain ビジネス・ネットワークの作成を可能にします。
ブロックチェーン
IBM Cloud での IBM のブロックチェーン・テクノロジーの使用
IBM
FEEDBACK

「米国南部」を選択

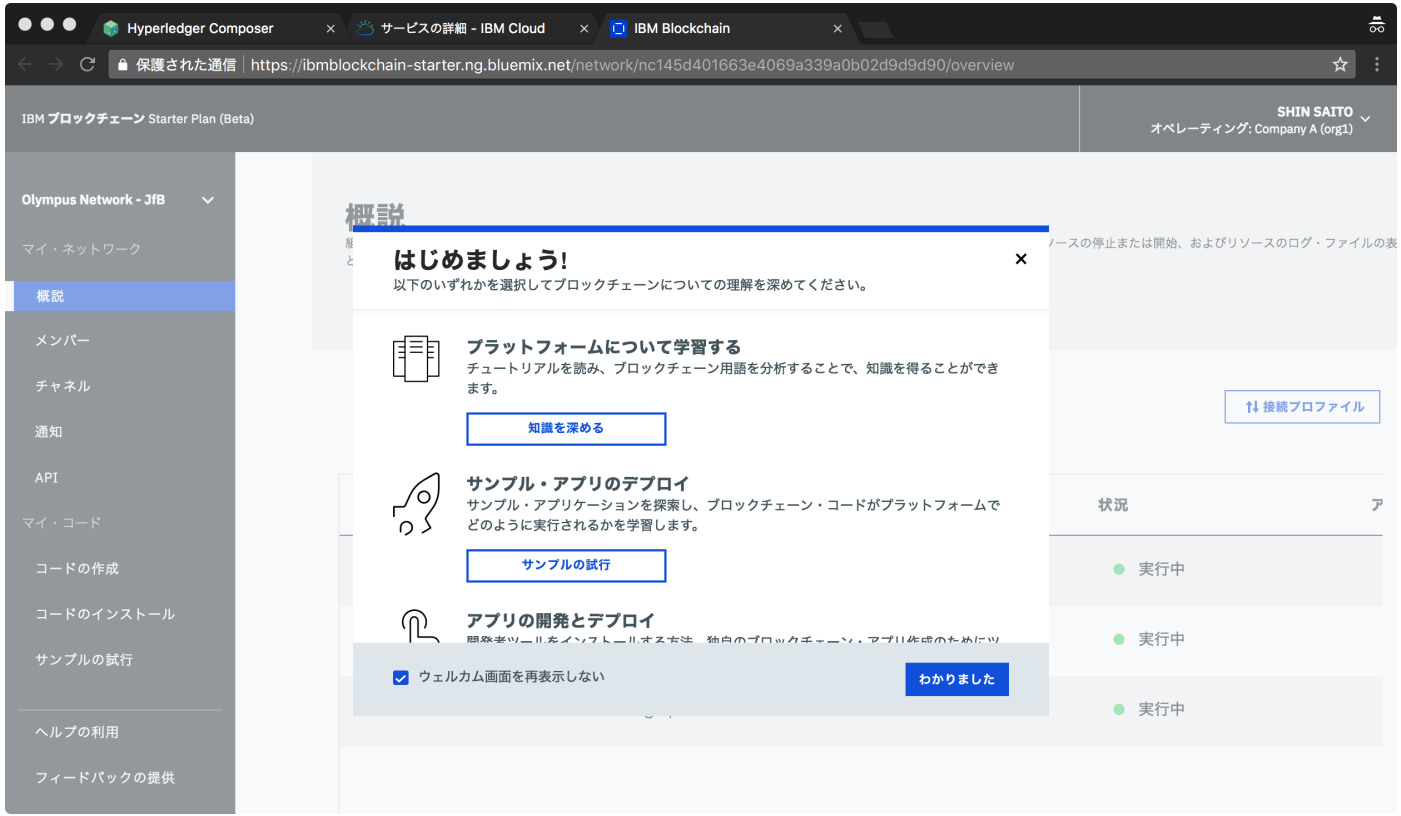

Hyperledger Composer
ブロックチェーン - IBM Cloud
https://console.bluemix.net/catalog/services/blockchain
IBM Cloud
カタログ
資料
サポート
管理
すべて表示
ブロックチェーン
IBM Blockchain Platform は、IBM Cloud を使用して配信される、柔軟なサービスとしてのソフトウェア・オファリングです。これにより、ネットワーク・メンバーは即時に開発を開始することができ、協調環境に簡単に移動することができます。このプラットフォームは、ネットワークを開発、管理、運用するブロックチェーン工程を単純化します。ご使用のエコシステムの必要性に基づいて、メンバーシップ・プランを選択します。
IBM
資料の表示
ご利用条件
サービス名:
ブロックチェーン-lx
デプロイする地域/ロケーションの選択:
米国南部
組織の選択:
shinsa@jp.ibm.com
スペースの選択:
tests
FEEDBACK
フィーチャー
IBM Blockchain Platform を使用して、ブロックチェーン・ソリューションの作成面の開発、管理、および運用を単純化します。以下のプランでは、POC からパイロットまで、セキュアで高性能で完全にスケーラブルな完璧な生産ネットワークでの、初めか
スターター・プランでネットワークのビルドを開始します。スターター・プランは、ネットワークの管理とガバナンスの時間、反復型開発プラットフォーム、およびパイロット評価または実動前 POC の基本的なサービス・レベルを削減する、使いやすい UI を提供します。
ヘルプが必要ですか?
IBM Cloud 営業担当へのお問い合わせ
月額費用の計算
費用計算
この価格プランを使用するためにアカウントをアップグレードします
アップグレード

スターター・メンバーシップ・プラン

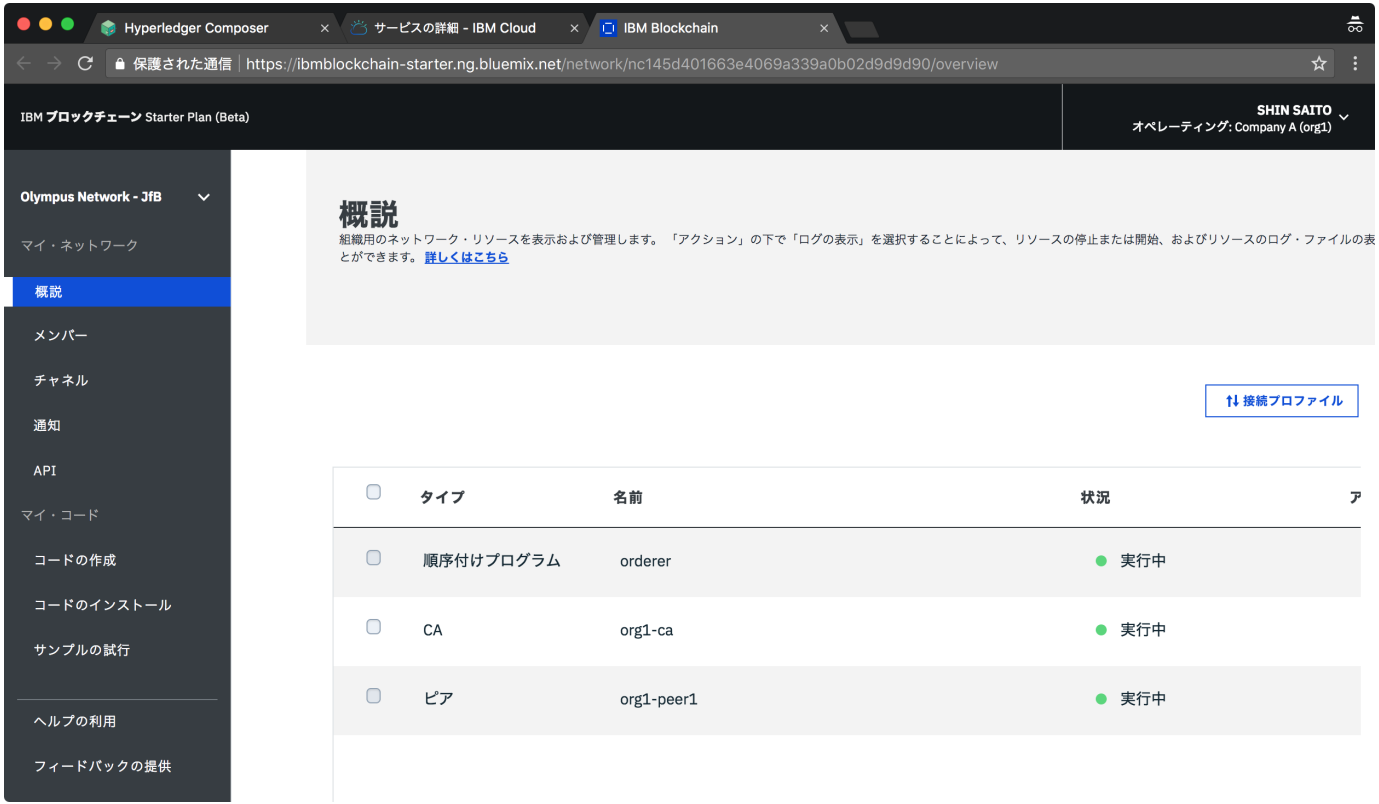
作成成功。起動する


Hyperledger Composer
サービスの詳細 – IBM Cloud
保護された通信 | https://console.bluemix.net/services/ibm-blockchain-5-prod/f8895b0e-2a4c-4d8e-bc16-4fce83036806/?paneId=manage&new=true&env_id=ibm...
IBM Cloud
カタログ
資料
サポート
管理
管理
サービス資格情報
接続
ブロックチェーン /
ブロックチェーン-lx
場所: 米国南部
組織: shinsa@jp.ibm.com
スペース: tests
ネットワークが作成されました
スターター・プランのネットワークには 2 つの組織があり、それぞれに独自のピアがあります。
このネットワークを使用して、チェーンコードのテスト、サンプルのデプロイ、他のメンバーを招待した共同作業を実行することができます。
起動
ORG 1
ORG 2
PEER
PEER
CHANNEL

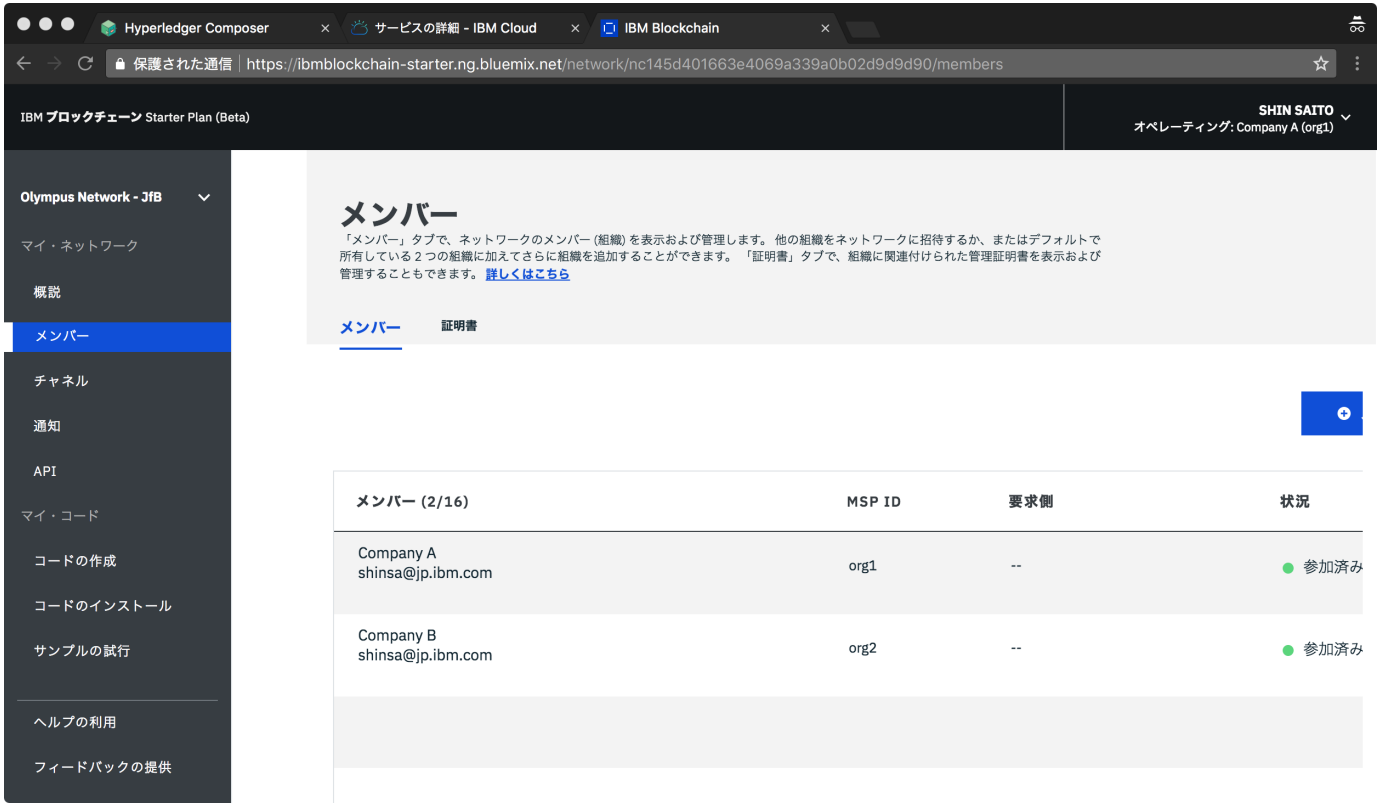
起動成功


Hyperledger Composer
サービスの詳細 - IBM Cloud
IBM Blockchain
保護された通信 | https://ibmblockchain-starter.ng.bluemix.net/network/nc145d401663e4069a339a0b02d9d9d90/overview
IBM ブロックチェーン Starter Plan (Beta)
SHIN SAITO
オペレーティング: Company A (org1)
Olympus Network - JfB
マイ・ネットワーク
概説
メンバー
チャネル
通知
API
マイ・コード
コードの作成
コードのインストール
サンプルの試行
ヘルプの利用
フィードバックの提供
概説
接続プロファイル
状況
実行中
実行中
実行中
はじめましょう!
以下のいずれかを選択してブロックチェーンについての理解を深めてください。
プラットフォームについて学習する
チュートリアルを読み、ブロックチェーン用語を分析することで、知識を得ることができます。
知識を深める
サンプル・アプリのデプロイ
サンプル・アプリケーションを探索し、ブロックチェーン・コードがプラットフォームでどのように実行されるかを学習します。
サンプルの試行
アプリの開発とデプロイ
ウェルカム画面を再表示しない
わかりました

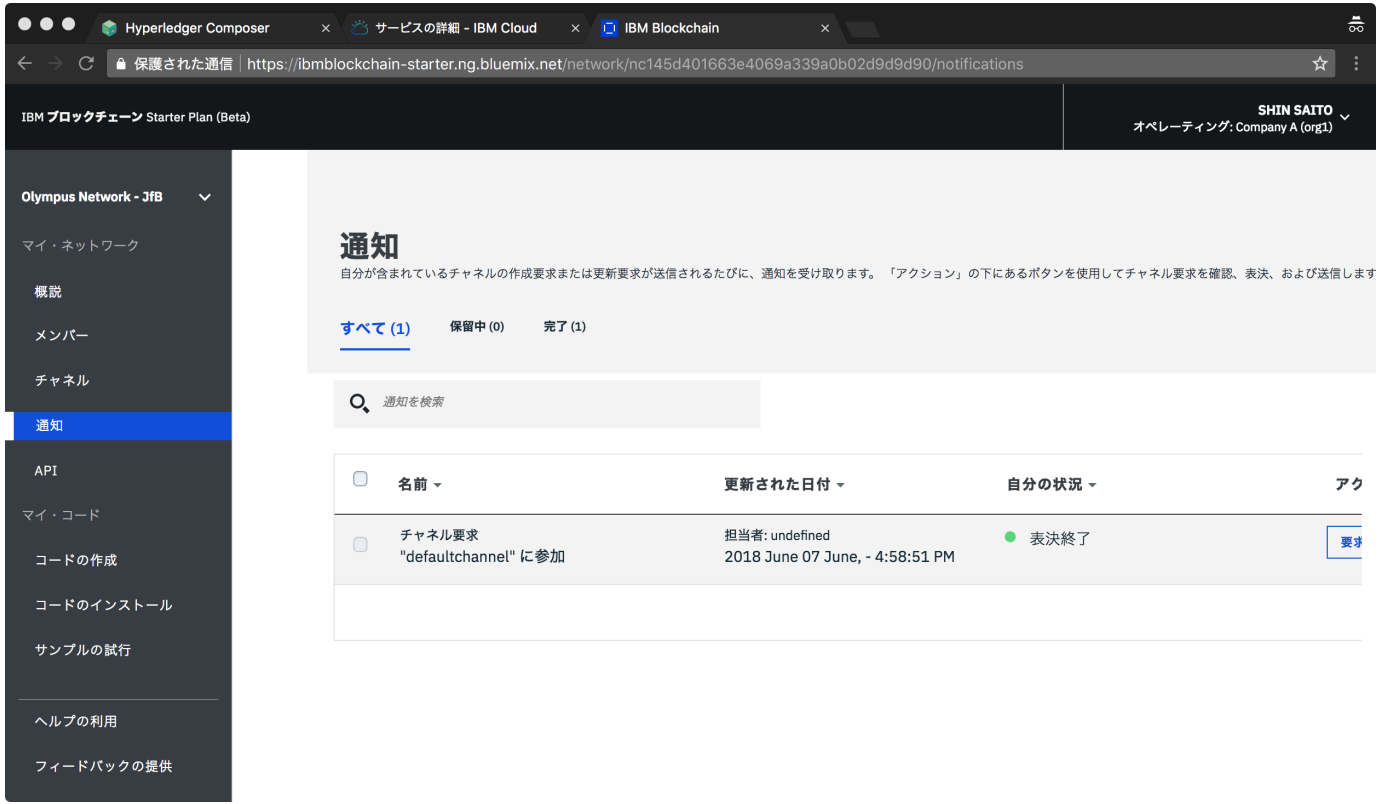
概要


Hyperledger Composer
サービスの詳細 - IBM Cloud
IBM Blockchain
保護された通信 | https://ibmblockchain-starter.ng.bluemix.net/network/nc145d401663e4069a339a0b02d9d9d90/overview
IBM ブロックチェーン Starter Plan (Beta)
SHIN SAITO
オペレーティング: Company A (org1)
Olympus Network - JfB
マイ・ネットワーク
概説
メンバー
チャネル
通知
API
マイ・コード
コードの作成
コードのインストール
サンプルの試行
ヘルプの利用
フィードバックの提供
概説
組織用のネットワーク・リソースを表示および管理します。「アクション」の下で「ログの表示」を選択することによって、リソースの停止または開始、およびリソースのログ・ファイルの表
とができます。詳しくはこちら
接続プロファイル
タイプ
名前
状況
順序付けプログラム
orderer
実行中
CA
org1-ca
実行中
ピア
org1-peer1
実行中

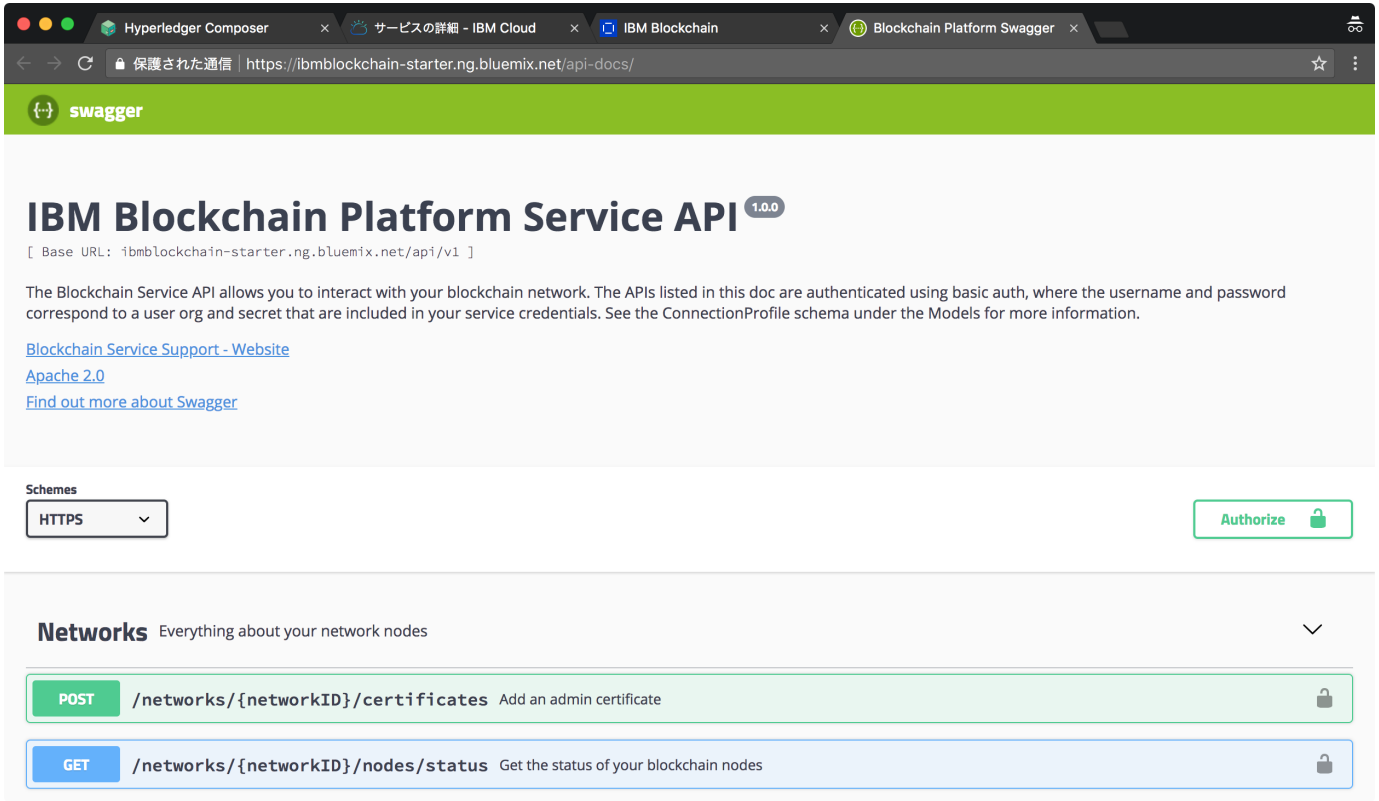
メンバーシップ管理


Hyperledger Composer
サービスの詳細 - IBM Cloud
IBM Blockchain
保護された通信 | https://ibmblockchain-starter.ng.bluemix.net/network/nc145d401663e4069a339a0b02d9d9d90/members
IBM ブロックチェーン Starter Plan (Beta)
SHIN SAITO
オペレーティング: Company A (org1)
Olympus Network - JfB
マイ・ネットワーク
概説
メンバー
チャネル
通知
API
マイ・コード
コードの作成
コードのインストール
サンプルの試行
ヘルプの利用
フィードバックの提供
メンバー
「メンバー」タブで、ネットワークのメンバー (組織) を表示および管理します。他の組織をネットワークに招待するか、またはデフォルトで所有している 2 つの組織に加えてさらに組織を追加することができます。「証明書」タブで、組織に関連付けられた管理証明書を表示および管理することもできます。詳しくはこちら
メンバー
証明書
メンバー (2/16)
MSP ID
要求側
状況
Company A
shinsa@jp.ibm.com
org1
--
参加済み
Company B
shinsa@jp.ibm.com
org2
--
参加済み

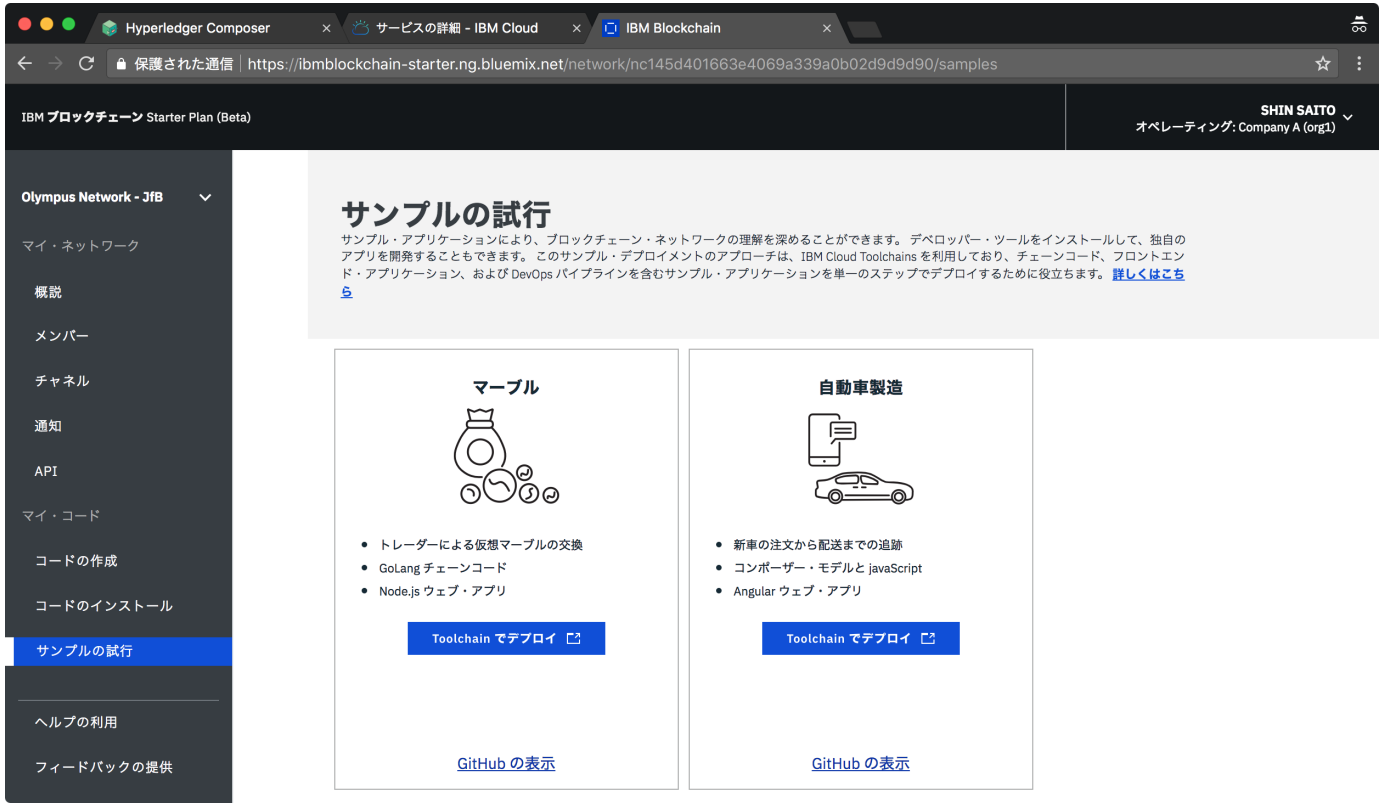
通知


Hyperledger Composer
サービスの詳細 - IBM Cloud
IBM Blockchain
保護された通信 | https://ibmblockchain-starter.ng.bluemix.net/network/nc145d401663e4069a339a0b02d9d9d90/notifications
IBM ブロックチェーン Starter Plan (Beta)
SHIN SAITO
オペレーティング: Company A (org1)
Olympus Network - JfB
マイ・ネットワーク
概説
メンバー
チャネル
通知
API
マイ・コード
コードの作成
コードのインストール
サンプルの試行
ヘルプの利用
フィードバックの提供
通知
自分が含まれているチャネルの作成要求または更新要求が送信されるたびに、通知を受け取ります。「アクション」の下にあるボタンを使用してチャネル要求を確認、表決、および送信します
すべて (1)
保留中 (0)
完了 (1)
通知を検索
名前
更新された日付
自分の状況
チャネル要求
"defaultchannel" に参加
担当者: undefined
2018 June 07 June, - 4:58:51 PM
表決終了

Web API


Hyperledger Composer
サービスの詳細 - IBM Cloud
IBM Blockchain
Blockchain Platform Swagger
保護された通信 | https://ibmblockchain-starter.ng.bluemix.net/api-docs/
swagger
IBM Blockchain Platform Service API
1.0.0
[Base URL: ibmblockchain-starter.ng.bluemix.net/api/v1]
The Blockchain Service API allows you to interact with your blockchain network. The APIs listed in this doc are authenticated using basic auth, where the username and password correspond to a user org and secret that are included in your service credentials. See the ConnectionProfile schema under the Models for more information.
Blockchain Service Support - Website
Apache 2.0
Find out more about Swagger
Schemes
HTTPS
Authorize
Networks
Everything about your network nodes
POST
/networks/{networkID}/certificates
Add an admin certificate
GET
/networks/{networkID}/nodes/status
Get the status of your blockchain nodes

サンプル・アプリケーション



参考文献・サイト・サービス

参考文献

ブロックチェーンの革新技術:
Hyperledger Fabricによる
アプリケーション開発
(リックテレコム)

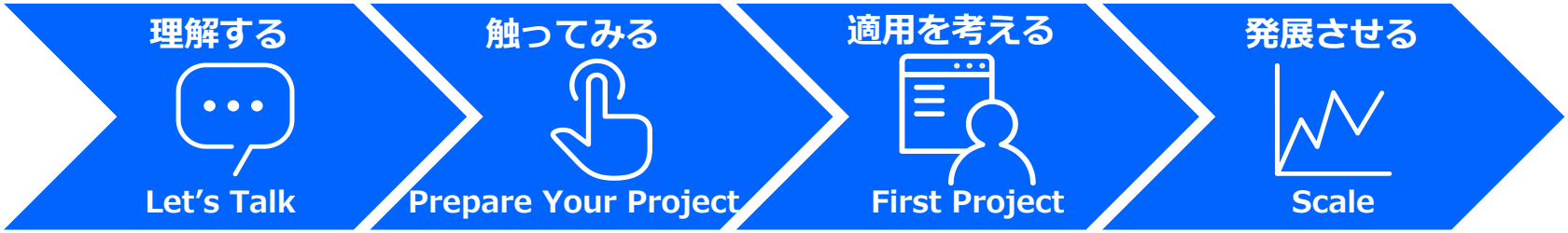
http://www.ric.co.jp/book/
contents/book_1146.html

参考Webサイト [1]

- IBM Blockchain Platform
  https://console.bluemix.net

- IBM Blockchain – Japan
  https://www.ibm.com/blockchain/jp-ja/
  ブロックチェーンのソリューション・事例を含む
  日本語情報サイト

ブロックチェーンの進め方



お客様の取り組み状況

ブロックチェーン技術や適用事例を理解	知識を身につけプロジェクト開始準備	ビジネス課題を定義し実証実験	新しいサービス提供に向け本格展開

IBMのご支援内容

ブロックチェーンご紹介セッション	スタートアップ・プログラム技術習得オファリング	IBM Cloud Garage	Industry Platform

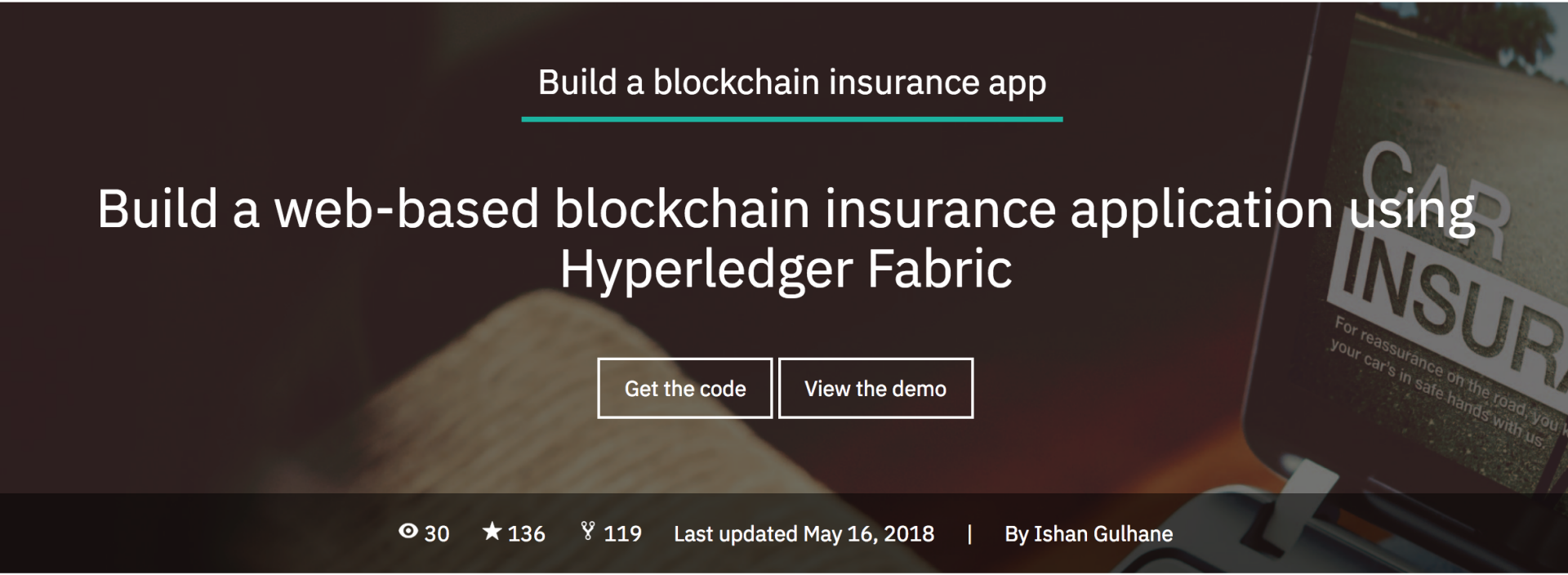
IBM Blockchain Platform (IBM Cloud)

開発者向けサンドボックス	フェーズや要件に応じた3つのプラン		
	Starter (Beta)	Enterprise(GA)	Enterprise Plus

無料

参考Webサイト [2]

- Hyperledger Fabric (for v1.1) http://hyperledger-fabric.readthedocs.io/en/release-1.1
- Hyperledger Composer (for v0.19) https://hyperledger.github.io/composer
- Zero to Blockchain https://github.com/rddill-IBM/ZeroToBlockchain



https://developer.ibm.com/code/patterns/build-a-blockchain-insurance-app/

まとめ

- Hyperledger Composerのコーディング例をPlayGroundで紹介
- 関連情報の紹介

気軽に始められます
ぜひアプリを開発してみてください！

ワークショップ、セッション、および資料は、IBMまたはセッション発表者によって準備され、それぞれ独自の見解を反映したものです。それらは情報提供の目的のみで提供されており、いかなる参加者に対しても法律的またはその他の指導や助言を意図したものではなく、またそのような結果を生むものでもありません。本講演資料に含まれている情報については、完全性と正確性を期するよう努力しましたが、「現状のまま」提供され、明示または暗示にかかわらずいかなる保証も伴わないものとします。本講演資料またはその他の資料の使用によって、あるいはその他の関連によって、いかなる損害が生じた場合も、IBMは責任を負わないものとします。 本講演資料に含まれている内容は、IBMまたはそのサプライヤーやライセンス交付者からいかなる保証または表明を引きだすことを意図したものでも、IBMソフトウェアの使用を規定する適用ライセンス契約の条項を変更することを意図したものでもなく、またそのような結果を生むものでもありません。

本講演資料でIBM製品、プログラム、またはサービスに言及していても、IBMが営業活動を行っているすべての国でそれらが使用可能であることを暗示するものではありません。本講演資料で言及している製品リリース日付や製品機能は、市場機会またはその他の要因に基づいてIBM独自の決定権をもっていつでも変更できるものとし、いかなる方法においても将来の製品または機能が使用可能になると確約することを意図したものではありません。本講演資料に含まれている内容は、参加者が開始する活動によって特定の販売、売上高の向上、またはその他の結果が生じると述べる、または暗示することを意図したものでも、またそのような結果を生むものでもありません。 パフォーマンスは、管理された環境において標準的なIBMベンチマークを使用した測定と予測に基づいています。ユーザーが経験する実際のスループットやパフォーマンスは、ユーザーのジョブ・ストリームにおけるマルチプログラミングの量、入出力構成、ストレージ構成、および処理されるワークロードなどの考慮事項を含む、数多くの要因に応じて変化します。したがって、個々のユーザーがここで述べられているものと同様の結果を得られると確約するものではありません。

記述されているすべてのお客様事例は、それらのお客様がどのようにIBM製品を使用したか、またそれらのお客様が達成した結果の実例として示されたものです。実際の環境コストおよびパフォーマンス特性は、お客様ごとに異なる場合があります。

IBM、IBM ロゴ、ibm.com、IBM Cloud、THINKは、 世界の多くの国で登録されたInternational Business Machines Corporationの商標です。他の製品名およびサービス名等は、それぞれIBMまたは各社の商標である場合があります。現時点での IBM の商標リストについては、www.ibm.com/legal/copytrade.shtmlをご覧ください。