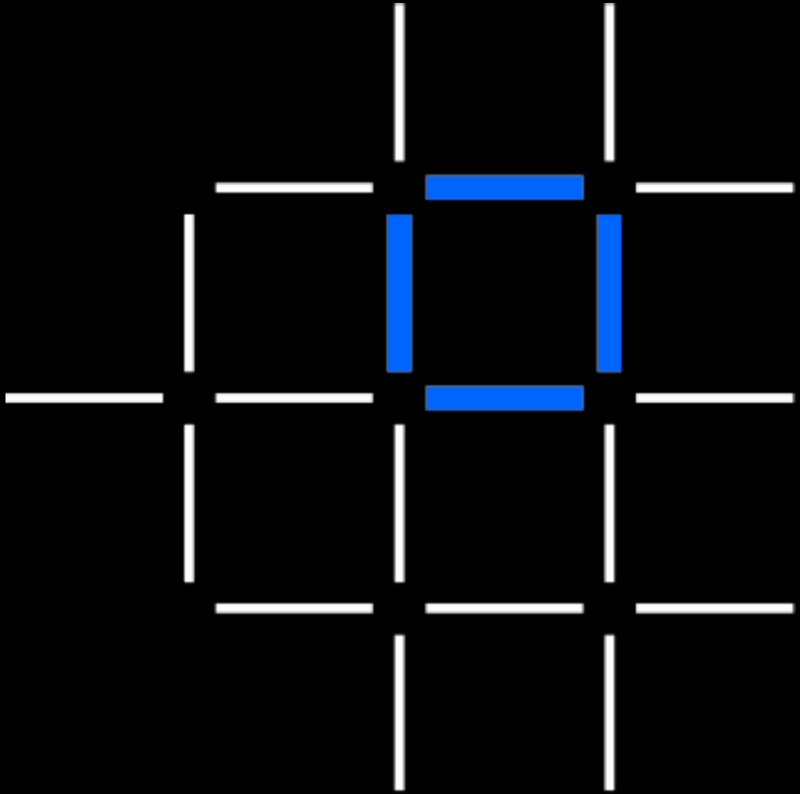
Using IBM Blockchain Platform

How to build, operate and grow blockchain networks

Barry Silliman
Blockchain Enablement on IBM Z and LinuxONE
IBM North America Technical Sales
silliman@us.ibm.com



24 January 2020

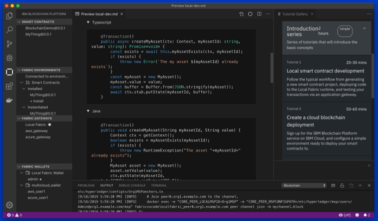
IBM **Blockchain**

The IBM Blockchain Platform **toolset**

- IBM Blockchain Platform comprises an intuitive set of tools for **building, operating and growing** Hyperledger Fabric networks

- The purpose of this presentation is not to guide you through every feature of the tools – you will find them intuitive!

- We will instead focus on useful tips, and things you might need to remember when using them



```
    @Transaction()
    public async createMyAsset(ctx: Context, myAssetId: string,
value: string): Promise<void> {
        const exists = await this.myAssetExists(ctx, myAssetId);
        if (exists) {
            throw new Error(`The my asset ${myAssetId} already
exists`);
        }
        const myAsset = new MyAsset();
        myAsset.value = value;
        const buffer = Buffer.from(JSON.stringify(myAsset));
        await ctx.stub.putState(myAssetId, buffer);
    }
```

```
    @Transaction()
    public void createMyAsset(String myAssetId, String value) {
        Context ctx = getContext();
        boolean exists = myAssetExists(myAssetId);
        if (exists) {
            throw new RuntimeException("The asset "+myAssetId+"
already exists");
        }
        MyAsset asset = new MyAsset();
        asset.setValue(value);
        ctx.putState(myAssetId,
```

Introduction series — 2 hours — simple

Series of tutorials that will introduce the basic concepts

Tutorial 1 — 20-30 mins

Local smart contract development

Follow the typical workflow from generating a new smart contract project, deploying code to the Local Fabric runtime, and testing your transactions via an application gateway.

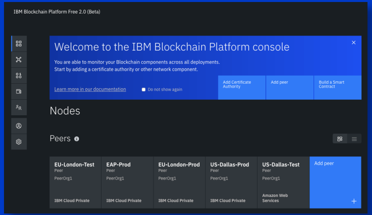
Tutorial 2 — 50-60 mins

Create a cloud blockchain deployment

Sign up for the IBM Blockchain Platform service on IBM Cloud, and configure a simple environment ready to deploy your smart contracts to.

```
etc/hyperledger/configtx/Org1MSPanchors.tx
[9/16/2019 5:59:20 PM] [INFO]     # Join peer0.org1.example.com to the channel.
[9/16/2019 5:59:20 PM] [INFO]     docker exec -e "CORE_PEER_LOCALMSPID=Org1MSP" -e "CORE_PEER_MSPCONFIGPATH=/etc/hyperledger/msp/users/
Admin@org1.example.com/msp" fabricvscodelocalfabric_peer0.org1.example.com peer channel join -b mychannel.block
```

VS Code
IBM Blockchain Platform Extension


IBM Blockchain Platform Free 2.0 (Beta)
Add Certificate Authority
Add peer
Build a Smart Contract
Learn more in our documentation
Do not show again
Add peer

Welcome to the IBM Blockchain Platform console

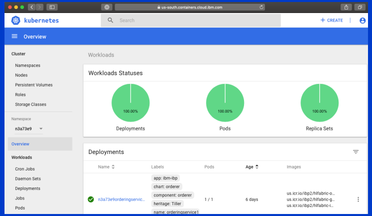
You are able to monitor your Blockchain components across all deployments.
Start by adding a certificate authority or other network component.

Nodes

Peers

EU-London-Test	EAP-Prod	EU-London-Prod	US-Dallas-Prod	US-Dallas-Test
Peer	Peer	Peer	Peer	Peer
PeerOrg1	PeerOrg1	PeerOrg1	PeerOrg1	PeerOrg1
IBM Cloud Private	IBM Cloud Private	IBM Cloud Private	IBM Cloud Private	Amazon Web Services

IBM Blockchain Platform **Console**



Deployments

Name	Labels	Pods	Age	Images
n3a73e9orderingservic...	app: ibm-ibp chart: orderer component: orderer heritage: Tiller name: orderingservice1	1 / 1	6 days	us.icr.io/ibp2/hlfabric-o... us.icr.io/ibp2/hlfabric-g... us.icr.io/ibp2/hlfabric-i...

Kubernetes
Dashboard

VS Code

Using the IBM Blockchain Platform VS Code extension

Web Console

Using the IBM Blockchain Platform network console

Your first network

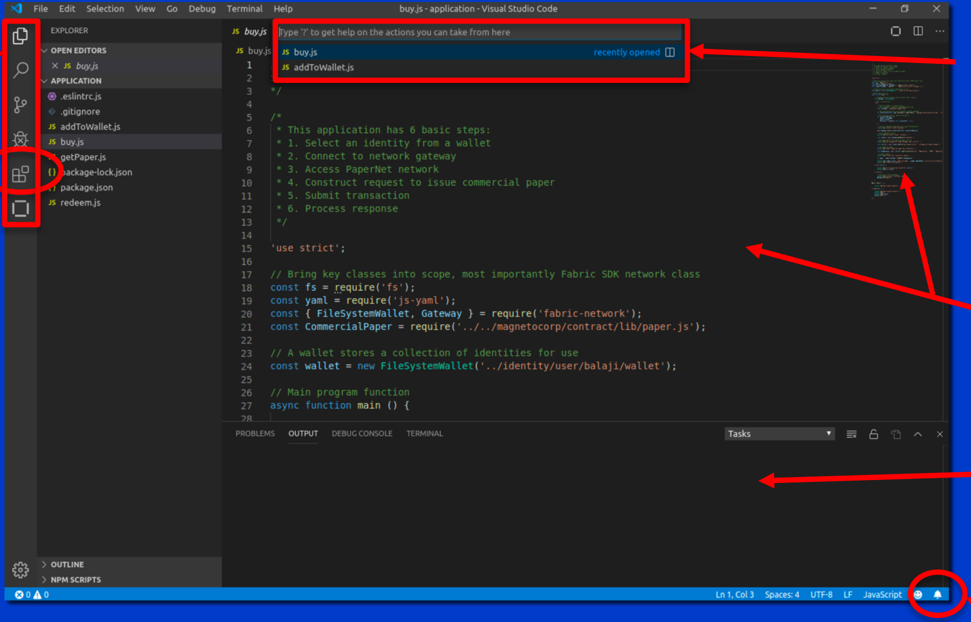
How to build and grow an IBM Blockchain Platform network

Navigating **VS Code**

The basics of this powerful, popular editor

Click in sidebar to expand and contract contextual set of panes

Marketplace for installing extensions



```
*/

/*
 * This application has 6 basic steps:
 * 1. Select an identity from a wallet
 * 2. Connect to network gateway
 * 3. Access PaperNet network
 * 4. Construct request to issue commercial paper
 * 5. Submit transaction
 * 6. Process response
 */

'use strict';

// Bring key classes into scope, most importantly Fabric SDK network class
const fs = require('fs');
const yaml = require('js-yaml');
const { FileSystemWallet, Gateway } = require('fabric-network');
const CommercialPaper = require('../../magnetocorp/contract/lib/paper.js');

// A wallet stores a collection of identities for use
const wallet = new FileSystemWallet('../identity/user/balaji/wallet');

// Main program function
async function main () {
```

Parameters for commands entered here.
Also used for quick navigation (Ctrl+P)

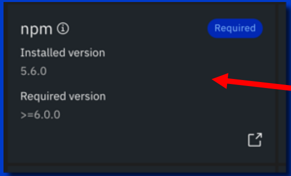
File editor and overview

Output, debug, terminal etc.

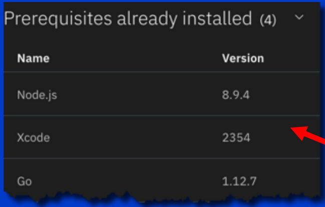
Notifications

Setting up the IBM Blockchain Platform **VS Code Extension**

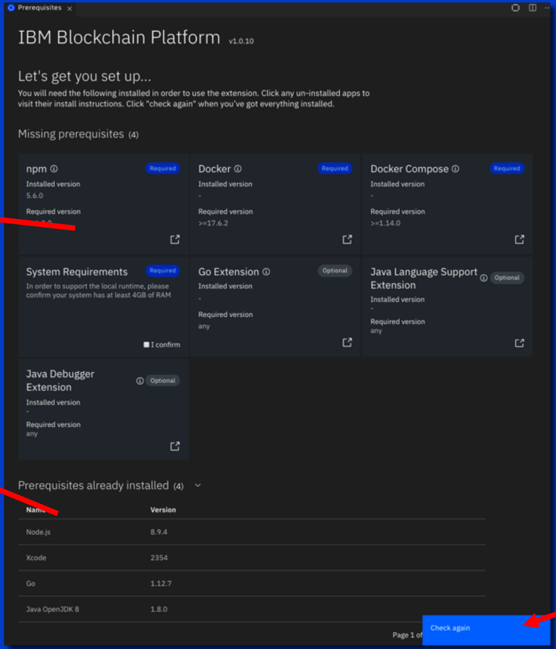
Required / optional components


npm ⓘ Required
Installed version
5.6.0
Required version
>=6.0.0

What you already have


Prerequisites already installed (4)

Name	Version
Node.js	8.9.4
Xcode	2354
Go	1.12.7



IBM Blockchain Platform v1.0.10

Let's get you set up...

You will need the following installed in order to use the extension. Click any un-installed apps to visit their install instructions. Click "check again" when you've got everything installed.

Missing prerequisites (4)

npm ⓘ Required
Installed version
5.6.0
Required version

Docker ⓘ Required
Installed version
-
Required version
>=17.6.2

Docker Compose ⓘ Required
Installed version
-
Required version
>=1.14.0

System Requirements Required
In order to support the local runtime, please confirm your system has at least 4GB of RAM
☐ I confirm

Go Extension ⓘ Optional
Installed version
-
Required version
any

Java Language Support Extension ⓘ Optional
Installed version
-
Required version
any

Java Debugger Extension ⓘ Optional
Installed version
-
Required version
any

Prerequisites already installed (4)

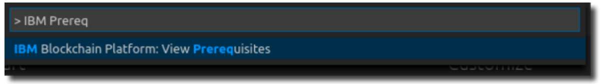
Name	Version
Node.js	8.9.4
Xcode	2354
Go	1.12.7
Java OpenJDK 8	1.8.0

Page 1 of

1. Install VS Code
2. Click Marketplace in sidebar
3. Search Marketplace for ”IBM Blockchain Platform”
4. Review and fix any pre-requisites
5. **Start!**

When you're ready, click here to begin

Tip: Ctrl+P "> IBM Prereq" will return to the pre-req checker once you're running

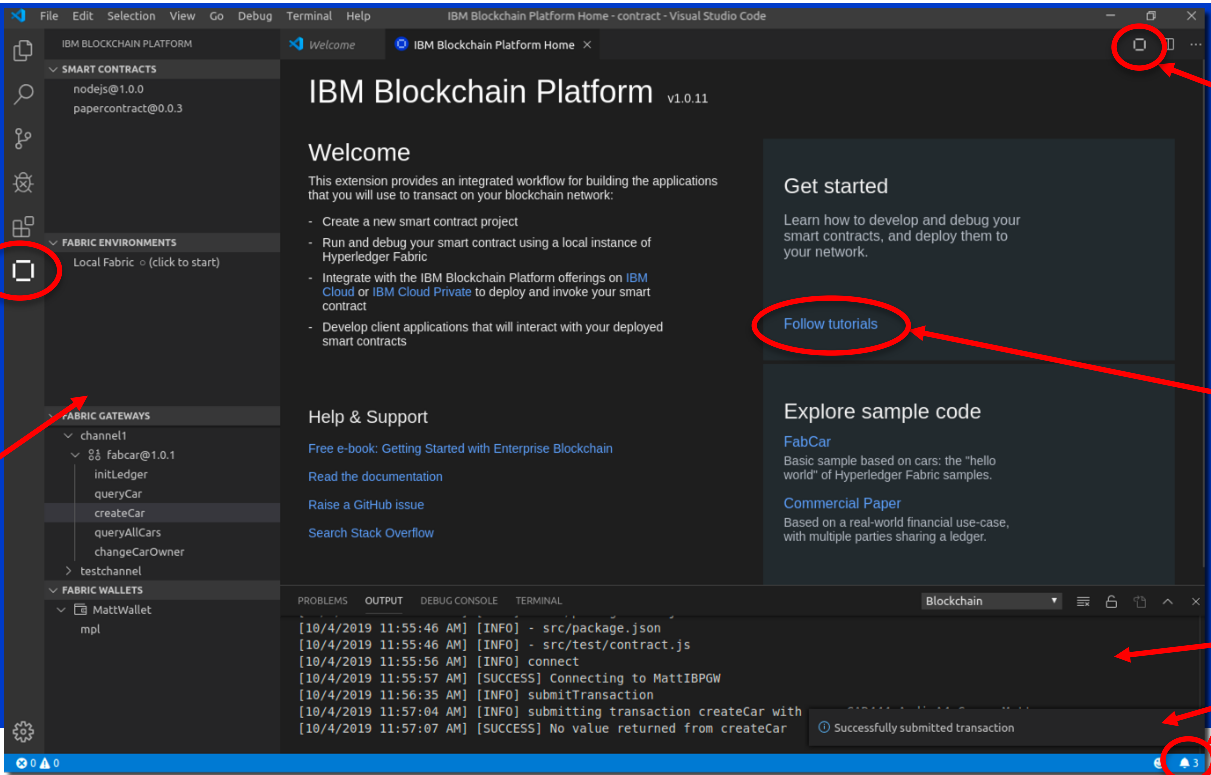

> IBM Prereq
IBM Blockchain Platform: View Prerequisites

Navigating the **VS Code Extension**

This is what you see when you launch the IBM Blockchain Platform extension for the first time

Switch to IBM Blockchain Platform view

<u>Views:</u>
Smart contracts
Fabric Environments
Gateways
Wallets



IBM Blockchain Platform v1.0.11

Welcome

This extension provides an integrated workflow for building the applications that you will use to transact on your blockchain network:

- Create a new smart contract project
- Run and debug your smart contract using a local instance of Hyperledger Fabric
- Integrate with the IBM Blockchain Platform offerings on IBM Cloud or IBM Cloud Private to deploy and invoke your smart contract
- Develop client applications that will interact with your deployed smart contracts

Help & Support

Free e-book: Getting Started with Enterprise Blockchain

Read the documentation

Raise a GitHub issue

Search Stack Overflow

Get started

Learn how to develop and debug your smart contracts, and deploy them to your network.

Follow tutorials

Explore sample code

FabCar

Basic sample based on cars: the "hello world" of Hyperledger Fabric samples.

Commercial Paper

Based on a real-world financial use-case, with multiple parties sharing a ledger.

```
[10/4/2019 11:55:46 AM] [INFO] - src/package.json
[10/4/2019 11:55:46 AM] [INFO] - src/test/contract.js
[10/4/2019 11:55:56 AM] [INFO] connect
[10/4/2019 11:55:57 AM] [SUCCESS] Connecting to MattIBPGW
[10/4/2019 11:56:35 AM] [INFO] submitTransaction
[10/4/2019 11:57:04 AM] [INFO] submitting transaction createCar with
[10/4/2019 11:57:07 AM] [SUCCESS] No value returned from createCar
```

ⓘ Successfully submitted transaction

Click here (Or Ctrl+P "> IBM Home") to return to this Home page in the future if you need to

Tutorials here!

Output from running commands etc.

Notifications

IBM **Blockchain**

IBM

Developing Smart Contracts: Concepts

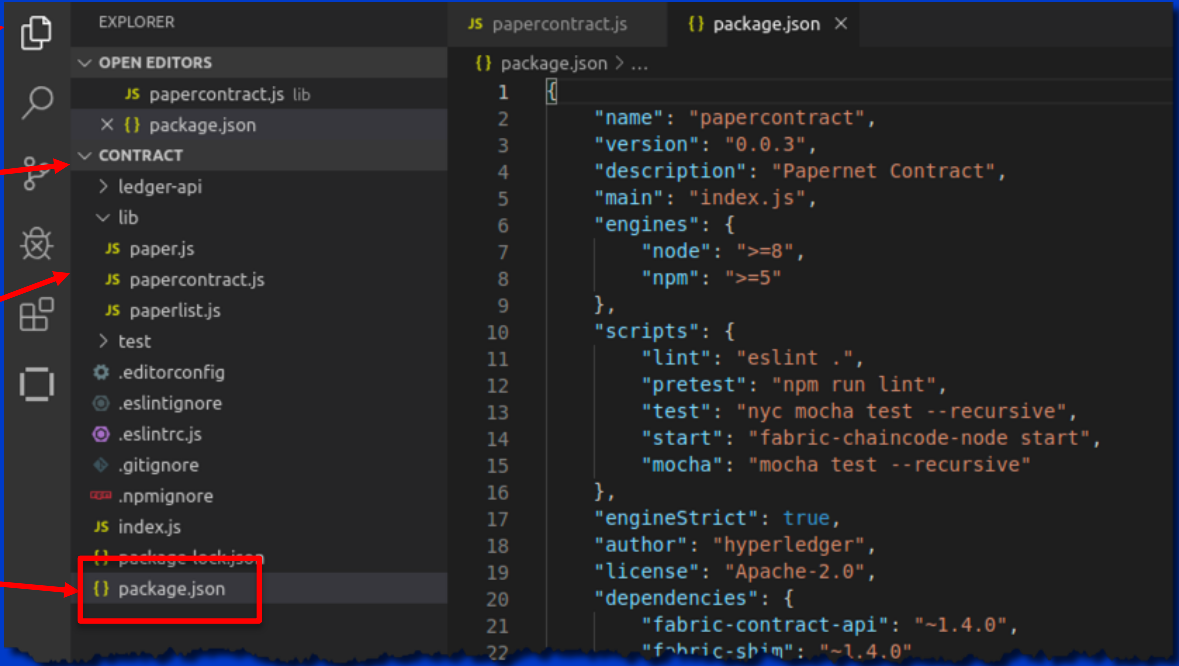
The structure of a smart contract project

Use Explorer view when editing

Folder view (from filesystem)

Smart contract code implements Contract interface

package.json describes smart contract



```
{
    "name": "papercontract",
    "version": "0.0.3",
    "description": "Papernet Contract",
    "main": "index.js",
    "engines": {
        "node": ">=8",
        "npm": ">=5"
    },
    "scripts": {
        "lint": "eslint .",
        "pretest": "npm run lint",
        "test": "nyc mocha test --recursive",
        "start": "fabric-chaincode-node start",
        "mocha": "mocha test --recursive"
    },
    "engineStrict": true,
    "author": "hyperledger",
    "license": "Apache-2.0",
    "dependencies": {
        "fabric-contract-api": "~1.4.0",
        "fabric-shim": "~1.4.0"
```

IBM **Blockchain**

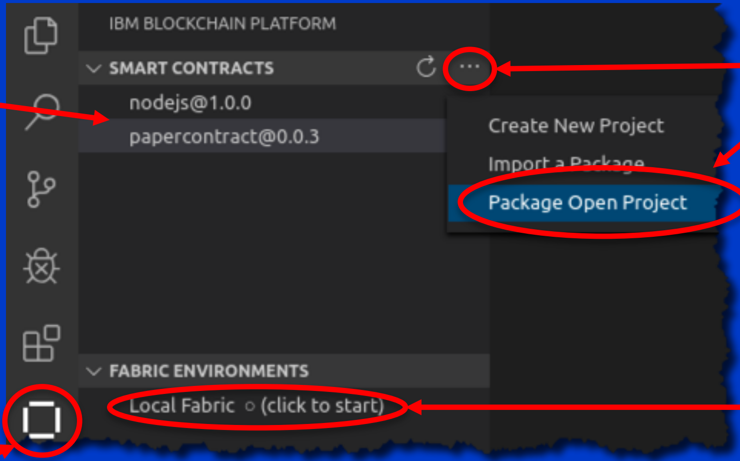
Tip: FabCar and Commercial Paper are good exemplars!

Testing Smart Contracts [1/2]

The basics for smart contract testing

Right click smart contracts to export as .cds file

Use IBM Blockchain Platform view when deploying



Hover here to reveal "…" and select Package Open Project.
This builds the smart contract package

Click here to start an embedded local Hyperledger Fabric instance

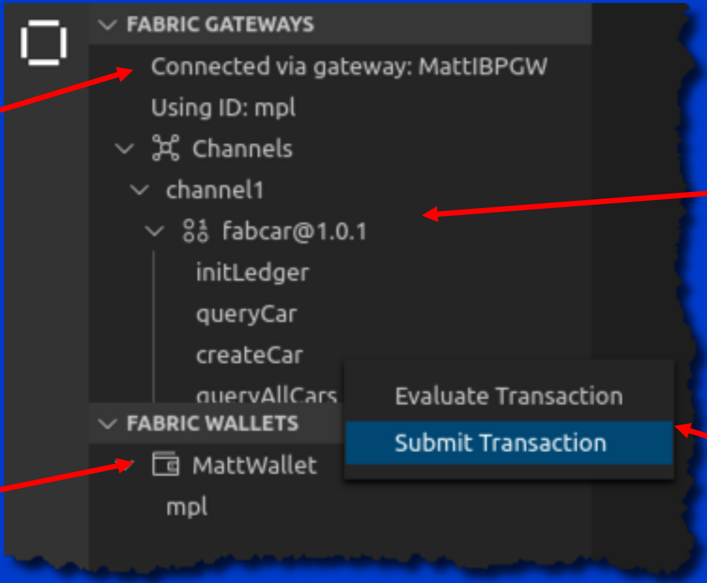
Install and instantiate your smart contract

Testing Smart Contracts [2/2]

The basics for smart contract testing

Connect to gateways for working with local or remote blockchain networks

Wallets show available identities, used for connecting to remote networks


FABRIC GATEWAYS
Connected via gateway: MattIBPGW
Using ID: mpl
Channels
channel1
fabcar@1.0.1
initLedger
queryCar
createCar
queryAllCars
Evaluate Transaction
Submit Transaction
FABRIC WALLETS
MattWallet
mpl

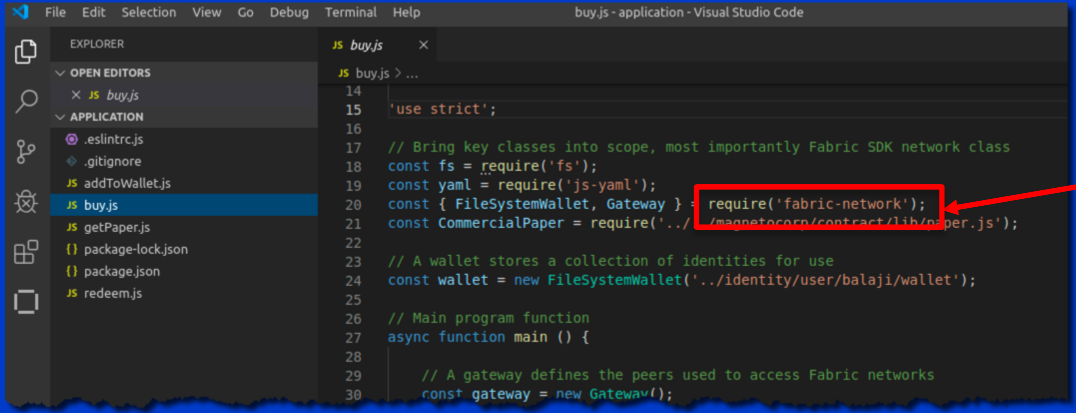
Shows available channels and smart contracts

Right click to submit / evaluate transactions without requiring a client application

Building **Applications**

The basics for client application development

- While you can submit and evaluate transactions from VS Code, real blockchain use-cases will require client applications to interact with the ledger.



```
'use strict';

// Bring key classes into scope, most importantly Fabric SDK network class
const fs = require('fs');
const yaml = require('js-yaml');
const { FileSystemWallet, Gateway } = require('fabric-network');
const CommercialPaper = require('../magnetocorp/contract/lib/paper.js');

// A wallet stores a collection of identities for use
const wallet = new FileSystemWallet('../identity/user/balaji/wallet');

// Main program function
async function main () {

    // A gateway defines the peers used to access Fabric networks
    const gateway = new Gateway();
```

Use Fabric SDK

- You can create and test these client applications from VS Code too, just like any other development project.
    - Test within VS Code, command line or whatever environment you choose

VS Code

Using the IBM Blockchain Platform VS Code extension

Web Console

Using the IBM Blockchain Platform network console

Your first network

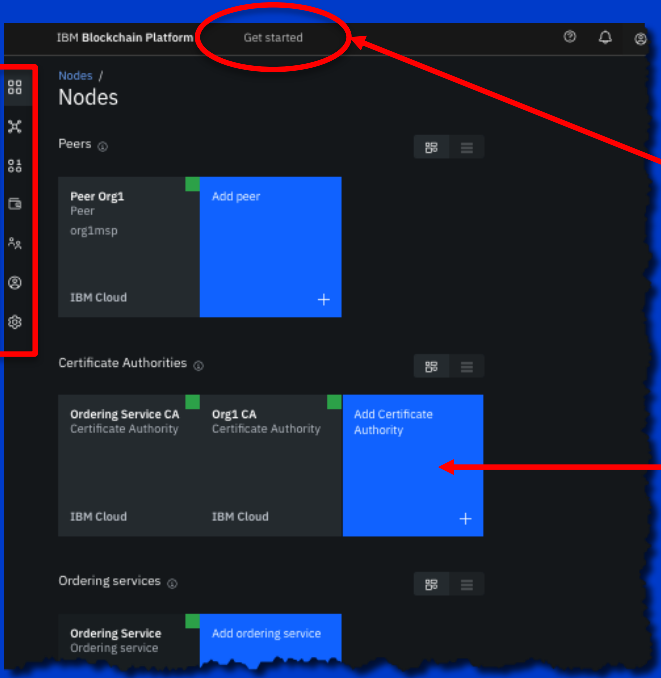
How to build and grow an IBM Blockchain Platform network

Navigating the **IBM Blockchain Platform Console**

This is what you see when you launch the IBM Blockchain Platform service

Sidebar:

- Nodes (selected)
- Channels
- Smart contracts
- Wallets
- Organizations
- Users
- Settings



Tutorials, e.g.

- Build, Join
- Develop, Deploy

Actions to create or add things are always shown in blue

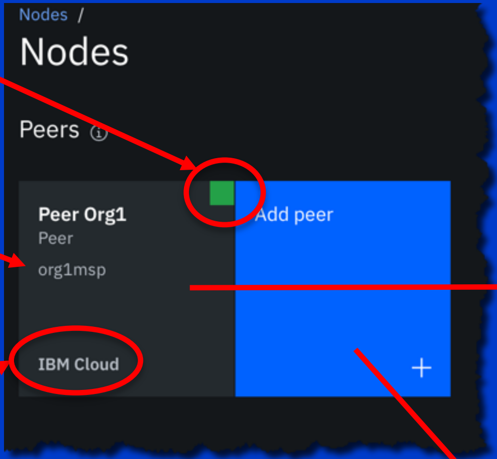
Working with **nodes** in the console

Manage peers, certificate authorities and ordering services from the same pane

Green square = running

ID of owning organization

Deployment location



Nodes /

Nodes

Peers

Peer Org1
Peer
org1msp

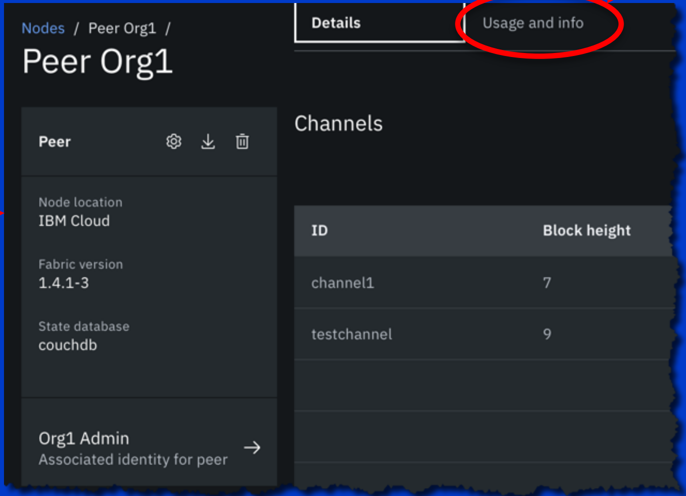
IBM Cloud

Add peer

Select for more details

Create and import additional nodes

Check here for resource utilization



Nodes / Peer Org1 /

Peer Org1

Details | Usage and info

Peer

Node location
IBM Cloud

Fabric version
1.4.1-3

State database
couchdb

Org1 Admin
Associated identity for peer

Channels

ID	Block height
channel1	7
testchannel	9

CA

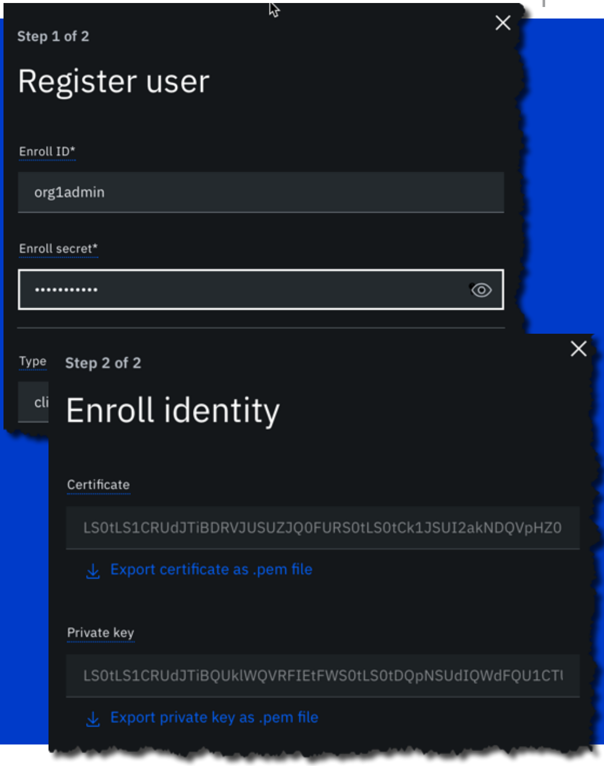
The importance of **identities**

Registering and enrolling using certificate authorities

- Managing identity is a **critical part** of a network
  - All users and components have an identity
  - These are managed in the console under the CA node
  - Make a note of what identities are used where; avoid reuse

- Two step process helps ensure admins can't hijack identities
  1. CA admin **registers** the identity in the CA with an enroll ID and secret; passes details to identity owner
  2. Owner **enrolls** the identity using these details (e.g. when creating nodes); certificates are generated for the owner to work with.

- Certificates are stored in wallets and stay in local browser storage by default
  - Certificates can move between wallets but are not managed by IBM.
  - Take care when switching browsers!


Step 1 of 2

Register user

Enroll ID*

org1admin

Enroll secret*

•••••••••••

Type

cli

Step 2 of 2

Enroll identity

Certificate

LS0tLS1CRUdJTiBDRVJUSUZJQ0FURS0tLS0tCk1JSUI2akNDQVpHZ0

Export certificate as .pem file

Private key

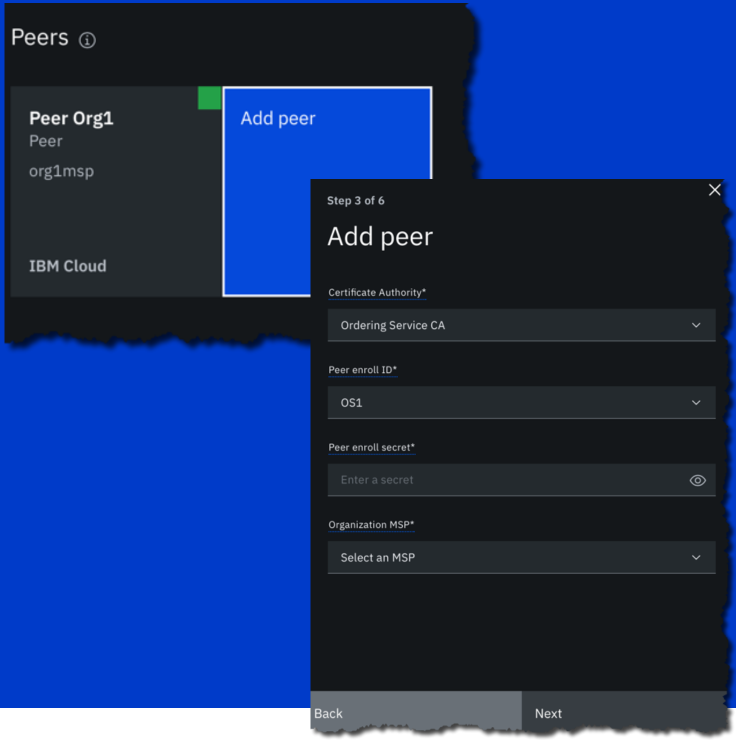
LS0tLS1CRUdJTiBQUklWQVRFIEtFWS0tLS0tDQpNSUdIQWdFQU1CTl

Export private key as .pem file

P1

Example: Creating a **peer** node

- Select *Add peer -> Create new*
- Several panels guide you through the new peer's details
  - Display name
  - **Peer's identity** to enroll (CA, peer enroll ID, secret)
  - Owning organization
  - **TLS identity** (for secure communication)
  - **Administrator's identity**
  - Additional options cater for advanced deployment options (e.g. CouchDB vs. LevelDB)
- Peer is then created and started automatically
- See the next section for a full build tutorial!


Peers

Peer Org1
Peer
org1msp
IBM Cloud

Add peer

Step 3 of 6
Add peer

Certificate Authority*
Ordering Service CA

Peer enroll ID*
OS1

Peer enroll secret*
Enter a secret

Organization MSP*
Select an MSP

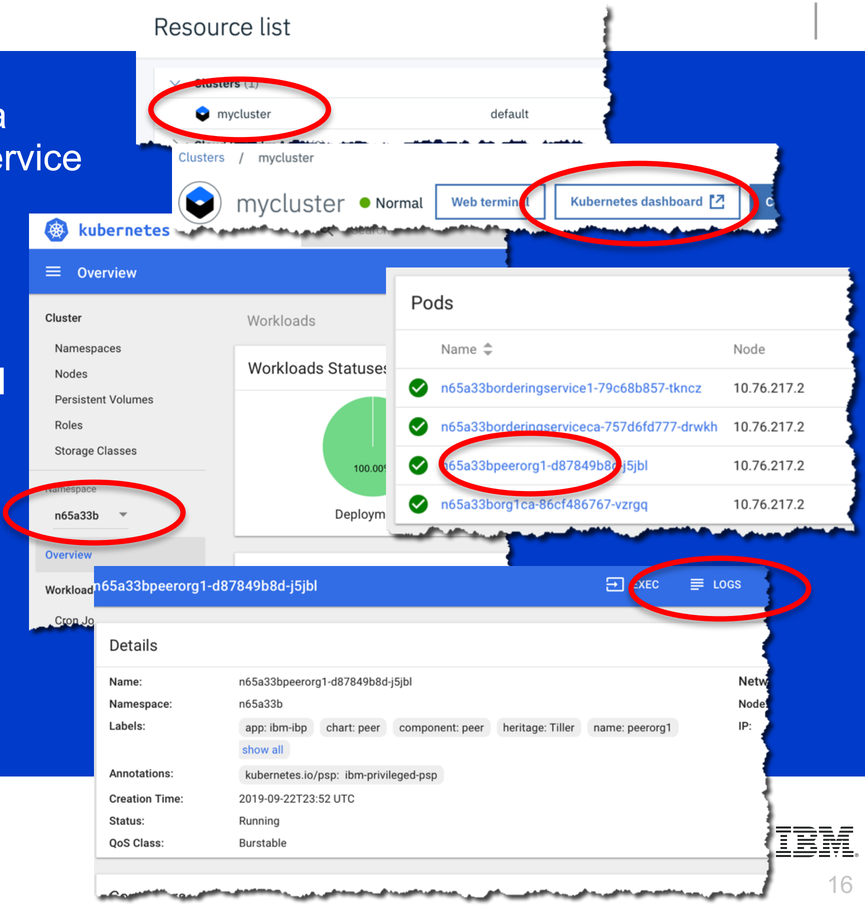
Back
Next

Logging and monitoring

Using Kubernetes to drill into the details

- Each IBM Blockchain Platform component is run within a docker container and managed through a Kubernetes service
  - IBM Kubernetes Service (for components on IBM Cloud) or OpenShift (for components deployed on other clouds or on-premises)

- Access component details from **Kubernetes dashboard**
  1. Select cluster from IBM Cloud Dashboard
  2. Click Kubernetes dashboard
  3. Select nXXXXXX namespace
  4. Click Pods to see components
  5. Select individual pods for further details
     - IP addresses, parameters, storage etc.
  6. Click Logs for debug information



Name	Node
n65a33borderingservice1-79c68b857-tkncz	10.76.217.2
n65a33borderingserviceca-757d6fd777-drwkh	10.76.217.2
n65a33bpeerorg1-d87849b8d-j5jbl	10.76.217.2
n65a33borg1ca-86cf486767-vzrgq	10.76.217.2

n65a33bpeerorg1-d87849b8d-j5jbl

Details

Name:	n65a33bpeerorg1-d87849b8d-j5jbl
Namespace:	n65a33b
Labels:	app: ibm-ibp; chart: peer; component: peer; heritage: Tiller; name: peerorg1; show all
Annotations:	kubernetes.io/psp: ibm-privileged-psp
Creation Time:	2019-09-22T23:52 UTC
Status:	Running
QoS Class:	Burstable

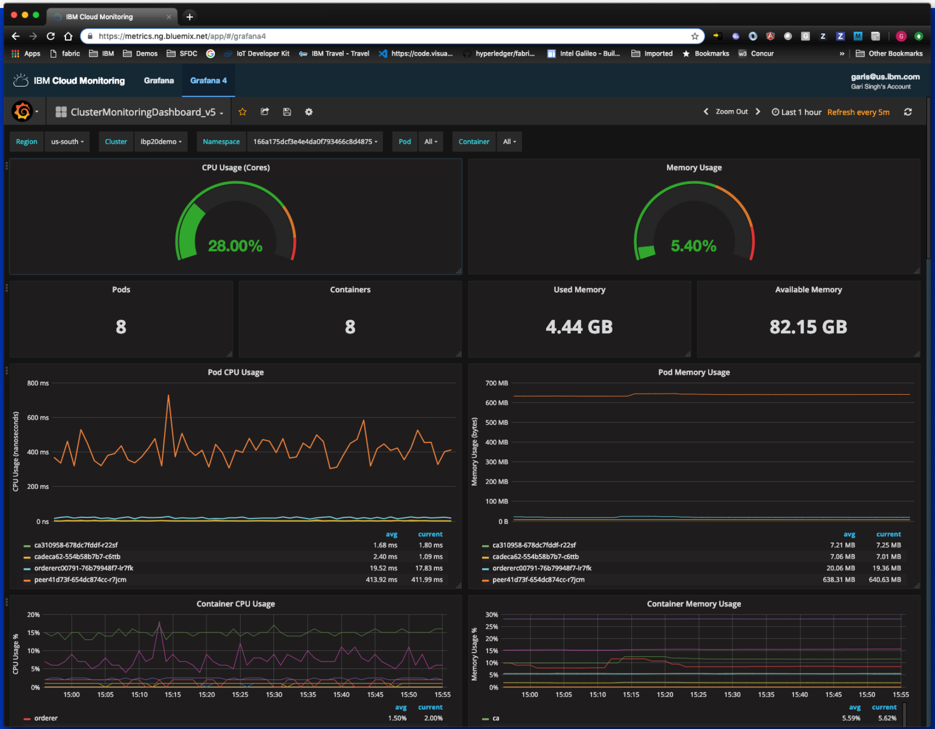
Netw
Node
IP:

Logging and monitoring

Exporting metrics for further analysis

- There are several monitoring tools which can extract and visualize blockchain metrics, e.g.:
  - **Grafana** is a general purpose dashboard and graph composer, available at metrics.[region.]bluemix.net
  - **Sysdig** is an systems monitoring and troubleshooting service, available on IBM Cloud
  - **Prometheus** is a monitoring and alerting toolkit that aggregates time series data

- IBM Blockchain Platform exposes a /metrics endpoint that can be used by tools for this purpose.



Pod CPU Usage

	avg	current
ca310958-678dc7fddf-r22sf	1.68 ms	1.80 ms
cadeca62-554b58b7b7-c6ttb	2.40 ms	1.09 ms
ordererc00791-76b79948f7-lr7fk	19.52 ms	17.83 ms
peer41d73f-654dc874cc-r7jcm	413.92 ms	411.99 ms

Pod Memory Usage

	avg	current
ca310958-678dc7fddf-r22sf	7.21 MB	7.25 MB
cadeca62-554b58b7b7-c6ttb	7.06 MB	7.01 MB
ordererc00791-76b79948f7-lr7fk	20.06 MB	19.36 MB
peer41d73f-654dc874cc-r7jcm	638.31 MB	640.63 MB

Container CPU Usage

	avg	current
orderer	1.50%	2.00%

Container Memory Usage

	avg	current
ca	5.59%	5.62%

VS Code

Using the IBM Blockchain Platform VS Code extension

Web Console

Using the IBM Blockchain Platform network console

[

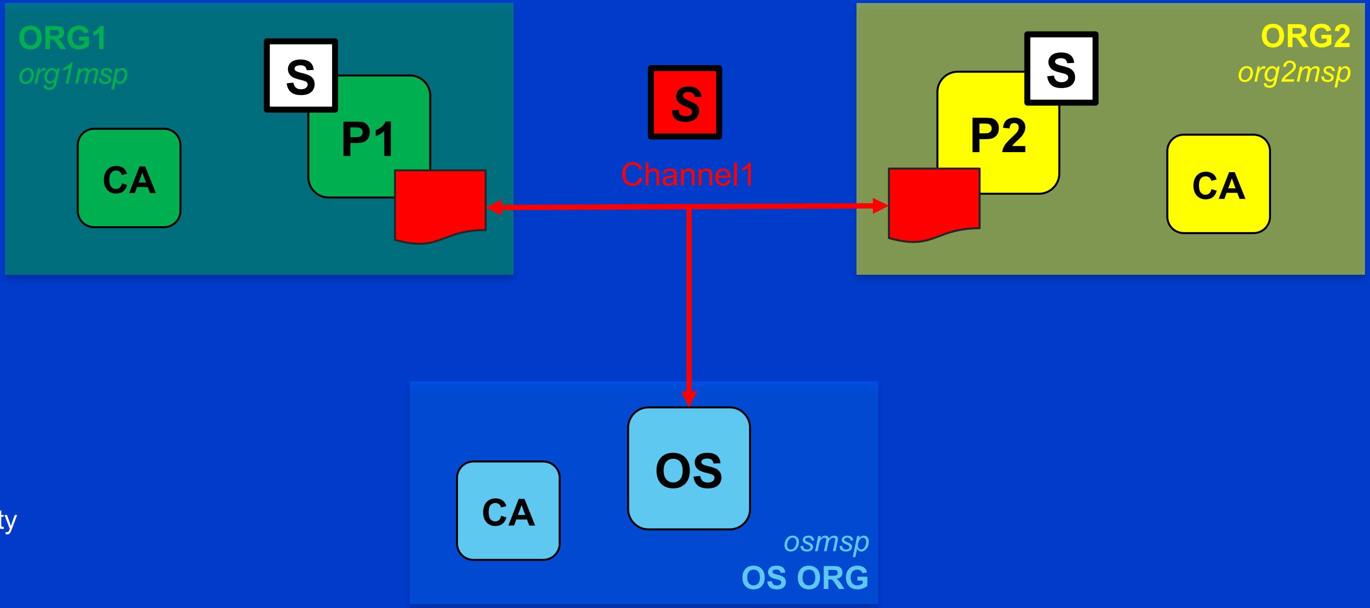
Your first network

How to build and grow an IBM Blockchain Platform network

]

What we are building

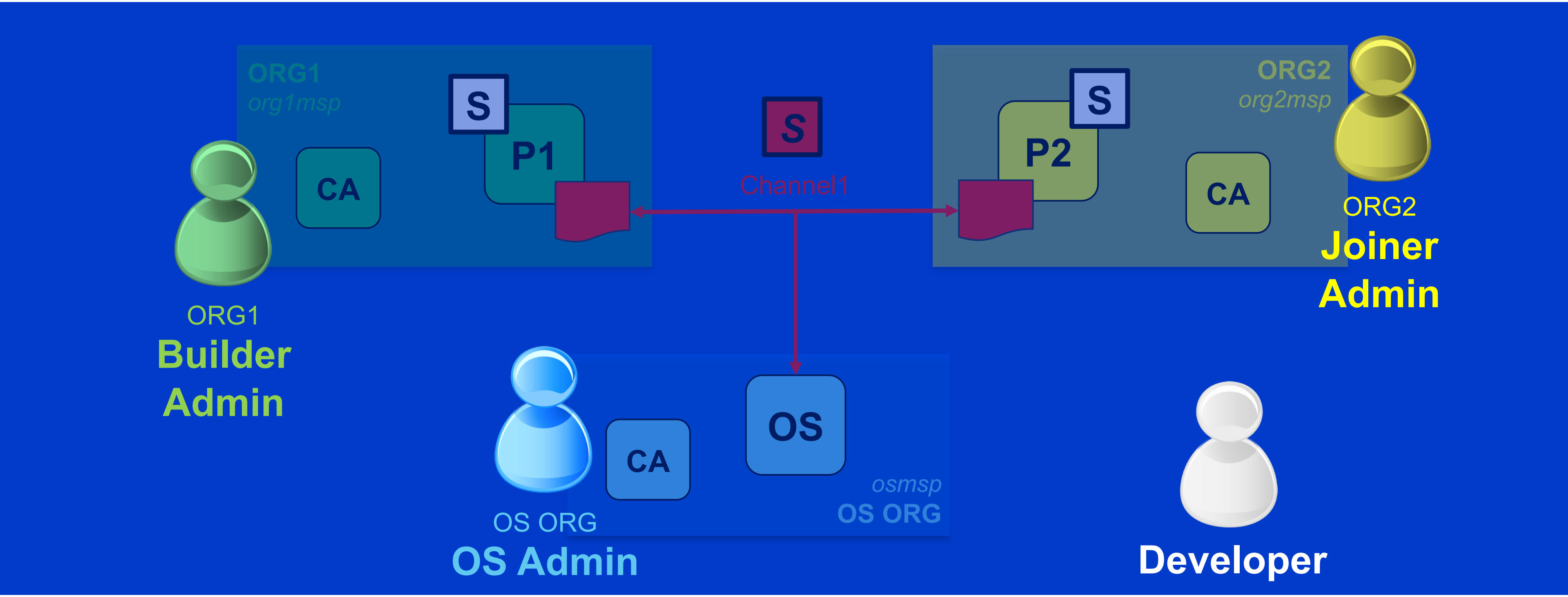
The target IBM Blockchain Platform environment



CA: Certificate Authority
P: Peer
S: Smart Contract
OS: Ordering Service
msp: Membership Services Provider (identifies the organization on the network)

Who is building the network

The organizations and roles



How we are building it

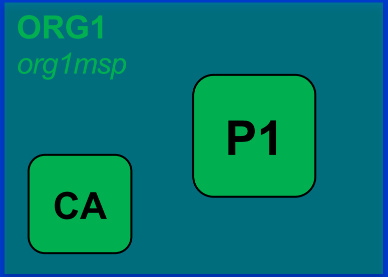
The high level sequence of steps

1. **Build** the network
   - Create an ordering service, peer, channel and first CAs
2. **Join** the network
   - Repeat for each additional organization in the consortium
3. **Deploy** smart contracts
   - And test transactions to make sure everything works

Building a network [1/5]

Detailed tutorial at IBM Blockchain Platform Console -> Get Started -> Build a Network

1. **Builder Admin** creates a peer organization and peer
   - Create the peer organization CA
   - Associate the CA admin identity
   - Using the CA to register identities
   - Create the peer organization MSP definition
   - Create the peer

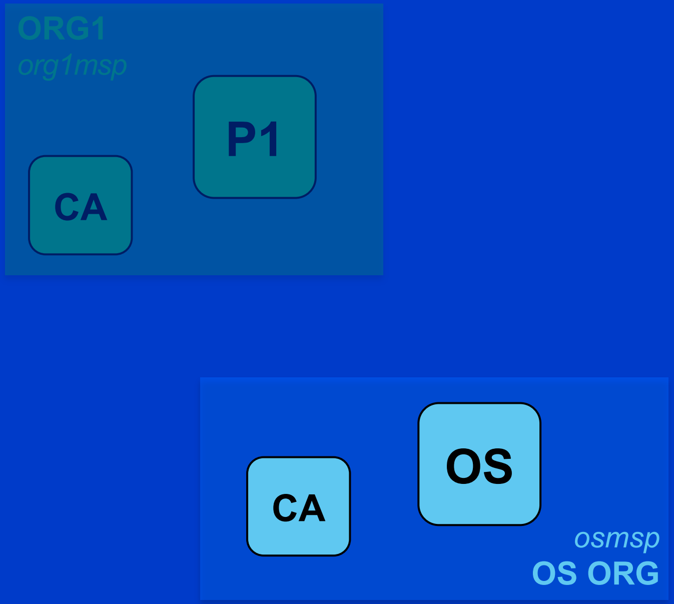


Building a network [2/5]

Detailed tutorial at IBM Blockchain Platform Console -> Get Started -> Build a Network

1. **Builder Admin** creates a peer organization and peer

2. **OS Admin** creates the ordering service
    - Create the ordering service organization CA
    - Associate the CA admin identity
    - Use the CA to register the ordering service node + OS Admin identities
    - Create the ordering service organization MSP definition
    - Deploy the ordering nodes



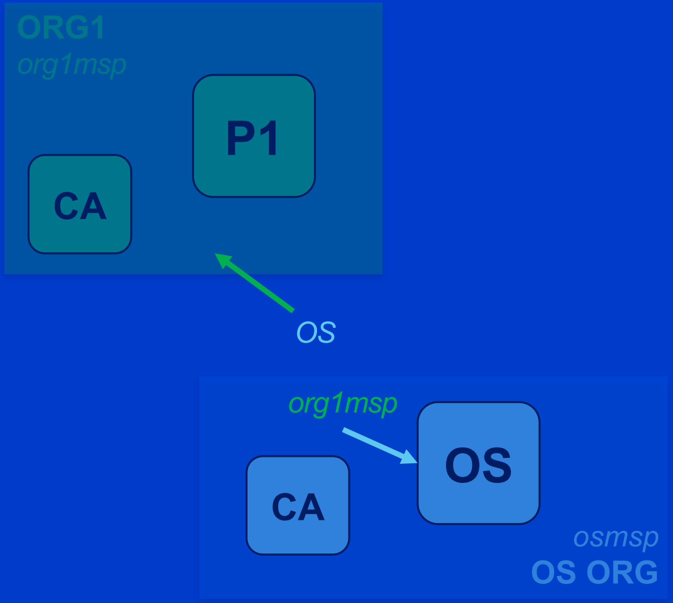
Building a network [3/5]

Detailed tutorial at IBM Blockchain Platform Console -> Get Started -> Build a Network

1. **Builder Admin** creates a peer organization and peer

2. **OS Admin** creates the ordering service

3. **OS Admin** adds **ORG1** to the consortium hosted by the ordering service *
   1. **Builder Admin** exports the ORG1 information and sends to the **OS Admin**
   2. **OS Admin** imports the ORG1 definition into the ordering service and add its peer's org to the OS
   3. **OS Admin** exports the OS definition
   4. **Builder Admin** imports the OS definition



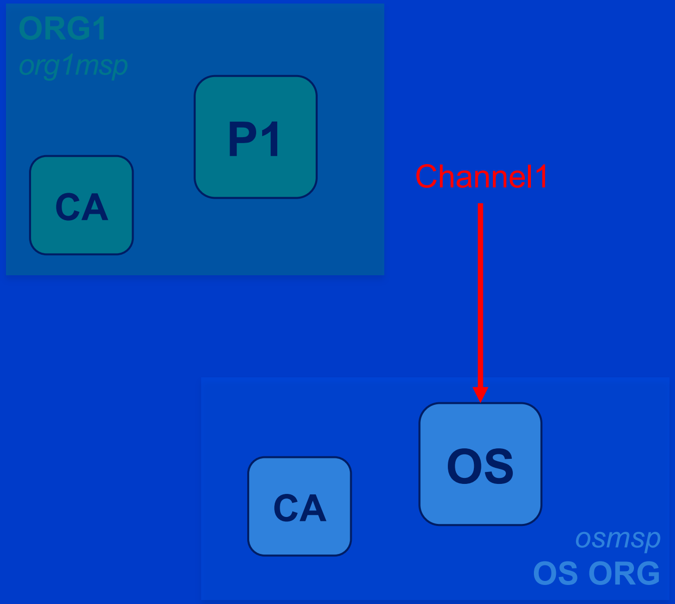
IBM **Blockchain**

* If the **OS Admin** and **Builder Admin** are running in the same console, the sub-steps are not required. The console will let you simply add ORG1 directly to the Ordering Service.

Building a network [4/5]

Detailed tutorial at IBM Blockchain Platform Console -> Get Started -> Build a Network

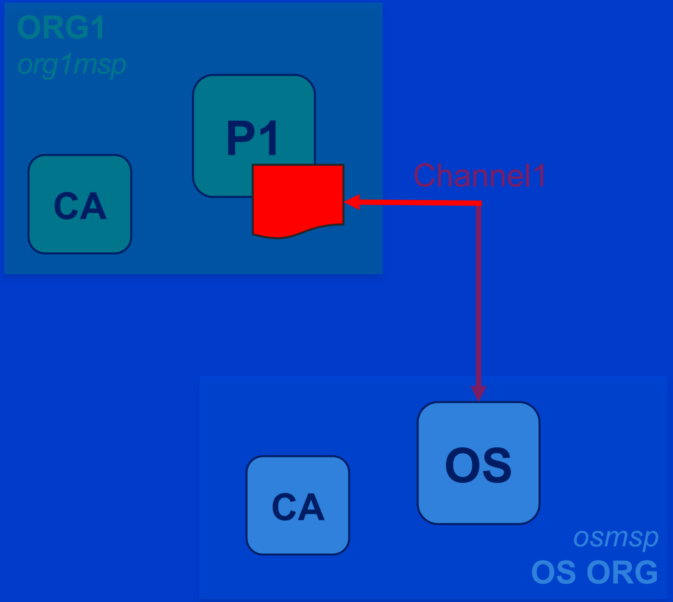
1. **Builder Admin** creates a peer organization and peer
2. **OS Admin** creates the ordering service
3. **OS Admin** adds **ORG1** to the consortium hosted by the ordering service
4. **Builder Admin** creates a channel



Building a network [5/5]

Detailed tutorial at IBM Blockchain Platform Console -> Get Started -> Build a Network

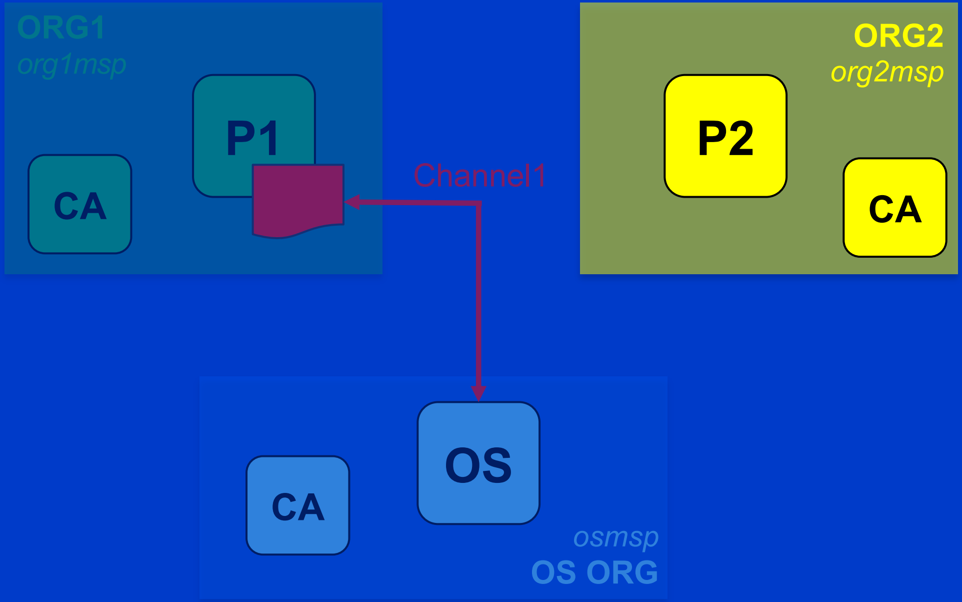
1. **Builder Admin** creates a peer organization and peer
2. **OS Admin** creates the ordering service
3. **OS Admin** adds **ORG1** to the consortium hosted by the ordering service
4. **Builder Admin** creates a channel
5. **Builder Admin** joins the peer to the channel



Joining a network [1/4]

Detailed tutorial at IBM Blockchain Platform Console -> Get Started -> Join a Network

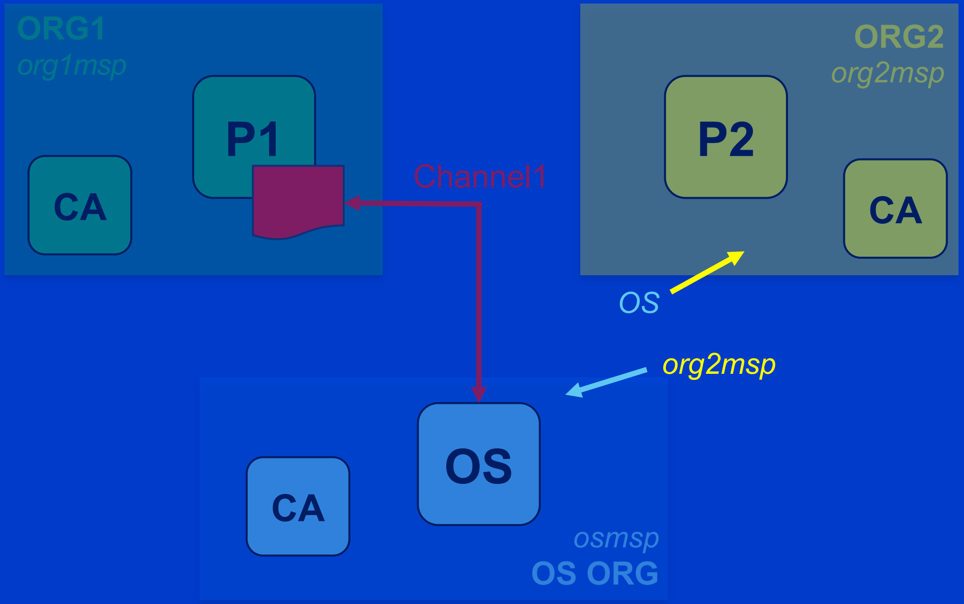
1. **Joiner Admin** creates a peer organization and peer
    1. Create ORG2 CA
    2. Associate the CA admin identity
    3. Use the CA to register ORG2 identities
    4. Create the ORG2 MSP definition
    5. Create the peer



Joining a network [2/4]

Detailed tutorial at IBM Blockchain Platform Console -> Get Started -> Join a Network

1. **Joiner Admin** creates a peer organization and peer
2. **OS Admin** adds ORG2 to the existing ordering service *
   1. **Joiner Admin** exports the organization information and sends to the OS admin
   2. **OS Admin** imports the ORG2 definition into the ordering service and add its peer's org to the OS
   3. **OS Admin** exports the OS definition
   4. **Joiner Admin** imports the OS definition



* If the **OS Admin** and **Joiner Admin** are running in the same console, the sub-steps are not required. The console will let you simply add ORG2 directly to the Ordering Service.

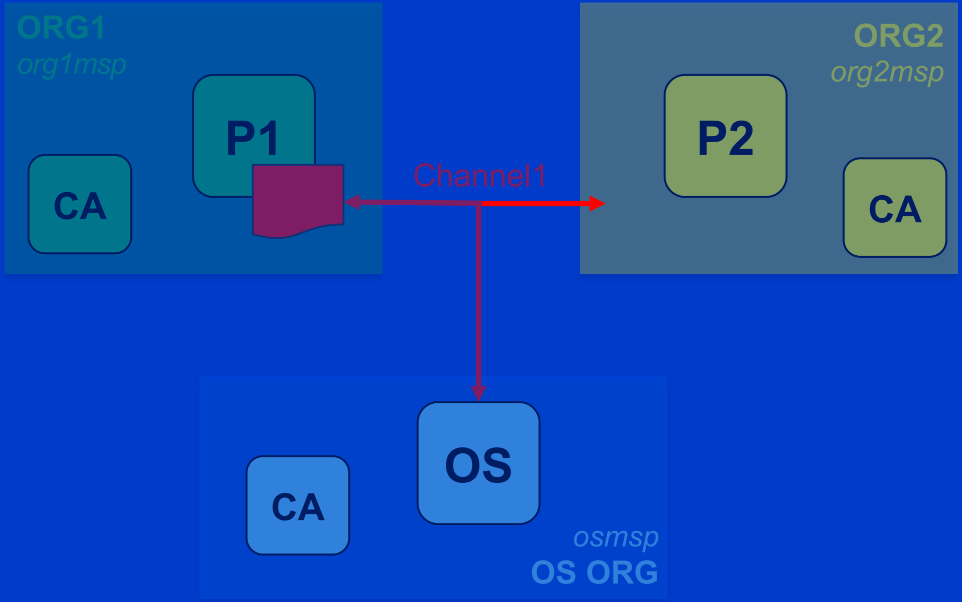
Joining a network [3/4]

Detailed tutorial at IBM Blockchain Platform Console -> Get Started -> Join a Network

1. **Joiner Admin** creates a peer organization and peer

2. **OS Admin** adds ORG2 to the existing ordering service

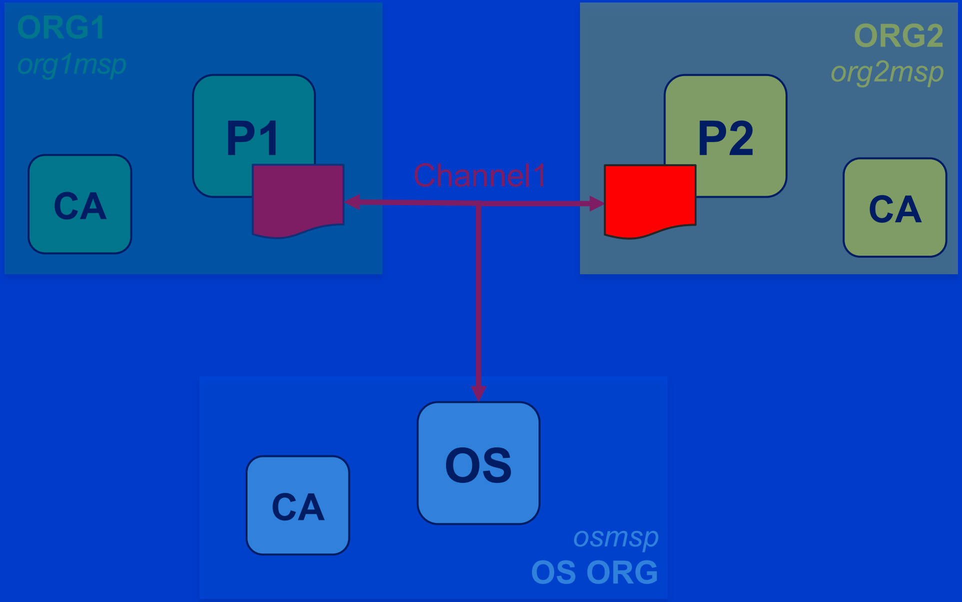
3. A channel admin (e.g. **Builder Admin**) must add the peer's organization to the existing channel



Joining a network [4/4]

Detailed tutorial at IBM Blockchain Platform Console -> Get Started -> Join a Network

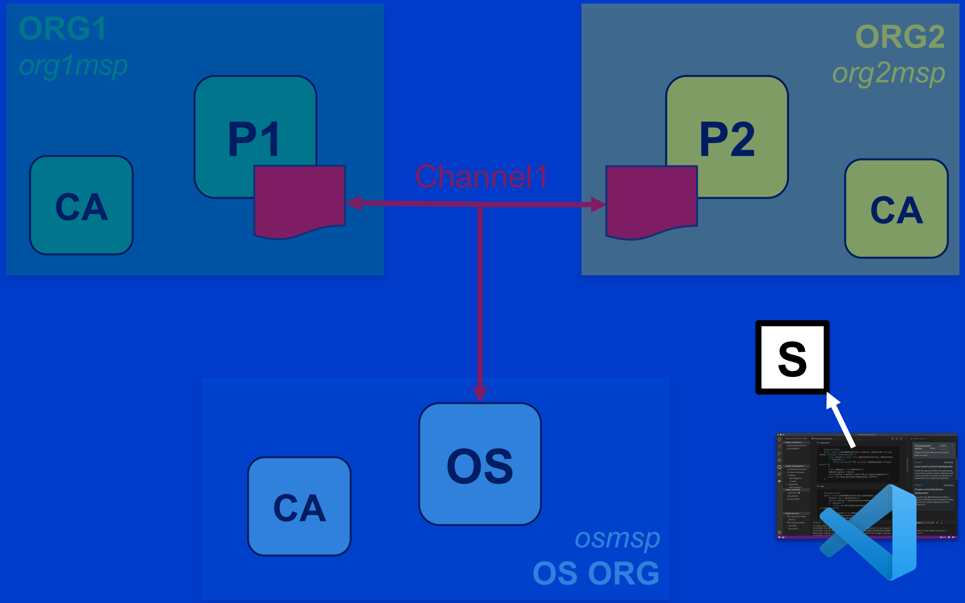
1. **Joiner Admin** creates a peer organization and peer
2. **OS Admin** adds ORG2 to the existing ordering service
3. A channel admin (e.g. **Builder Admin**) must add the peer's organization to the existing channel
4. **Joiner Admin** joins the peer to the channel
   …and/or create more channels as required



Deploying a smart contract [1/3]

Detailed tutorial at IBM Blockchain Platform Console -> Get Started -> Deploy Smart Contracts

1. **Developer** writes a smart contract in VSCode and packages as a *.cds* file

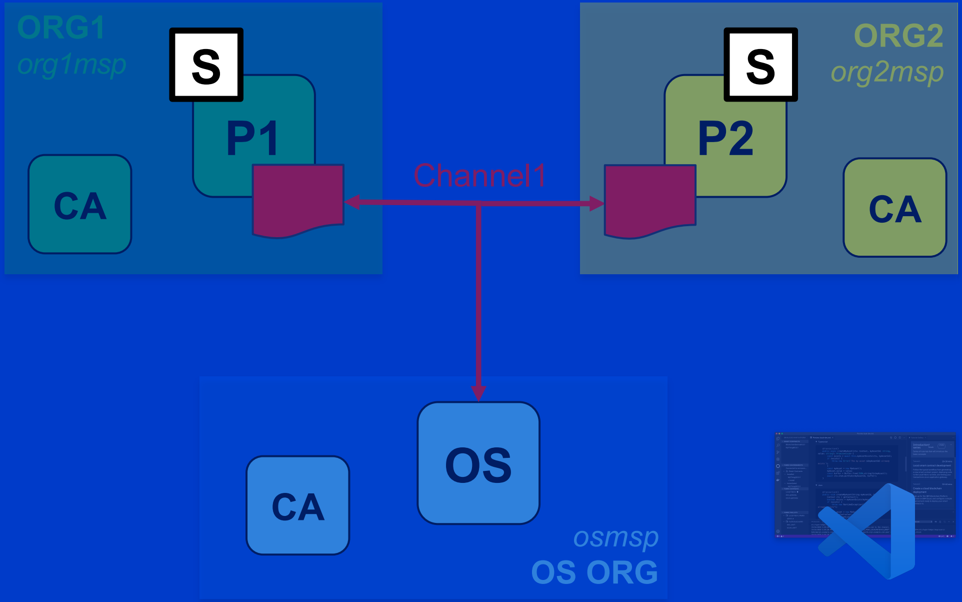


Deploying a smart contract [2/3]

Detailed tutorial at IBM Blockchain Platform Console -> Get Started -> Deploy Smart Contracts

1. **Developer** writes a smart contract in VSCode and packages as a *.cds* file

2. **Admins** install the .cds file on each endorsing peer



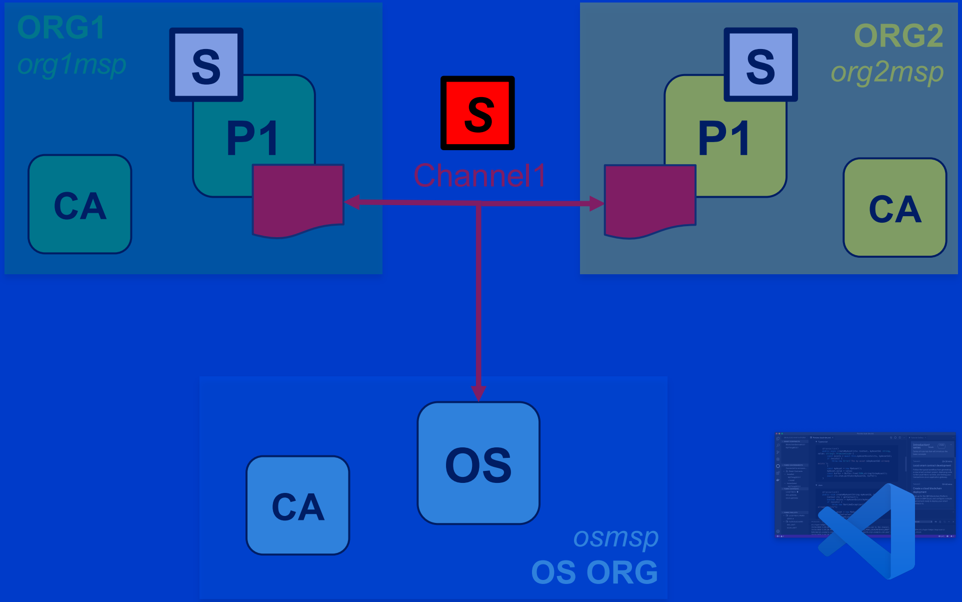
Deploying a smart contract [3/3]

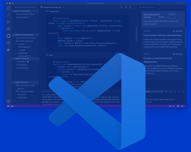
Detailed tutorial at IBM Blockchain Platform Console -> Get Started -> Deploy Smart Contracts

1. **Developer** writes a smart contract in VSCode and packages as a *.cds* file

2. **Admins** install the .cds file on each endorsing peer

3. A channel operator (e.g. **Builder Admin** or **Joiner Admin**) instantiates the smart contract once per channel.

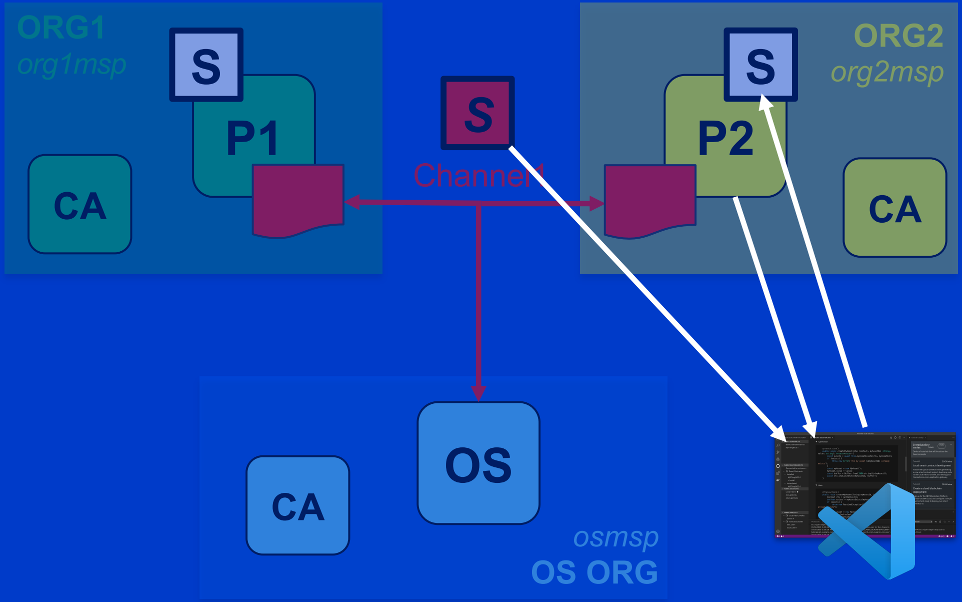




Testing a smart contract

Detailed tutorial at IBM Blockchain Platform Console -> Get Started -> Deploy Smart Contracts

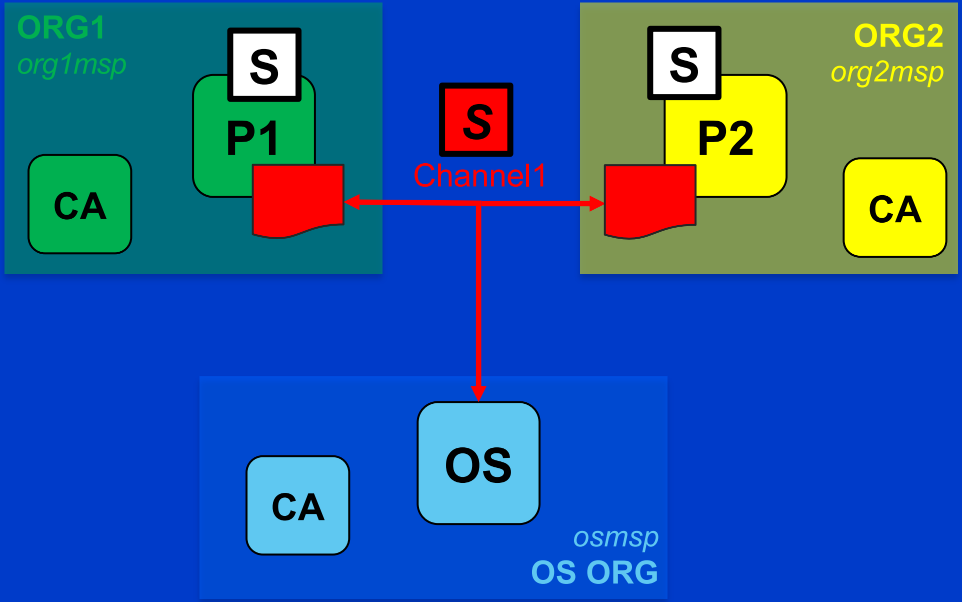
1. Download connection profile using "Connect with SDK" option against chaincode in console
2. If endorsement from multiple organizations is needed, nominate anchor peers on channel to allow discovery
3. Use connection profile to add gateway in VSCode
4. In VSCode add wallet and create identity from the CA enrollment ID
5. Connect to gateway
6. Discover channels and submit / evaluate transactions



Success!

- Now that you've seen how to create a **basic network**, you should also be able to see how more advanced environments can evolve:
  - Multiple channels, peers etc.
  - Import and manage components running on-premises or on other clouds
  - Achieve high availability
  - Govern changes to the network (e.g. onboarding and offboarding)
  - Deploy additional smart contracts, endorsement policies etc.

- Often, you just need to rerun the relevant **join** or **deploy** steps to get the configuration you need

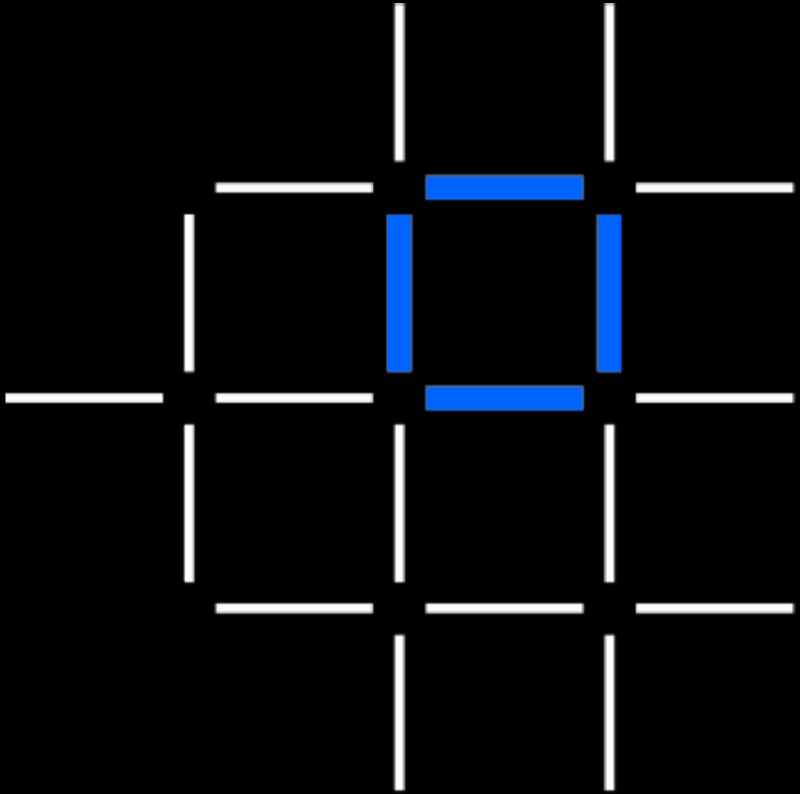


Thank you

Barry Silliman
Blockchain Enablement on IBM Z and LinuxONE
IBM North America Technical Sales
silliman@us.ibm.com

Questions? Tweet us or
go to ibm.com/blockchain

@IBMBlockchain
IBM Blockchain
IBM Blockchain



IBM **Blockchain**