← **A2: Creating a smart contract**

Tutorial A3: Deploying a smart contract

Estimated time: `10 minutes`

In the last tutorial we used the IBM Blockchain Platform to create a basic TypeScript smart contract. In this tutorial we will:

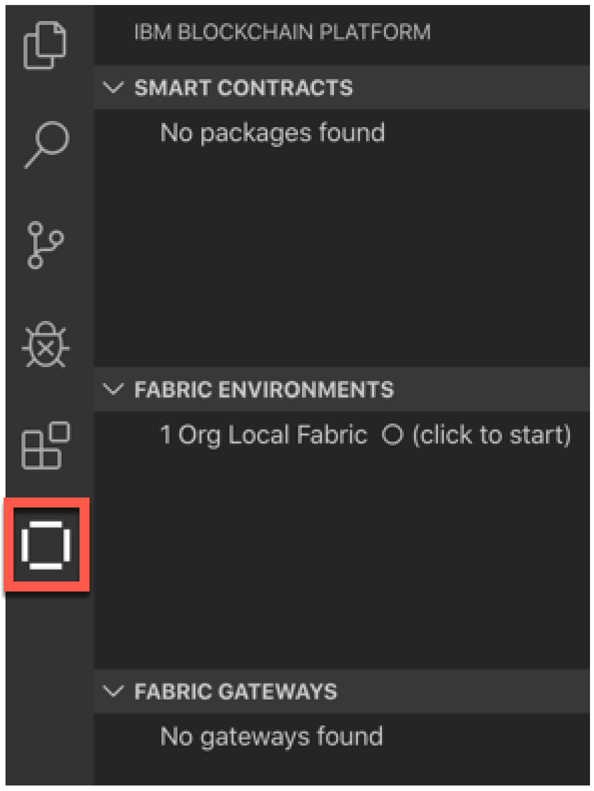
- Start an instance of Hyperledger Fabric in the local workspace
- Package the smart contract we previously created
- Deploy the smart contract to the running Hyperledger Fabric

In order to successfully complete this tutorial, you must have first completed tutorial A2: Creating a smart contract in the active VS Code workspace.

`A3.1`: Expand the first section below to get started.

▶ Start the VS Code Hyperledger Fabric environment

`A3.2`: Click on the IBM Blockchain Platform icon in the activity bar to show the sidebar.



IBM BLOCKCHAIN PLATFORM

SMART CONTRACTS

No packages found

FABRIC ENVIRONMENTS

1 Org Local Fabric ○ (click to start)

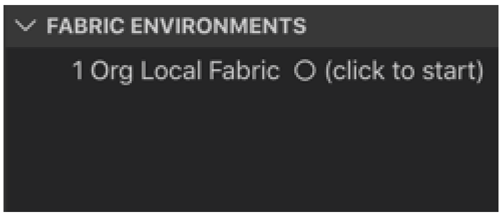
FABRIC GATEWAYS

No gateways found

The Fabric Environments view

The IBM Blockchain Platform VS Code Extension allows you to connect to Hyperledger Fabric networks and use them to test smart contracts.

The connected networks are shown in the Fabric Environments view.


FABRIC ENVIRONMENTS
1 Org Local Fabric ○ (click to start)

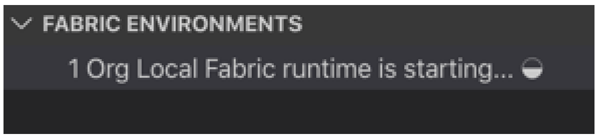
The same view also allows you to configure instances of Hyperledger Fabric that run entirely within the local VS Code environment. The required Hyperledger Fabric components are automatically downloaded and started by the tool itself.

A local Hyperledger Fabric environment is a convenient way of unit testing smart contracts. This environment cannot exercise the distributed connectivity that differentiates a blockchain, but it's useful in understanding how Hyperledger Fabric will behave in real networks.

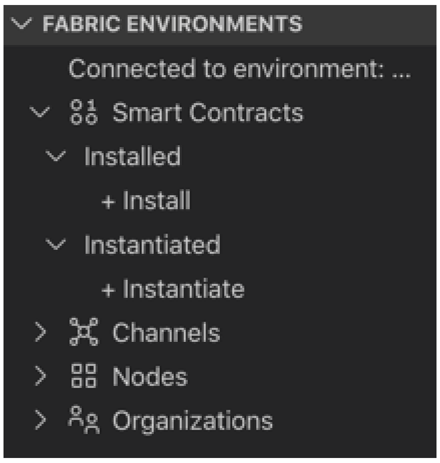
We will use a simple, one organization, local Hyperledger Fabric environment throughout this set of tutorials. We will connect to remote Hyperledger Fabric networks in later tutorial sets.

☐ `A3.3`: In the Fabric Environments view, click "*1 Org Local Fabric O (click to start)*"

This will download and start the embedded instance of Hyperledger Fabric, and may take up to five minutes to complete.


FABRIC ENVIRONMENTS
1 Org Local Fabric runtime is starting...

When Hyperledger Fabric has fully initialized, the view will change to show the smart contracts, channels, nodes and organizations in the local environment.

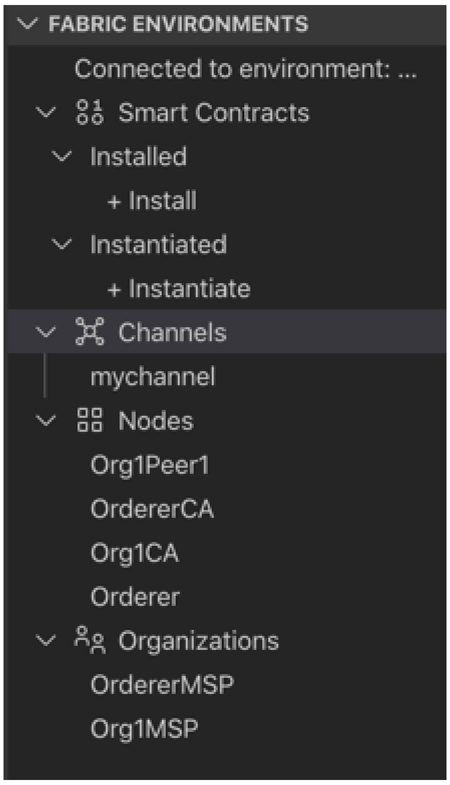

FABRIC ENVIRONMENTS
Connected to environment: ...
Smart Contracts
Installed
+ Install
Instantiated
+ Instantiate
Channels
Nodes
Organizations

Each of these elements tell you what's configured in the connected environment:

- **Smart contracts** show the smart contracts that are available. They can be *installed*, which means that the code has been copied to the peers, and *instantiated*, which means that they are running.
- **Channels** define the scope of each network, and form one method of choosing how organizations share data. We will look at channels in a later tutorial.
- **Nodes** are the Hyperledger Fabric components that make the system work. There are three types of nodes: peers (which run ledgers and execute smart contracts), orderers (which assert transaction order and distribute blocks to peers), and certificate authorities (which provide the means of identifying users and organizations on the network).
- **Organizations** are the members of the blockchain network. Each organization can consist of many users.

If you expand the various sections you'll the various defaults for each of these elements:

- By default there are no **smart contracts** installed or instantiated.
- There is a single default **channel** called *mychannel*.
- There are four **nodes**: a single peer called *Org1Peer1*, an ordering node called *Orderer* and a certificate authories for each of the two default organizations.
- Two **organizations**: one for the peer (*Org1MSP*) and one for the orderer (*OrdererMSP*). It is good practice to use separate organizations for the orderer nodes and peers.



Starting again?
If you ever need to start with a new Hyperledger Fabric instance, hover over the Fabric Environments view, click the ellipsis ('...') and select 'Teardown Fabric Environment'. Use with caution: this will completely wipe the Hyperledger Fabric instance and anything deployed to it. Development files in your workspace (e.g. smart contract projects) will remain.

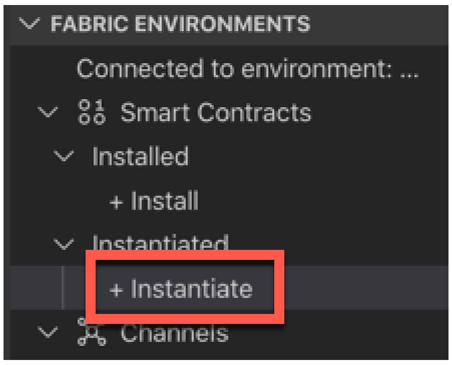
☐ `A3.4`: Expand the next section of the tutorial to continue.

▶ **Package, install and instantiate the smart contract**

We will now package, install and instantiate our smart contract into the local environment. It is possible to do this as three separate actions in VS Code, or by just using the 'Instantiate' action, which will also do the package and install steps if necessary.

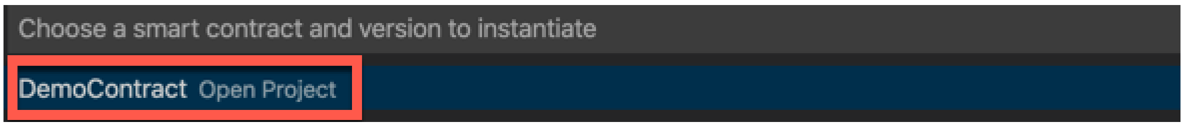
For simplicity, we will just use the single-step instantiate for now. When we explore the upgrade process in tutorial A6: Upgrading a smart contract, we will split this process up.

☐ A3.5: In the Fabric Environments view, click "Smart Contracts" -> "Instantiated" -> "+ Instantiate".



IBM Blockchain Platform detects the smart contract project created in the previous tutorial.

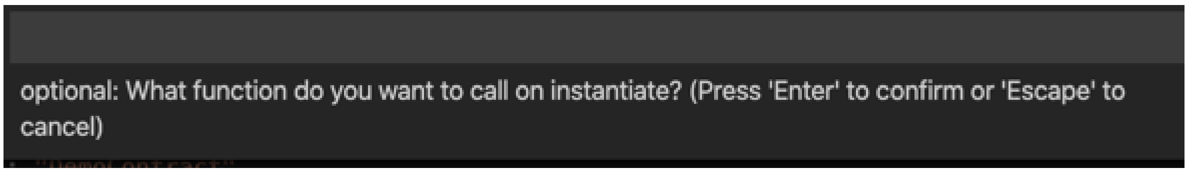
☐ A3.6: In the Command Palette, click 'DemoContract'.


Choose a smart contract and version to instantiate
DemoContract Open Project

It may take up to a minute to package and install the smart contract.

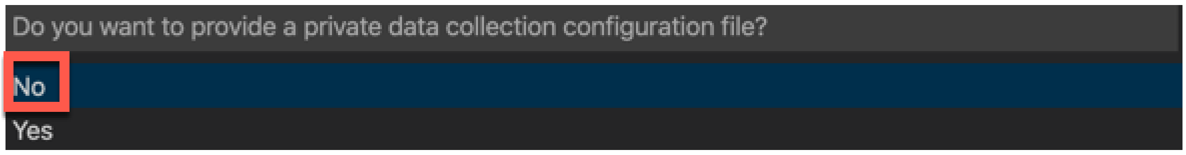
Before the smart contract is instantiated you will be prompted to enter the name of any required instantiation function.

☐ A3.7: In the Command Palette, press Enter when prompted for an instantiate function name.


optional: What function do you want to call on instantiate? (Press 'Enter' to confirm or 'Escape' to cancel)

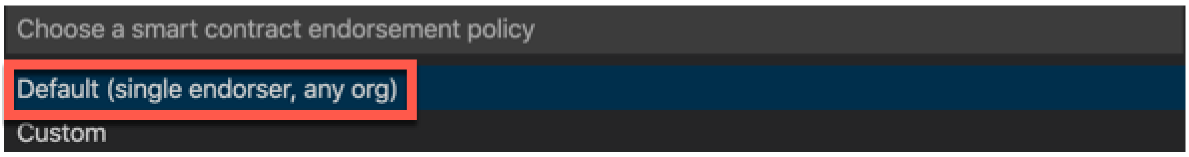
Private data is an advanced technique for sharing data between organizations. We will not be using that feature for now; it is the subject of a later tutorial.

☐ A3.8: In the Command Palette, click 'No' to not provide a private data collection configuration file.


Do you want to provide a private data collection configuration file?
No
Yes

The endorsement policy determines which peers get to run the smart contract. As we only have one peer in our organization, we can accept the default.

A3.9: In the Command Palette, click 'Default' to accept the default smart contract endorsement policy.

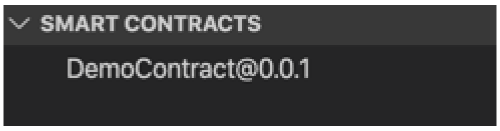

Choose a smart contract endorsement policy

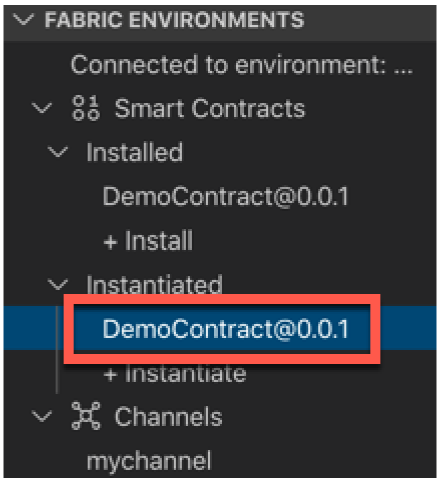
Default (single endorser, any org)

Custom

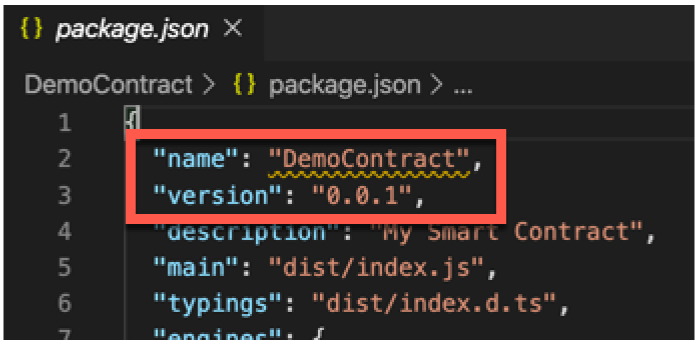
Instantiation may take a further few minutes to complete.

When instantiation has completed you will see the new smart contract package listed in the Smart Contracts view, and also underneath the Instantiated section of the connected environment.





The text after the '@' sign is the version; note that it is set to '0.0.1'. For TypeScript smart contracts, both the name and version are taken from the *package.json* file in the root of the smart contract project:



```
{
    "name": "DemoContract",
    "version": "0.0.1",
    "description": "My Smart Contract",
    "main": "dist/index.js",
    "typings": "dist/index.d.ts",
```

We will update this in a later tutorial.

Summary

In this tutorial we have started the embedded Hyperledger Fabric environment, packaged our smart contract, installed it and instantiated it.

In the next tutorial we will try running some transactions that use it, to see how they are handled in Hyperledger Fabric.

→ A4: Invoking a smart contract from VS Code