# CREDIT SUISSE BLOCKCHAIN HACKATHON
**January 2018**

## Hackathon Training Material

## Document Information

| | | | |
|---|---|---|---|
| **Project Name:** | Credit Suisse  Blockchain Hackathon Jan 2018 – HyperLedger Fabric & Corda Documentation | | |
| **Prepared By:** | Cognizant, Microsoft, R3 | **Approved Version No:** | V1.8 |
| **Approved By:** | | **Approved Date:** | Jan 5th 2018 |

### Document Version History

| Version Number | Version Date | Revised By | Description |
|---|---|---|---|
| | | | |
| | | | |
| | | | |
| | | | |

# Table of Contents

# 1 Introduction

The Credit Suisse (CS) Blockchain Hackathon delivers a 2-day, hands-on workshop where a Microsoft & Cognizant team leads CS associates through the development of a minimally viable product (MVP) to demonstrate the value of a distributed ledger solution. The expected outcome is a working example that the CS can take forward and update it according to their needs, however completeness of the MVP in the confines of the 2-day hack-fest is not guaranteed.

This document provides an overview of Blockchain in general, along with introduction to the two shortlisted DLT frameworks (Hyperledger Fabric and Corda). It also provides steps to install Hyperledger Fabric on your machine and building solutions leveraging Hyperledger Fabric 1.0. This is designed as a pre-read material for the CS associates participating in the hack-fest containing documents such as reading materials, links, installation documentation, sample codes etc. which makes them familiar with the steps required to develop the MVP leveraging their DLT framework of choice.

## 1.1 THEMES SHORTLISTED FOR THE HACKATHON

For the purpose of the Hackathon, The following themes have been shortlisted.

- NDA Tracker – to be able to agree on terms of Non-Disclosure Agreements and provide signatures on a DLT platform
- Develop a ETF Creation / Redemption Platform
- IT Allocations / Budgeting - Create a distributed ledger solution to streamline the entire IT allocations process of distributing budgets throughout various departments at the bank

## 1.2 GUIDELINES FOR PARTICIPANTS

- Bring your own device (PC or Mac compatible)
- Read all material and information on ledgers prior to hackathon. Ask questions, give feedback, document ideas
- Have fun!

## 1.3 INFRASTRUCTURE FOR THE HACKATHON

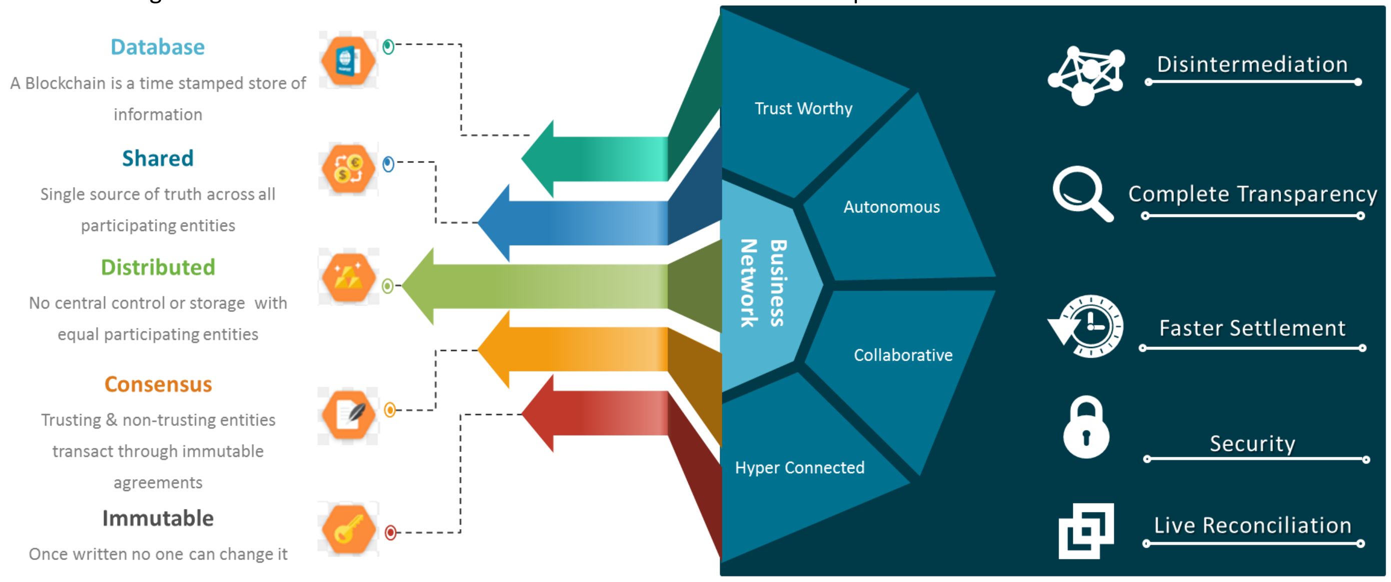
The development of the MVP would be done on Laptops/Desktops that participants currently own. Once the development of the MVP is done, the same can be deployed on Azure Infrastructure provided by Microsoft for integration testing and demo/presentation. One Azure VM would be provided for each team. The participants need to install the prerequisite software on their laptop themselves.

## 2 Blockchain/DLT Overview

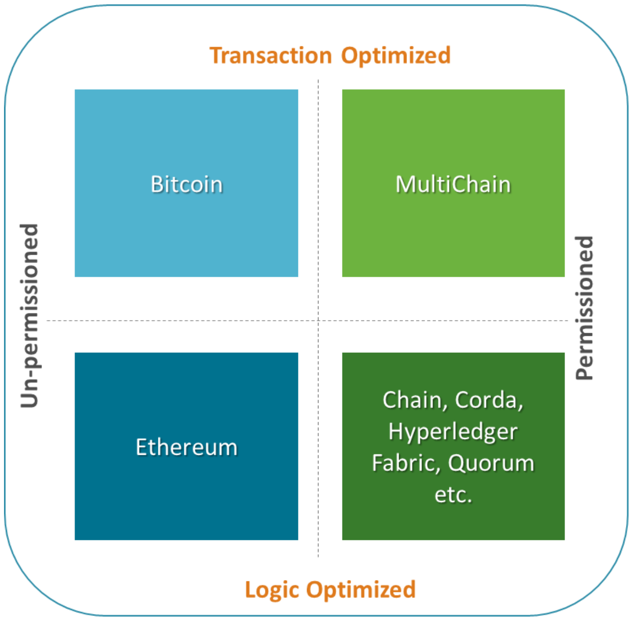
The links below provide an overview of Blockchain/DLT concepts and the need for Private Permissioned Blockchain/DLTs for Enterprise.

1. What is Blockchain?
2. Introduction to Blockchain Technology
3. How Blockchain Works
4. Why is Blockchain relevant for Business
5. Fundamental Components for Blockchain for Business
6. Problem Area in today's business networks And Blockchain

The below diagram summarizes what a DLT is and what are the benefits that it provides


Database
A Blockchain is a time stamped store of information
Shared
Single source of truth across all participating entities
Distributed
No central control or storage with equal participating entities
Consensus
Trusting & non-trusting entities transact through immutable agreements
Immutable
Once written no one can change it
Business Network
Trust Worthy
Autonomous
Collaborative
Hyper Connected
Disintermediation
Complete Transparency
Faster Settlement
Security
Live Reconciliation

There are various Blockchain/DLTs frameworks that are present today. They can be generally categorized into the following 4 Quadrants



### Permissioned Vs Un-Permissioned

- **In a permissioned Blockchain**, the Identity of the participants are known, such that entitlements can be assigned to them e.g. who can transact, who can validate etc. As such there is some degree of governance of the network. Private Blockchains e.g. MultiChain, Corda, HyperLedger Fabric are good examples of permissioned Blockchains/DLTs.
- **In an un-permissioned Blockchain**, there are no special privileges that are assigned to participants. As such the need for governance of the network is relatively low. Public blockchain frameworks e.g. Bitcoin/Ethereum are good examples of un-permissioned Blockchains.

### Transaction Optimized Vs Logic Optimized

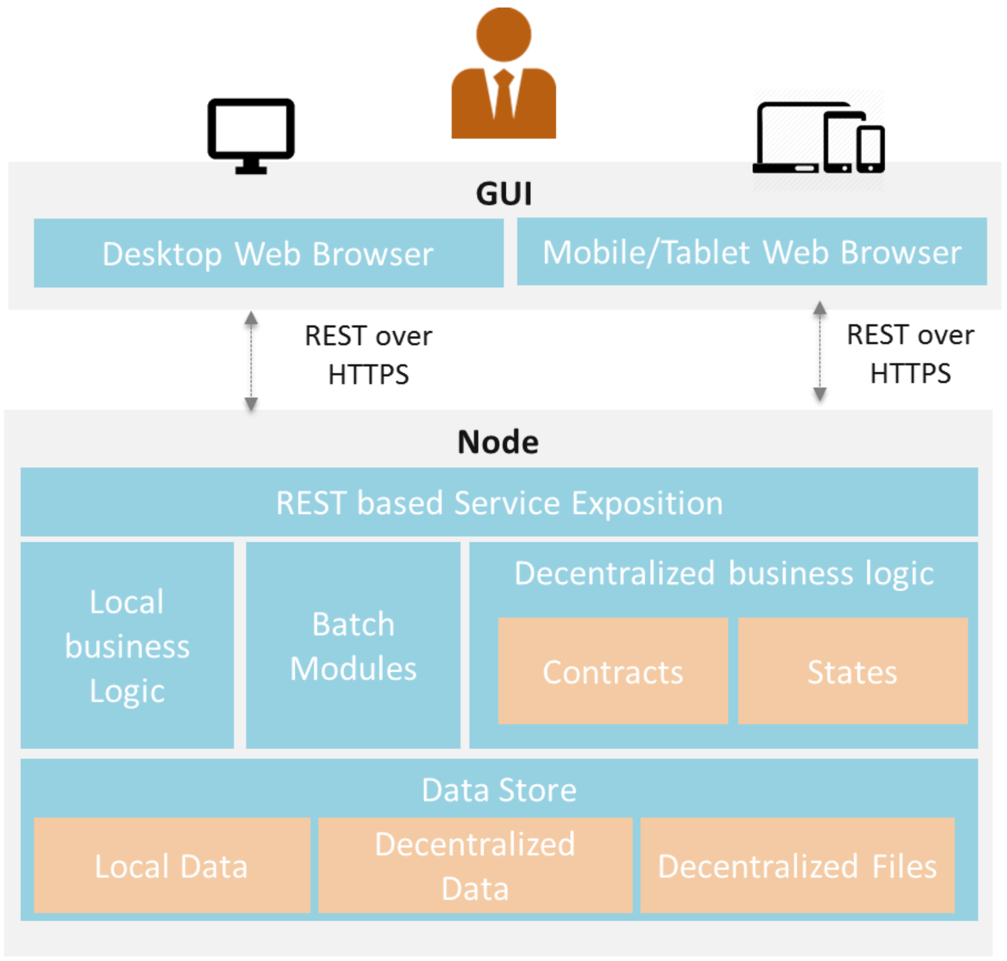
- **In a Transaction Optimized Blockchain**, the business logic to be executed on the Blockchain is already coded as part of the framework, with limited configurable features. These are typically API based and are good for certain types of use cases, e.g. Coin/asset issuance and transfer. Bitcoin/Multichain are good examples of this type of Blockchain/DLT
- **In a Logic optimized Blockchain**, provides developers to write their own business logic and execute them on the DLT. Thus the use cases where they can be used is a lot broader. The Business logic written is called Smart Contracts. Ethereum, Corda, HyperLedger Fabric are good examples of this type of Blockchain/DLT

# 3 Important considerations when developing DLT solutions

Following are some key decision points/Considerations that should be kept in mind, when designing solutions leveraging a DLT framework.

## 3.1 LOGICAL ARCHITECTURE

Like any other solution development creating a logical architecture for the solution where business application is abstracted from the platform choices is key. Following a layered architecture enables loose coupling between the different components and thus ensures lower cost of change. The following diagram elaborates a typical logical architecture that should be followed for developing a DLT based solution.



***DLT Based Solution Logical Architecture***

As per the links in section 2, Blockchain is a decentralized technology, where each node is managed by a participant and they communicate with each other over DLT. As such when developing the use cases, participants need to ensure that the necessary components displayed above are built out for each participant and deployed accordingly (e.g. UI components for each counterparty interfacing with Nodes running on different Port Numbers etc.)

## 3.2 OFF-CHAIN VS ON-CHAIN PROCESSING

A DLT solution cannot be developed in a silo. It will need to interact with other systems/components. As such decision of which components are on-chain and which components are off-chain is also important. Some of the reasons to develop off-chain components are

- The data/logic is private to the enterprise and doesn’t need agreement from counterparty
- Blockchain/DLT is not meant for that kind of processing e.g. file handling etc.

Please ensure that the design of the system incorporates, which system components are off-chain/on-chain along with justifications.

## 3.3 INTEGRATION TOUCHPOINTS

As mentioned above, a DLT based solution cannot exist in a silo. It needs to integrate with other enterprise solutions for end to end functionality to work. As such the exposition of integration touchpoints also need to be considered. Most frameworks provide examples of how to create REST based interfaces on top of the Blockchain/DLT nodes (e.g. using Java for Corda or Node.JS based JavaScript for HyperLedger Fabric). Providing the right abstraction through REST interface is also an important aspect of the solution development.

## 3.4 TECHNOLOGY ARCHITECTURE

A DLT framework is one of the technologies to be used in the overall solution, for the end to end solution to work, decision needs to be arrived at to ensure that the participants map each of the logical architecture components with the corresponding technology choice e.g. Angular.JS for UI, Java for REST service(corda), MySQL for Local data storage. Please ensure that you identify the software components that you would be using upfront. VMs will be made available before the hack for the preparation in advance.

# 4 Develop solution using Hyperledger Fabric

Please use this section if you leveraging Hyperledger Fabric.

## 4.1 INTRODUCTION TO HYPERLEDGER FABRIC

This section provides links to information that provides an overview of the HyperLedger Fabric Blockchain Framework. This would provide a good understanding of the basic constructs of HyperLedger fabric as well as provide access to sample codes/hands on exercises. Participants are encouraged to read the links below and also try the examples to become familiar with HyperLedger Fabric.
Overview - Introduction to Hyperledger and Permissioned Blockchain

**Must Read**: Please thoroughly go through the Hyperledger Fabric documentation : At minimum cover following (links provided below)

Hyperledger Fabric Model : This section summarizes the key design components built into Hyperledger Fabric
Hyperledger Architecture : Please go through this section to have a good understanding of following topics:
Roles within Hyperledger Fabric Network: Clients, Peers, Ordering Service
Transaction Flow and Consensus: In Hyperledger Fabric consensus is made up of three distinct steps: Transaction endorsement, ordering, validation and commitment.
State Database and Ledger. *Additionally review the following link*
Endorsement Policies. Additionally review the following link
Channels
Channels allow organizations to utilize the same network, while maintaining separation between multiple blockchains. Only the members of the channel on which the transaction was performed can see the specifics of the transaction.
Chaincode
Hyperledger Fabric smart contracts are called chaincode and are written in Go. The chaincode serves as the business logic for a Hyperledger Fabric network, in that the chaincode directs how you manipulate assets within the network.
Membership Service Provider (MSP)
The membership service provider, or MSP, is a component that defines the rules in which identities are validated, authenticated, and allowed access to a network.
Node SDK
Applications use APIs to run smart contracts/Chaincode. These contracts are hosted on the network, and identified by name and version. APIs are accessible with a software development kit, or SDK. Currently, Hyperledger Fabric has three options for developers: Node.js SDK, Java SDK, and CLI. For the purposes of this Hackathon we will be using Node SDK.

## 4.2 CONFIGURING THE DEVELOPMENT MACHINE (UBUNTU VM ON WINDOWS)

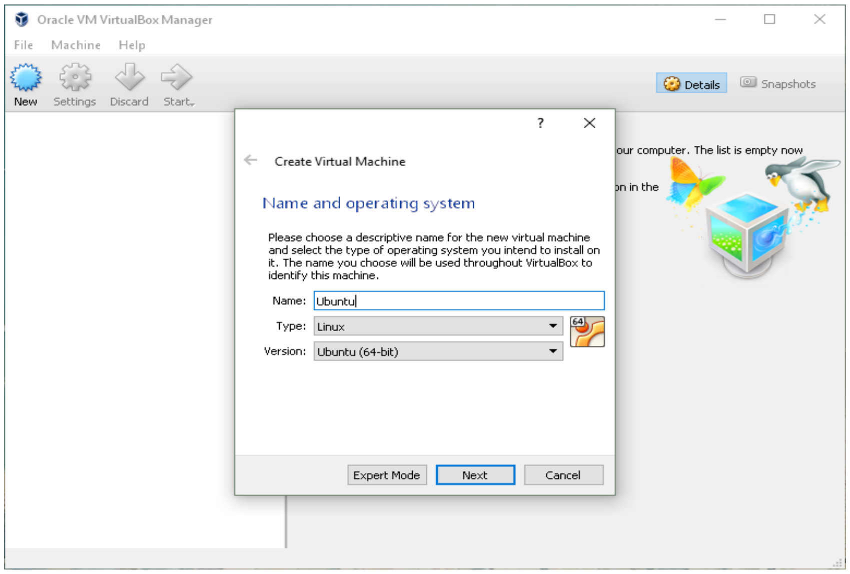
In order to develop using HyperLedger Fabric, the following steps need to be performed to configure your development machine. Please note: Depending upon use case there might be additional software/frameworks that might be needed e.g. IPFS for decentralized file storage, RDBMS like MySQL for local data storage, Angular.JS for UI development etc. The instructions in this section pertains only to setup of Hyperledger Fabric environment.

The instructions provided are used to prepare an Ubuntu Virtual Machine on Windows. Note this can be done using virtual machines in Azure, and the same instructions apply. Note that you may use non-Windows machines for development as well, e.g. OSX, but that process is not documented in this planning document; however, the tools identified are all supported in an OSX environment.

***Another alternative for development machine setup is covered in section 4.3 which is a native docker deployment on Windows. That is only applicable for Machines with Windows 10***

If these tools are already installed, it is strongly recommended you reinstall to make sure you are at the right versions of the products.
***Note : Please note you must not have Docker for Windows installed, or HyperV enabled for VirtualBox to run.***
***See here(http://www.poweronplatforms.com/enable-disable-hyper-v-windows-10-8/) on how to disable HyperV***.

1. Install Ubuntu Virtual Machine in Oracle Virtualbox
    a. Download and install latest Oracle Virtualbox from https://www.virtualbox.org/wiki/Downloads .
    b. You will be greeted with below screen


Oracle VM VirtualBox Manager
File Machine Help
New Settings Discard Start
Details Snapshots
our computer. The list is empty now
on in the
Create Virtual Machine
Name and operating system
Expert Mode
Next
Cancel

Please choose a descriptive name for the new virtual machine and select the type of operating system you intend to install on it. The name you choose will be used throughout VirtualBox to identify this machine.

Name: Ubuntu

Type: Linux

Version: Ubuntu (64-bit)

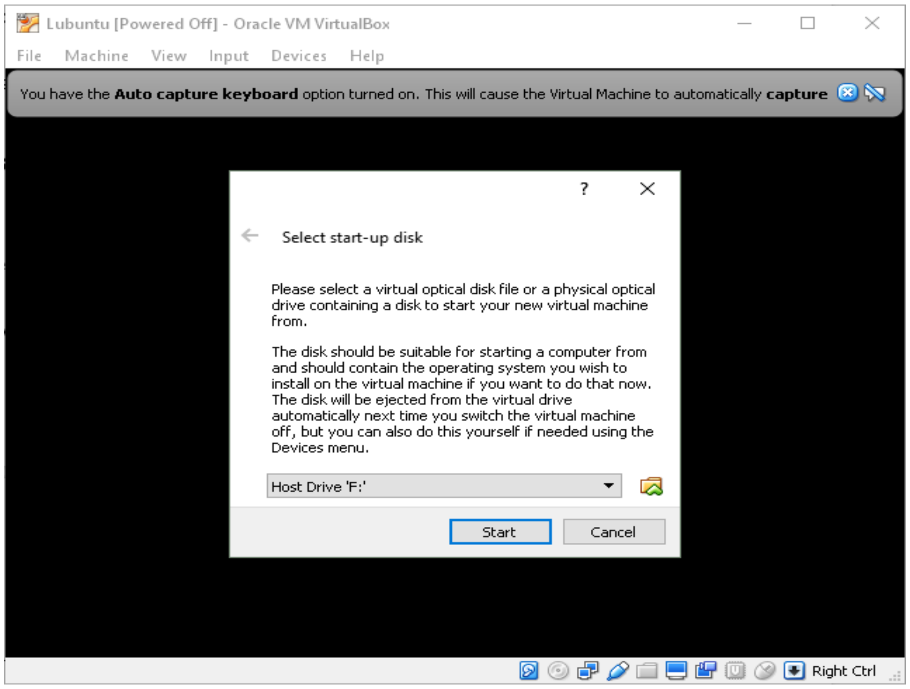
c. Click "New" to create a new virtual machine. At this point, you will need to choose a Linux distribution to install. Many Linux distributions will work but we recommend using Ubuntu 16.04, which is a very popular one

d. If you are sticking with our recommendation of Ubuntu, you should select Ubuntu (64-bit) (32-bit will have challenges as Docker installation will need 64 bit machine).

e. For the rest of the setup, use the following options:

    i. 2 GB system memory
    ii. 25 GB of free hard drive space

f. Before starting the virtual machine, you will need to install your Linux distribution.If using Ubuntu, please download the latest 16.04 desktop version here. Once the image file download is complete, follow steps below to boot your virtual machine.


Lubuntu [Powered Off] - Oracle VM VirtualBox
File Machine View Input Devices Help
Select start-up disk
Start
Cancel
Right Ctrl

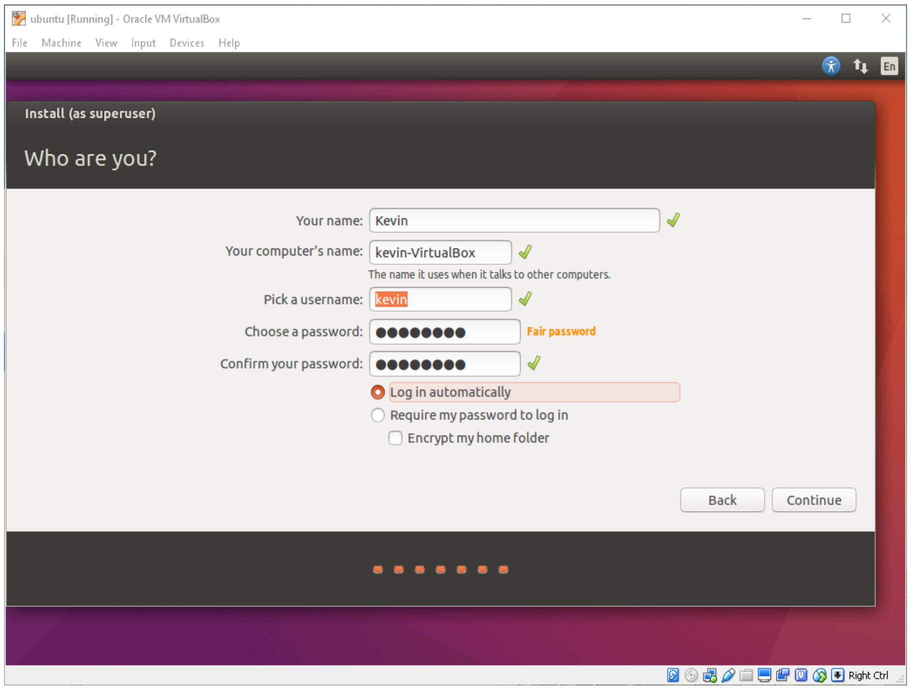
You have the **Auto capture keyboard** option turned on. This will cause the Virtual Machine to automatically **capture**

Please select a virtual optical disk file or a physical optical drive containing a disk to start your new virtual machine from.

The disk should be suitable for starting a computer from and should contain the operating system you wish to install on the virtual machine if you want to do that now. The disk will be ejected from the virtual drive automatically next time you switch the virtual machine off, but you can also do this yourself if needed using the Devices menu.

Host Drive 'F:'

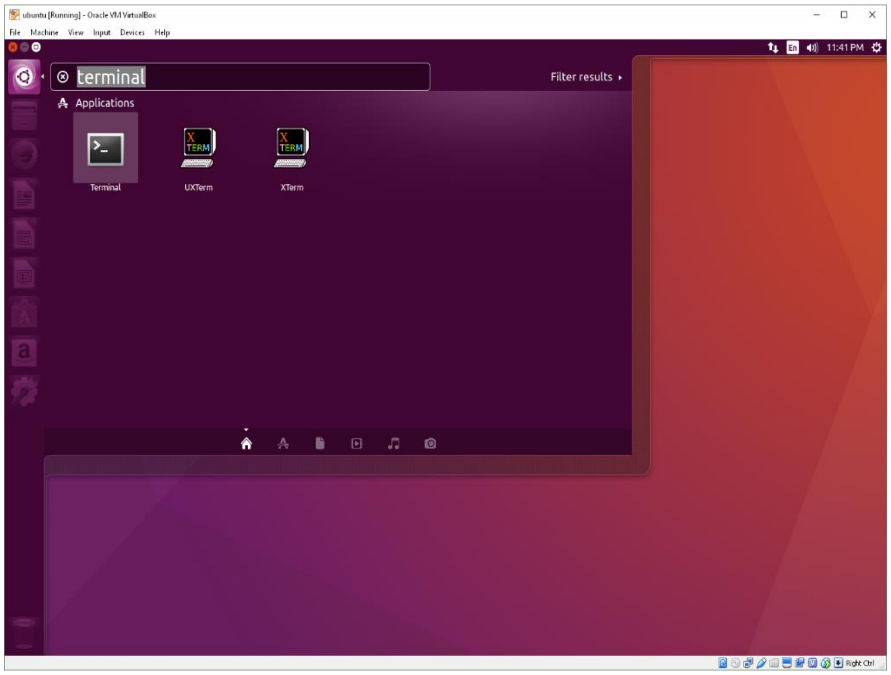
g. Browse for and open the image file you just downloaded, and click "Start". Follow the install prompts to install Linux. Restart when prompted


ubuntu [Running] - Oracle VM VirtualBox
File Machine View Input Devices Help
Install (as superuser)
Back
Continue
Right Ctrl

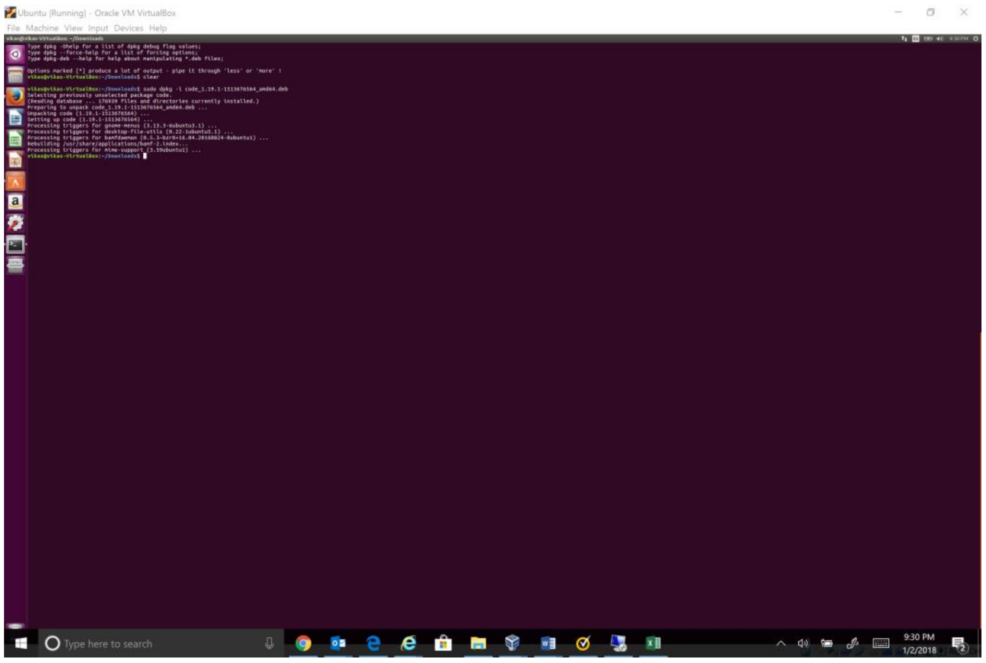
Who are you?

Your name: Kevin
Your computer's name: kevin-VirtualBox
The name it uses when it talks to other computers.
Pick a username: kevin
Choose a password: ●●●●●●●● Fair password
Confirm your password: ●●●●●●●●
[✓ Log in automatically]
Require my password to log in
Encrypt my home folder

h. To **open a terminal**, you can press **CTRL+ALT+T**, or find it by clicking the Ubuntu Home button and searching for 'terminal'.


ubuntu [Running] - Oracle VM VirtualBox
File Machine View Input Devices Help
11:41 PM
terminal
Filter results
Applications
Terminal
UXTerm
XTerm
Right Ctrl

2. Next install cURL using “sudo apt install curl”. To check run “curl -V”
3. Next install docker using Docker CE for Ubuntu
4. To Install Docker Compose
    a. sudo apt update
    b. Install docker-compose by following the steps in this Link
    c. Check to make sure that you have Docker version 17.03.1-ce or greater, and Docker Compose version 1.8.0 or greater using $ docker --version && docker-compose --version
5. Download .deb file from Visual Studio Code  into Downloads directory.
    a. cd /Downloads folder and then run the command “sudo dpkg -i code_1.19.1-1513676564_amd64.deb”.  Before you run the command check the file name in the Downloads folder



```
Type dpkg -Dhelp for a list of dpkg debug flag values;
Type dpkg --force-help for a list of forcing options;
Type dpkg-deb --help for help about manipulating *.deb files;

Options marked [*] produce a lot of output - pipe it through 'less' or 'more' !
vikas@vikas-VirtualBox:~/Downloads$ clear

vikas@vikas-VirtualBox:~/Downloads$ sudo dpkg -i code_1.19.1-1513676564_amd64.deb
Selecting previously unselected package code.
(Reading database ... 176939 files and directories currently installed.)
Preparing to unpack code_1.19.1-1513676564_amd64.deb ...
Unpacking code (1.19.1-1513676564) ...
Setting up code (1.19.1-1513676564) ...
Processing triggers for gnome-menus (3.13.3-6ubuntu3.1) ...
Processing triggers for desktop-file-utils (0.22-1ubuntu5.1) ...
Processing triggers for bamfdaemon (0.5.3~bzr0+16.04.20160824-0ubuntu1) ...
Rebuilding /usr/share/applications/bamf-2.index...
Processing triggers for mime-support (3.59ubuntu1) ...
vikas@vikas-VirtualBox:~/Downloads$ 
```

Ubuntu [Running] - Oracle VM VirtualBox
File Machine View Input Devices Help
vikas@vikas-VirtualBox: ~/Downloads
Type here to search
9:30 PM
1/2/2018

   b. Find the Vscode IDE by clicking the Ubuntu Home button and searching for 'code '. Once inside it do ctrl+shift+x to open available extensions and select Go extension to install
   c. Please note you can use any other IDE as long as it supports Go
6. Hyperledger Fabric uses the Go programming language 1.9.x for many of its components. Go Install using this link
7. Node.js Runtime and NPM
   a. curl -sL https://deb.nodesource.com/setup_6.x | sudo -E bash –
   b. sudo apt-get install nodejs
   c. sudo npm install npm@3.10.10 -g
8. The Fabric Node.js SDK requires an iteration of Python 2.7 in order for `npm install` operations to complete successfully. Retrieve the 2.7 version with the following command:
   a. sudo apt-get install python
   b. python --version
9. Next create a directory for Hyperledger Fabric under home folder
   a. /mkdir hlf
10. Go to above location and run
   a. run "sudi git clone https://github.com/hyperledger/fabric-samples.git"
   b. next run "cd fabric-samples"
11. Next run a script to download the images and binaries you will need:
   a. First add current user to the docker group "sudo usermod -aG docker vikas" ( replace vikas here with your username). Now logout and log in again
   b. cd hlf/fabric-samples
   c. now "sudo curl -ssL https://goo.gl/8EKzmZ | bash"
12. Following binaries and docker images are downloaded

| **Binary** | **Description** |
|---|---|
| cryptogen | A tool to generate x509 certificates used to identify and authenticate the components on the network |
| configtxgen | Generates configuration artifacts for ordering service and channel creation |

| | |
|---|---|
| configtxlator | A support tool to enable users to translate channel configuration  transaction into a readable form |

| **Image** | **Description** |
|---|---|
| peer | This image provides an environment for peer nodes |
| Orderer | This image provides an environment for orderer nodes |
| tools | Provides a container with cryptogen and configtxgen |

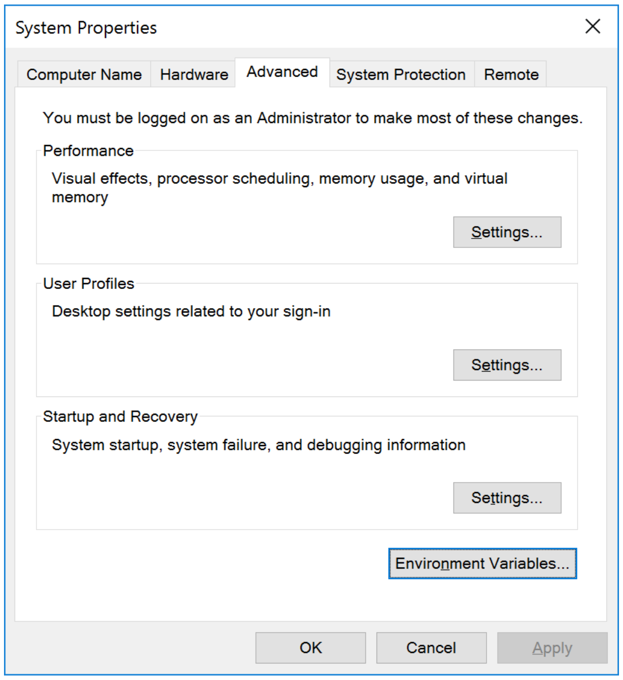
13. export the binary path:
    a. export PATH=<your HLF location>/fabric-samples/bin:$PATH

## 4.3 CONFIGURING THE DEVELOPMENT MACHINE NATIVELY ON WINDOWS (ALTERNATE SETUP TO 4.2)

**Machine Prerequisites:** Following are the minimum laptop or desktop configuration for this installation

- Docker CE– Windows 10 Professional Or Enterprize, 64 bit OS, RAM – 4 GB, HDD – standard size
- Docker Toolbox–Windows 7 or greater upto Windows 10 Home Edition, 64 bit OS, RAM – 4 GB, HDD – standard

1. Download the latest version of curl tool from link cURL. Download the following file
   "Win64 x86_64 7zip 7.53.1 binary SSL   Darren Owen"
    a. Extract the file to location C:\cURL using http://www.7-zip.org/ . After the extract you will see a certificate and curl.exe files under folder C:\cURL
    b. Right click on the certificate and click install. Select "local machine" and "Automatically Select the certificate store" options in the next steps. You should see a final message with "Import Successful"
    c. Next go to the Environment Variables under System Properties. On the System Variables, select PATH and click Edit
    d. On the next window, add a new Path variable "C:\cURL"



System Properties

Computer Name | Hardware | Advanced | System Protection | Remote

You must be logged on as an Administrator to make most of these changes.

Performance

Visual effects, processor scheduling, memory usage, and virtual memory

Settings...

User Profiles

Desktop settings related to your sign-in

Settings...

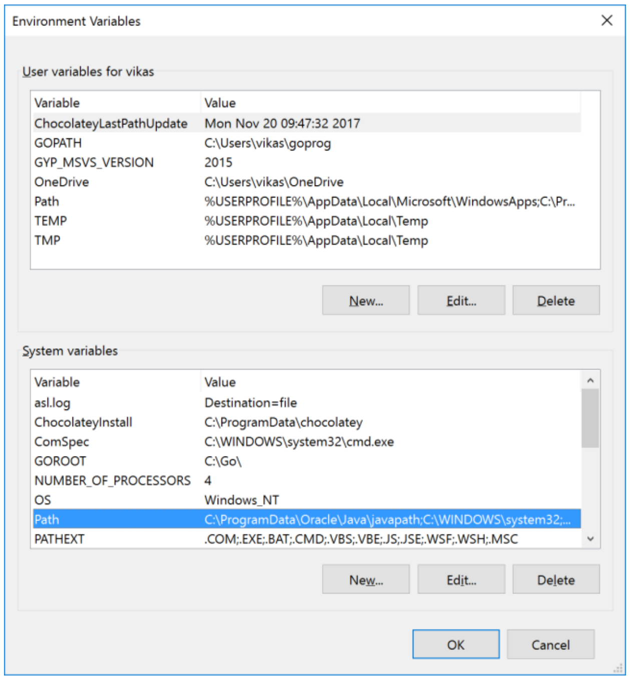
Startup and Recovery

System startup, system failure, and debugging information

Settings...

Environment Variables...

OK | Cancel | Apply



Environment Variables

User variables for vikas

| Variable | Value |
|---|---|
| ChocolateyLastPathUpdate | Mon Nov 20 09:47:32 2017 |
| GOPATH | C:\Users\vikas\goprog |
| GYP_MSVS_VERSION | 2015 |
| OneDrive | C:\Users\vikas\OneDrive |
| Path | %USERPROFILE%\AppData\Local\Microsoft\WindowsApps;C:\Pr... |
| TEMP | %USERPROFILE%\AppData\Local\Temp |
| TMP | %USERPROFILE%\AppData\Local\Temp |

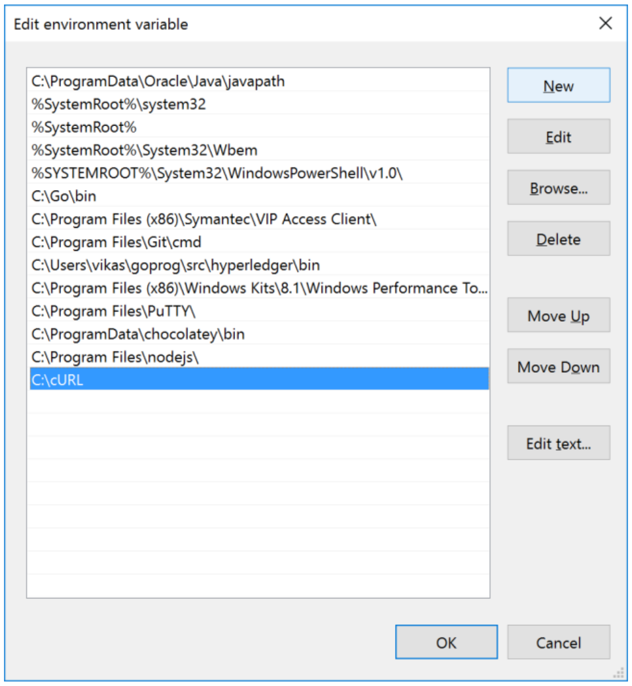
New... | Edit... | Delete

System variables

| Variable | Value |
|---|---|
| asl.log | Destination=file |
| ChocolateyInstall | C:\ProgramData\chocolatey |
| ComSpec | C:\WINDOWS\system32\cmd.exe |
| GOROOT | C:\Go\ |
| NUMBER_OF_PROCESSORS | 4 |
| OS | Windows_NT |
| Path | C:\ProgramData\Oracle\Java\javapath;C:\WINDOWS\system32;... |
| PATHEXT | .COM;.EXE;.BAT;.CMD;.VBS;.VBE;.JS;.JSE;.WSF;.WSH;.MSC |

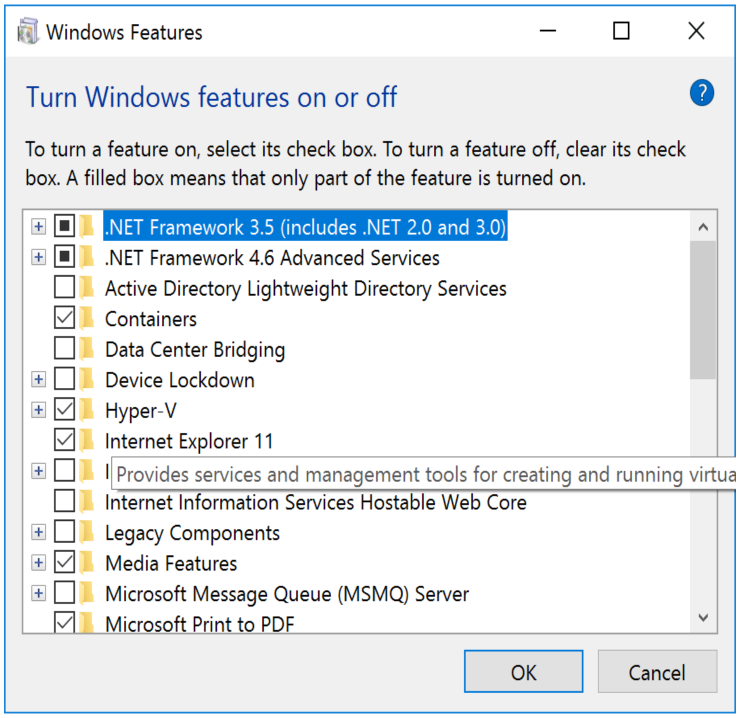
New... | Edit... | Delete

OK | Cancel


Edit environment variable

```
C:\ProgramData\Oracle\Java\javapath
%SystemRoot%\system32
%SystemRoot%
%SystemRoot%\System32\Wbem
%SYSTEMROOT%\System32\WindowsPowerShell\v1.0\
C:\Go\bin
C:\Program Files (x86)\Symantec\VIP Access Client\
C:\Program Files\Git\cmd
C:\Users\vikas\goprog\src\hyperledger\bin
C:\Program Files (x86)\Windows Kits\8.1\Windows Performance To...
C:\Program Files\PuTTY\
C:\ProgramData\chocolatey\bin
C:\Program Files\nodejs\
C:\cURL
```

2. Docker Install (Only for Windows 10)
   a. For Windows 10 machine please follow the link to install Docker CE (Stable). Docker version 17.03.0-ce or greater is required.
   b. Docker CE for Windows uses Windows-native Hyper-V virtualization. Hence the Hyper-V feature needs to be enabled. Search for "Turn Windows features On or Off". Then as shown in below screen print ensure Hyper-V feature is turned on


Windows Features

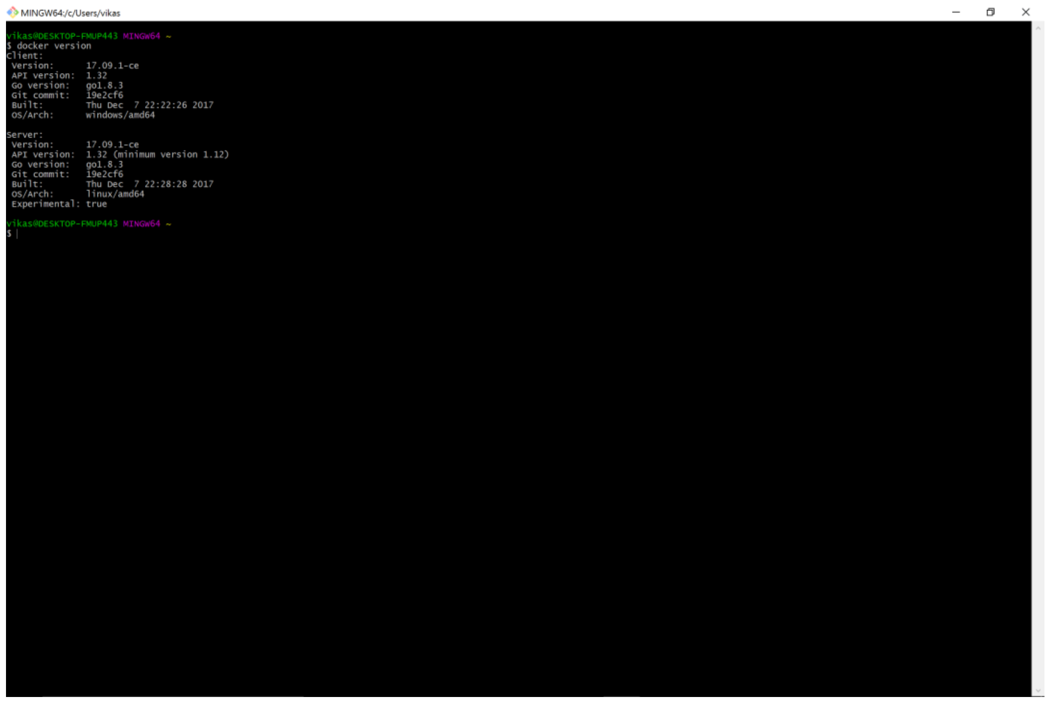
Turn Windows features on or off

To turn a feature on, select its check box. To turn a feature off, clear its check box. A filled box means that only part of the feature is turned on.

- ■ .NET Framework 3.5 (includes .NET 2.0 and 3.0)
- ■ .NET Framework 4.6 Advanced Services
- ☐ Active Directory Lightweight Directory Services
- ☑ Containers
- ☐ Data Center Bridging
- ☐ Device Lockdown
- ☑ Hyper-V
- ☑ Internet Explorer 11
- ☐ I Provides services and management tools for creating and running virtua
- ☐ Internet Information Services Hostable Web Core
- ☐ Legacy Components
- ☑ Media Features
- ☐ Microsoft Message Queue (MSMQ) Server
- ☑ Microsoft Print to PDF

   c. It's a good idea to restart you machine once docker install is complete
   d. Docker CE for Windows uses Windows-native Hyper-V virtualization. Hence the Hyper-V feature needs to be enabled. Search for "Turn Windows features On or Off". Then on the below window ensure Hyper-V feature is turned on

   e. The whale in the notification area indicates that Docker is running, and accessible from a terminal.
   f. Do a quick test below to confirm docker is working fine. Run "docker version" on terminal



```
MINGW64:/c/Users/vikas
vikas@DESKTOP-FMUP443 MINGW64 ~
$ docker version
Client:
 Version:      17.09.1-ce
 API version:  1.32
 Go version:   go1.8.3
 Git commit:   19e2cf6
 Built:        Thu Dec  7 22:22:26 2017
 OS/Arch:      windows/amd64

Server:
 Version:      17.09.1-ce
 API version:  1.32 (minimum version 1.12)
 Go version:   go1.8.3
 Git commit:   19e2cf6
 Built:        Thu Dec  7 22:28:28 2017
 OS/Arch:      linux/amd64
 Experimental: true

vikas@DESKTOP-FMUP443 MINGW64 ~
$ |
```

   g. Installing Docker for Windows, will also install Docker Compose. If you already had Docker installed, you should check that you have Docker Compose version 1.8 or greater installed. If not, we recommend that you install a more recent version of Docker. Run command "docker-compose –version" to confirm the version installed on your machine

3. For older versions of Windows (7 or greater) use following link to setup docker toolbox. (Though the recommendation is to avoid this setup option as this might not be very stable environment) [https://docs.docker.com/toolbox/toolbox_install_windows](https://docs.docker.com/toolbox/toolbox_install_windows)
4. Download visual studio code using link [Visual Studio Code](). You can use another IDE as long as it support Go.
5. GOPATH : Hyperledger Fabric uses the Go programming language for many of its components.
   Given that we are writing a Go chaincode program, we need to be sure that the source code is located somewhere within the $GOPATH tree. Therefore open the terminal and set the GOPATH and PATH variables as follows
   a. export GOPATH=$HOME/go
   b. export PATH=$PATH:$GOPATH/bin
6. We will be developing applications for Hyperledger Fabric leveraging the Hyperledger Fabric SDK for Node.js, you will need to have version 6.9.x or greater of Node.js installed.
   a. Use this [link]() to download the [node-v6.12.3-x64.msi]() and run the installer
   b. Next in the terminal window run "npm install npm@3.10.10 -g"
   c. Then "npm install --global windows-build-tools". To run this terminal window needs to be open in administrator mode. Among other things it will install python 2.7.
   d. Also run "npm install --global grpc". To run this terminal window needs to be open in administrator mode
7. Next in your terminal windows run following commands
   a. git config --global core.autocrlf false
   b. git config --global core.longpaths true
8. Next create a new directory hlf under home directory (C:\Users\..) i.e. run "mkdir $HOME/hlf"
9. Move to directory hlf and run "git clone https://github.com/hyperledger/fabric-samples.git"
10. Next do "cd fabric-samples"
11. Under fabric-samples, download platform specific binaries by running the following command
   a. curl -ssL [https://goo.gl/8EKzmZ](https://goo.gl/8EKzmZ) | bash
   b. Following binaries and docker images are downloaded

| Binary | Description |
|---|---|
| cryptogen | A tool to generate x509 certificates used to identify and authenticate the components on the network |
| configtxgen | Generates configuration artifacts for ordering service and channel creation |
| configtxlator | A support tool to enable users to translate channel configuration transaction into a readable form |

| Image | Description |
|---|---|
| peer | This image provides an environment for peer nodes |
| Orderer | This image provides an environment for orderer nodes |
| tools | Provides a container with cryptogen and configtxgen |

c. export PATH=<your HLF location>/fabric-samples/bin:$PATH

## 4.4 BUILDING YOUR FIRST NETWORK

The BYFN sample is useful to learn the structure and flow of a typical Hyperledger Fabric network. Each developer should complete this tutorial prior to the hands-on hackfest. The BFYN sample offers a script to quickly bootstrap a HLF network with two organizations, each with two peers and an Ordering Service (OS). It will also join peers to a channel, deploy chaincode and execute transactions.
http://hyperledger-fabric.readthedocs.io/en/release/build_network.html
Please use the above link for step-by-step procedure for this tutorial. This tutorial is key to understand each and every component required to bring up the HLF network. At minimum, do following

12. Go to <your HLF location>/fabric-samples/first-network
    a. run “sudo ./byfn.sh -h” and have a look into help text
    b. next run “sudo ./byfn.sh -m generate”
13. You will see certificates generated using Cryptogen :
    a. You will see as an output, that cryptogen generates x509 certificates. It uses crypto-config.yaml to generate the certificates. Have a look at crypto-config.yaml configuration file. Comments are very useful.
    b. The certificates are stored in folder crypto-config. Observe the tree structure
14. You will see configuration artifacts generated using configtxgen . You can find them in the channel-artifacts/ folder. There you will see
    a. Orderer bootstrap block: Genesis block for the ordering service(genesis.block)
    b. Fabric channel configuration transaction: Channel configuration for the OS(channel.tx)

   c. Anchor peer transaction: The anchor peer of the first organisation(Org1MSPanchors.tx)
   d. Anchor peer transaction: The anchor peer of the second organisation(Org2MSPanchors.tx)

15. You will see configuration artifacts generated using configtxgen . You can find them in the channel-artifacts/ folder. There you will see
16. Run a test
    a. Bring the network up using "sudo ./byfn.sh -m up""
    b. You can stop and clean the network in another terming with "sudo ./byfn.sh -m down"

## 4.5 WRITING YOUR FIRST CHAIN CODE

A chaincode typically handles business logic agreed to by members of the network, so it is similar to a "sma rt contract". The below link present a simple chaincode example and walks you through the purpose of each method in the Chaincode Shim API. This is a must do exercise before hackathon
https://hyperledger-fabric.readthedocs.io/en/latest/chaincode4ade.html
***Notice** : Use the above link only to help write and understand structure of the chaincode. For the directory structure and deploying the sacc.go to the dev network follow the instructions in next section 4.5*

As you go through the above tutorial, try to absorb following

1. Chaincode has two functions "set" and "get" to store/read a key-value pair in the blockchain
2. It implements the Chaincode interface and as with every Chaincode it has Init and Invoke methods.
   a. Init is called during chaincode instantiation to initialize any data
   b. Invoke is called per transaction on the chaincode. Each transaction is either a 'get' or a 'set' on the asset created by Init function. Or Set function may create a new asset by providing new key-value pair
   c. Notice that to access the ledger's state, we will leverage the ChaincodeStubInterface.PutState and ChaincodeStubInterface.GetState functions of the chaincode shim API.
   d. PutState puts the specified `key` and `value` into the transaction's writeset as a data-write proposal. PutState doesn't effect the ledger until the transaction is validated and successfully committed.
   e. GetState returns the value of the specified `key` from the ledger. Note that GetState doesn't read data from the writeset, which has not been committed to the ledger. In other words, GetState doesn't consider data modified by PutState that has not been committed.

## 4.6 CHAINCODE AND DEV MODE

DevMode lets you setup a test environment (development network) to run your chaincode.
***Notice** : We will communicate with the network by using the peer client in cli container. This client will communicate with the peer server in the peer container. At the same time, we will build and run our chaincode in the chaincode container.*

1. Terminal # 1. Create a folder sacc under <YOUR_HLF_LOCATION>/fabric-samples/chaincode.
2. Open the editor in the new folder, create the file sacc.go.
3. Go to "<your HLF location>/fabric-samples/chaincode-docker-devmode/"
4. Now use "sudo docker-compose -f docker-compose-simple.yaml up" to start the development network
5. In another terminal #2, use "docker ps" to confirm each container is running. You should see a peer container, a cli container, a chaincode container and Orderer container
6. Now in Terminal #2 run **"sudo docker exec -it chaincode bash"**. This runs an interactive (-it) bash in the chaincode container
7. Now enter "cd sacc" and then "go build"
8. Now run "CORE_CHAINCODE_ID_NAME=mycc:0 ./sacc". So here we basically named our chaincode mycc
9. Start another terminal # 3 to go into cli container. Run "docker exec -it cli bash'
10. Run "peer chaincode install -p chaincodedev/chaincode/sacc -n mycc -v 0". This installs the chaincode on the peer with name mycc and version 0.
11. Next instantiate the chaincode (call Init) with "peer chaincode instantiate -n mycc -v 0 -c '{"Args":["a","10"]}' -C myc"
12. Now issue an invoke "peer chaincode invoke -n mycc -c '{"Args":["set", "a", "20"]}' -C myc" to change the value of "a" to "20".
13. Finally query a using "peer chaincode query -n mycc -c '{"Args":["query","a"]}' -C myc". We should see a value of 20.

***Notice** : By default, we mount only `sacc`. However, you can easily test different chaincodes by adding them to the `chaincode` subdirectory and relaunching your network. At this point they will be accessible in your `chaincode` container.*

## 4.7 NODE SDK HYPERLEDGER EXAMPLES

Your first point of information about HFC SDK for Node.js is https://github.com/hyperledger/fabric-sdk-node. You will notice we can interact with peers and orderers using fabric-client.

For now, ignore the Fabric-CA client (as we will using Cryptogen tool for this Hackathon).

Hyperledger has couple of good examples. Go through the ones mentioned below and it will be helpful as you go through the sample usecase exercise in section 5

1. Under your fabric-samples/ folder there is an example for Node SDK.
    a. cd balance-transfer
    b. ./runApp.sh
2. Open another terminal and run " ./testAPIs.sh"
3. Go to balance-transfer/artifacts/docker-compose.yaml to understand how the network looks like. After the network has started, runApp.sh starts the app.js with:
4. Explore app.js. It includes following. Try to focus on those marked in bold
    a. it sets and starts a server with Express
    b. it defines REST endpoints to:
    **c. Register and enroll users**
    d. Create Channel
    e. Join Channel
    f. Install chaincode on target peers
    g. Instantiate chaincode on target peers
    **h. Invoke transaction on chaincode on target peers**
    **i. Query on chaincode on target peers**
    j. Query Get Block by BlockNumber
    k. Query Get Transaction by Transaction ID
    l. Query Get Block by Hash
    m. Query for Channel Information
    n. Query to fetch all Installed/instantiated chaincodes
    o. Query to fetch channels
5. Focus on invoke transaction on chaincode. For that, open ./app/invoke-transaction.js. Please see the structure of a request required by channel.sendTransactionProposal.
6. Also open testAPIs.sh and you can see the structure of the REST calls.

Another good example to go through is https://hyperledger-fabric.readthedocs.io/en/latest/write_first_app.html

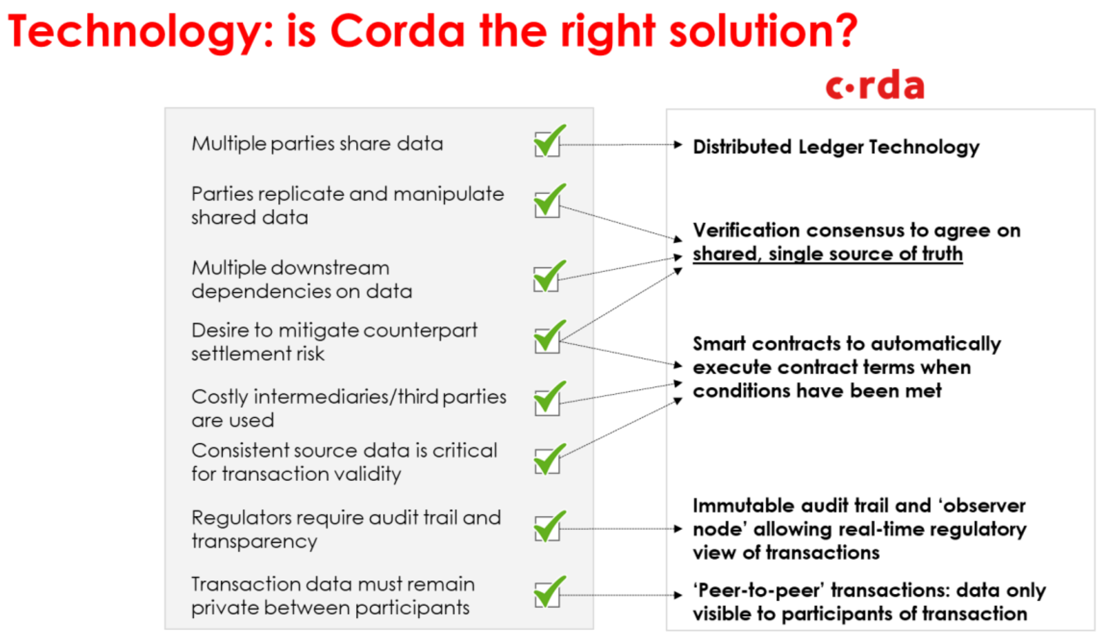
## 4.8 COGNIZANT USE CASE

Please review a Hyperledger Fabric example use case here:

**Document:** "CS Blockchain Hackthon Jan 2018 – Cognizant Use Case"
**Link:** https://cognizantglobal.box.com/s/m8egcvxpvlxcmfz7m84scjntungln3t3
**Password:** CS2018

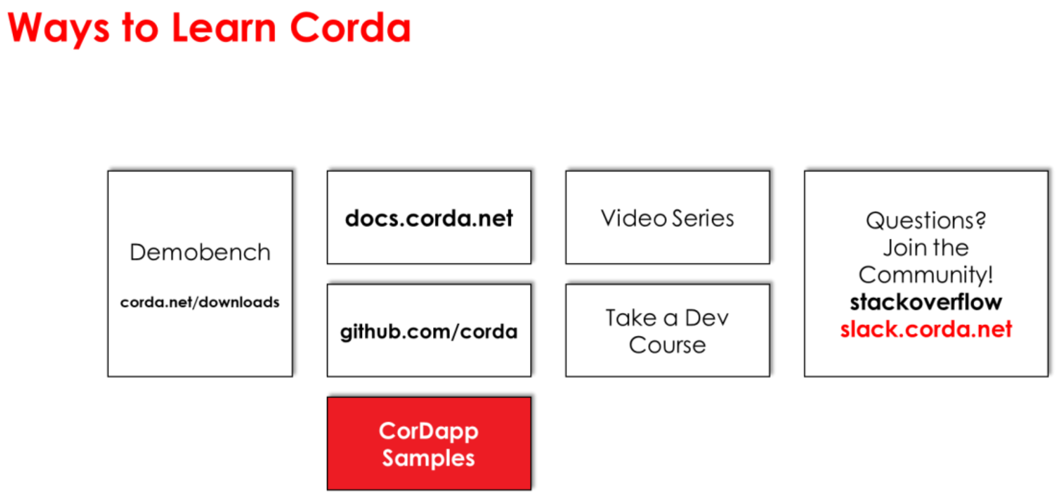
# 5 Develop solution using R3 Corda

Please use this section if you are leveraging Corda


Technology: is Corda the right solution?
corda
Multiple parties share data
Parties replicate and manipulate shared data
Multiple downstream dependencies on data
Desire to mitigate counterpart settlement risk
Costly intermediaries/third parties are used
Consistent source data is critical for transaction validity
Regulators require audit trail and transparency
Transaction data must remain private between participants
Distributed Ledger Technology
Verification consensus to agree on shared, single source of truth
Smart contracts to automatically execute contract terms when conditions have been met
Immutable audit trail and 'observer node' allowing real-time regulatory view of transactions
'Peer-to-peer' transactions: data only visible to participants of transaction

## 5.1 INTRODUCTION TO CORDA

This section provides information relevant to learning Corda. Depending on one's learning style, there are multiple approaches:



For those less technical and more visual, we recommend starting with the corda demobench download. A video is available describing the demo walkthrough. Note that this is for business demo-ing purposes where clients may not have wifi access.
For those who prefer a video series, we have a 2.5 hour split of key concepts modules that mimic the first half-day of the Corda Developer Training.
For developers, we recommend starting in our docs site. There's a quick start guide to help set up your developer environment. These links will ultimately lead you to the open source corda github repository.
After understanding the basic templates and built-in samples, our dev relations team maintains separate open source CorDapp samples that can be leveraged as starting points. These will demonstrate some of the main reusable functionalities and design patterns.
If there are any questions or issues, our first line of support is through stackoverflow (tag: corda) or directly in our cordaledger slack channel. Here, you can ask our directly ask our devs questions on designs and bug fixes.

## 5.2 DESIGNING ON CORDA

The standard approach to design a CorDapp is to follow these major areas:

1. **Parties** (nodes & identities)
2. **States** (data models)
3. **Contracts** (verify())
4. **Transactions** (evolution of data)
5. **Flows** (Consensus)

To best inform developers of the functionality, we've used different templates and mock-ups to demonstrate the design patterns that would be used.

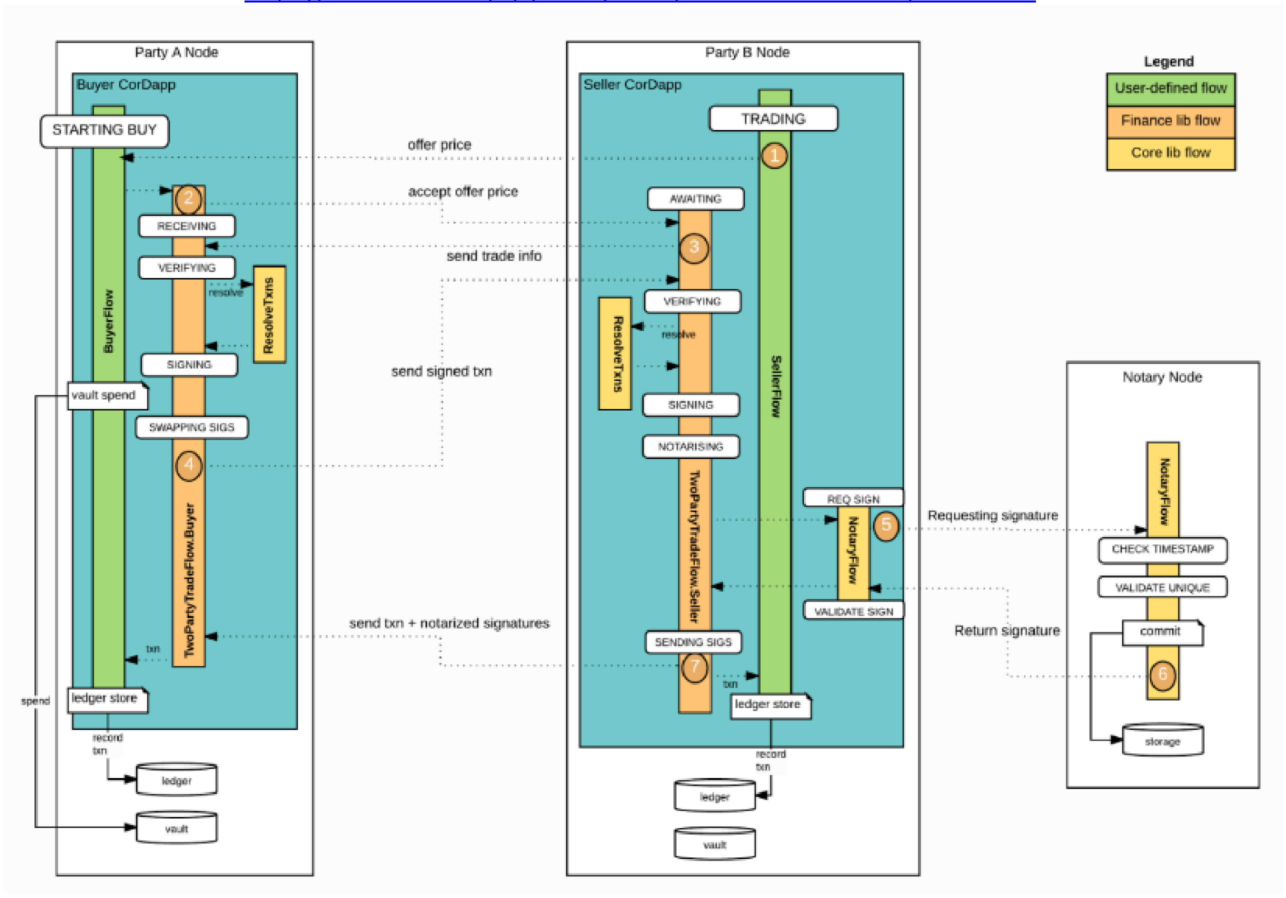
| Questions | What we Learn / Why we Ask |
|---|---|
| • Who are the actors in this use case?<br>• Who needs to own the data (either for regulatory or other reasons)? | • Fills out our "Parties" page. This determines who will be hosting a Corda Node and where this application would be deployed |
| • Are there special nodes/actors with elevated permissions (e.g. regulator, central bank, etc) | • Provides more information on "node types" and split of CorDapps |
| • What data is being moved around?<br>• Do these fields have existing standards? | • Fills out the initial "State" of what's being shared between parties<br>• Important to use existing ISO standards for fields |
| • Is this data linked to a legal document?<br>• Please provide a sample if relevant | • Helps with "State" and will be used as an attachment for the legal prose |
| • Are there core scenarios that you can gather from the given Full Architecture diagram (see below - Figure 1) | • Helps with initial mapping of scenarios and general design patterns available<br>• |
| • In which ways does this data evolve over time?<br>• Please list some general user stories. | • Starts the writing of "contracts" that define the verification of the new state |
| • For each user story, which parties are saving/review the data?<br>• Note any privacy requirements for the data (i.e. when you receive the state, are you allowed to see previous transactions? Need for chain of ownership?) | • Begins the creation of transactions and mechanisms for the states |
| • Are there examples of existing applications that cover this use case?<br>• Screens or any documentation is useful for the document creation process | • More color to understand the different states of data and what would be shared "on-ledger" vs kept private within companies "off-ledger" |

First, as a designer, it's important to shift the thinking from a central webapp published to a cloud provider into a shared decentralized app that gets installed on multiple corda nodes. Since Corda is building a decentralized network, each party will have control of its own node, installed applications, and contents (via private-public keys).
Before starting the design, think about who will own a corda node and what they will need to store on-ledger (shared facts) and off-ledger (private or temporary data). These corda nodes and their identities are crucial for understanding the movement of data via these protocols across multiple nodes.
**States** are java objects passed between parties. These are rows in the database (vault) which are queryable and represent your overall ownership of this data.
Each state is governed by **contract code**. Traditional contract code logic is Boolean – you either pass the check, which leads to a private key signature, or it fails, which leads to a rejection of the transaction/change. The contract code Boolean checks are under the verify() function for each state evolution.
**Transactions** are mechanisms to group one or more state changes. In the Bitcoin world, a transaction is one party (Alice) sending over her permission to spend X bitcoins to a public address (Bob). The transaction is informing the database to update the ledger by

giving spending permission to the new public address. In the Corda world, we're effectively doing the same thing but with more complex transactions. We're able to take the smart contract code and add further restrictions and variations to negotiate more complex changes to each parties' corda database (vault).

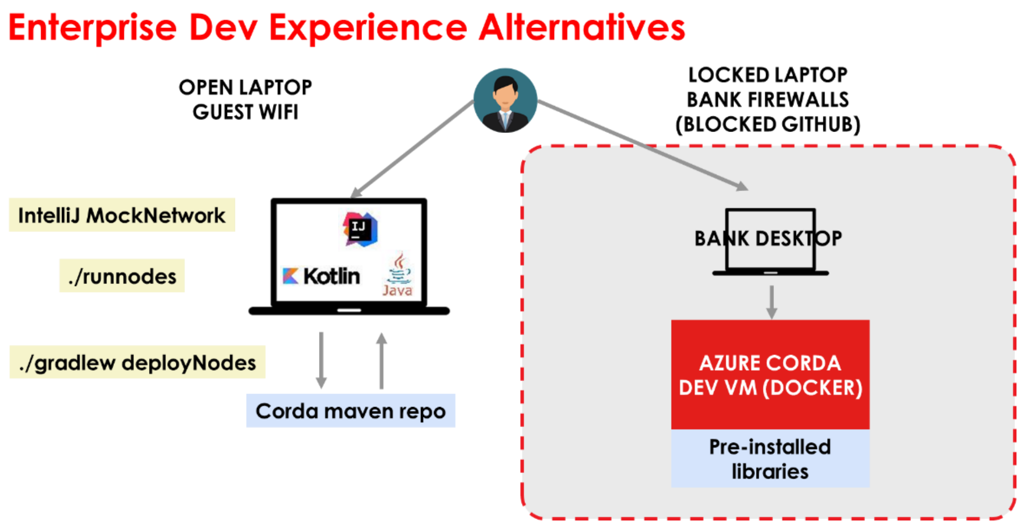
**Flows** are the way in which we reach real-time consensus for the transaction change. The flow controls the passing around of transactions between different parties for collecting signatures. Once all signatures are collected, the transactions can be committed to each corda node's database. This is all written within the cordapp in a generic way of indicating initiators and receivers of the states.

Note that the most generic flows have already been written in a flow library which can be called with subflow() routines. These flows can be found in the API docs: https://docs.corda.net/api/kotlin/corda/net.corda.core.flows/index.html



## 5.3 CORDA DEVELOPER EXPERIENCE

We understand most enterprise institutions have firewalls, blocked sites, and locked down machines.

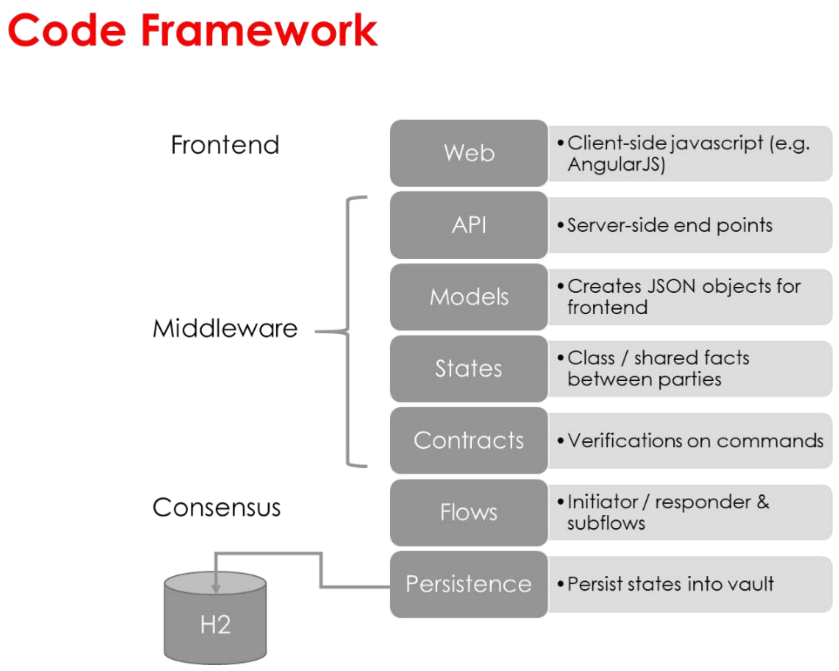


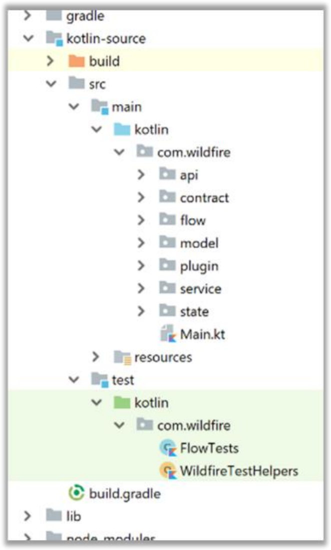
For the January 2018 CS Blockchain Hackfest, Credit Suisse will have provided laptops with pre-installed JDK and IntelliJ (IDE) with access to the public Guest WiFi.
Note: WiFi access is required when running the gradlew command as this pulls the indicated version of corda from the maven repository. Most of the documentation and github samples sites should already be whitelisted for regular developer review.
The instructions for installations on Windows or Mac are in videos, but also available on the docs site.
For the institutions that are unable to provide open laptops and non-firewall internet access, we have seen a few alternatives:

1. Microsoft Azure Dev VM (Docker) – These machines can be procured through Azure and access can be granted through command line jump-boxes or X-interfaces to these VMs. Corda has a docker image that can be used for this.

2. Bank-packaged Installations – This method is dependent on the vendor management restrictions of the institution

After the proper pre-requisite libraries and tools are installed, it is easy to get started by cloning the corda github repository.
Since Corda is written with the JVM, our developers can leverage existing Java tools and write their CorDapps in any language that compiles to the JVM (e.g. Kotlin, Scala, etc). Internally, we prefer IntelliJ and coding with Kotlin.
Corda's data is saved into an H2 database in a standard SQL-queryable format. One can access the database by installing the H2 Console and connecting to the database instance.
When coding on Corda, developers are writing CorDapps (Corda Distributed Applications). This contains the following framework:


Code Framework
Frontend
Middleware
Consensus
H2
Web
•Client-side javascript (e.g. AngularJS)
API
•Server-side end points
Models
•Creates JSON objects for frontend
States
•Class / shared facts between parties
Contracts
•Verifications on commands
Flows
•Initiator / responder & subflows
Persistence
•Persist states into vault

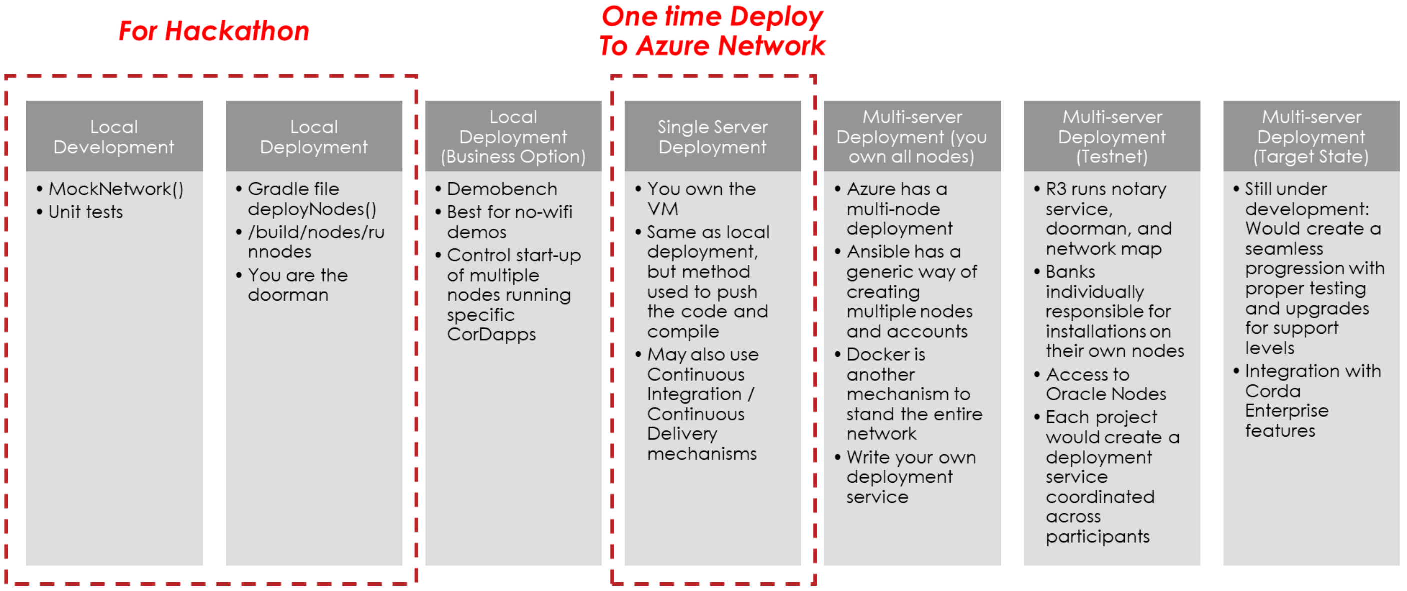


The build.gradle file within this cordapp specifies the number of cordapp nodes and their configurations (i.e. node.conf file). This is a deploy mechanism for a single machine local deployment. See next section.
Testing and Debugging DOES NOT REQUIRE constant starting of corda nodes via *runnodes*! One can test cordapps in real time via junit tests. A blogpost is available that provides details for writing these tests.
One can write multiple cordapps within a single project. This is recommended for creating multiple node permissions.
Ultimately, your cordapp will compile into a .jar file. This .jar file when placed within the /plugins folder of your corda node will combine all the coded packages and instructions. The developer experience's interaction with the deployment of production-level cordapps will require vendor deployment coordination across multiple institutions.
CorDapps will be discoverable within a Corda Connect user experience. This allows for members and corda node owners the ability to review different production-ready CorDapps for further marketing and integration into existing systems.

### 5.4 DEPLOYMENT OPTIONS

The Hackathon preferred method of development mostly occurs in the MockNetwork() local testing. Internally, your team can present and locally deploy via the runnodes format.

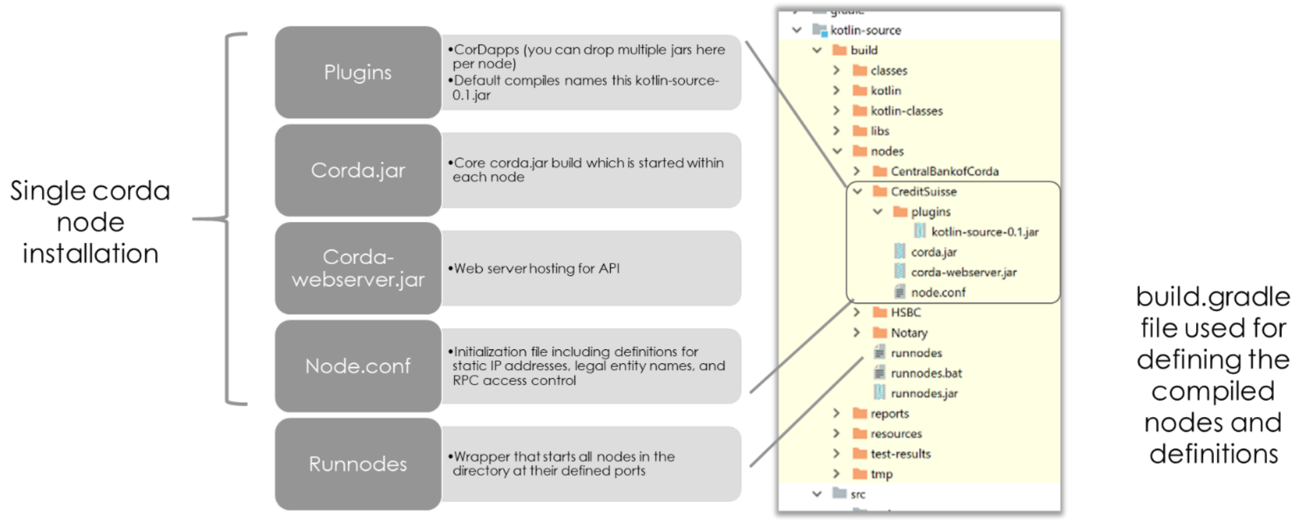
## Overview of Stages of Deployment



| Local Development | Local Deployment | Local Deployment (Business Option) | Single Server Deployment | Multi-server Deployment (you own all nodes) | Multi-server Deployment (Testnet) | Multi-server Deployment (Target State) |
|---|---|---|---|---|---|---|
| • MockNetwork()<br>• Unit tests | • Gradle file deployNodes()<br>• /build/nodes/runnodes<br>• You are the doorman | • Demobench<br>• Best for no-wifi demos<br>• Control start-up of multiple nodes running specific CorDapps | • You own the VM<br>• Same as local deployment, but method used to push the code and compile<br>• May also use Continuous Integration / Continuous Delivery mechanisms | • Azure has a multi-node deployment<br>• Ansible has a generic way of creating multiple nodes and accounts<br>• Docker is another mechanism to stand the entire network<br>• Write your own deployment service | • R3 runs notary service, doorman, and network map<br>• Banks individually responsible for installations on their own nodes<br>• Access to Oracle Nodes<br>• Each project would create a deployment service coordinated across participants | • Still under development: Would create a seamless progression with proper testing and upgrades for support levels<br>• Integration with Corda Enterprise features |

- Local development - [https://www.corda.net/2017/11/testing-cordapps/](https://www.corda.net/2017/11/testing-cordapps/)
  - MockNetwork() creates these locally through IntelliJ and closes them at the end of the unit tests
- Local deployment – your gradle file will provide instructions for build via deployNodes() which creates folders within your /build/nodes
  - /build/nodes/runnodes – starts nodes at given ports in localhost to do a single machine local deployment of set nodes (you are your own doorman)

## Corda node – Gradlew deployNodes


Single corda node installation
build.gradle file used for defining the compiled nodes and definitions

| | |
|---|---|
| Plugins | •CorDapps (you can drop multiple jars here per node)<br>•Default compiles names this kotlin-source-0.1.jar |
| Corda.jar | •Core corda.jar build which is started within each node |
| Corda-webserver.jar | •Web server hosting for API |
| Node.conf | •Initialization file including definitions for static IP addresses, legal entity names, and RPC access control |
| Runnodes | •Wrapper that starts all nodes in the directory at their defined ports |

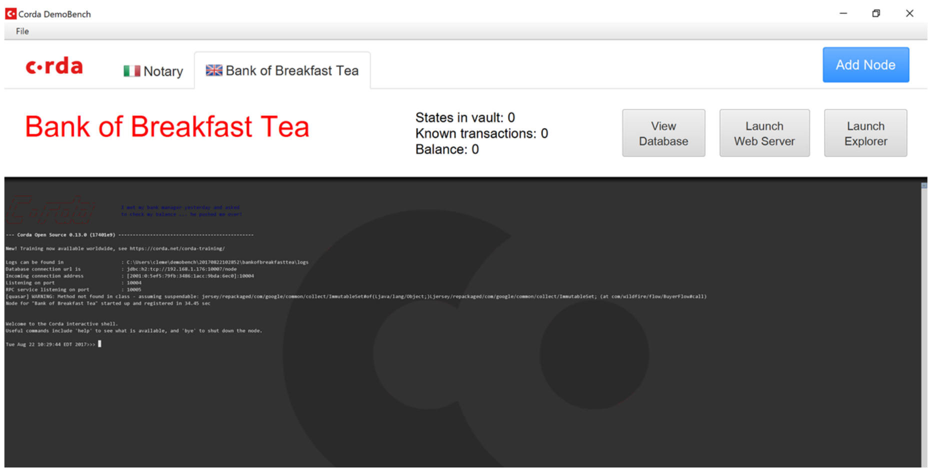
```
kotlin-source
  build
    classes
    kotlin
    kotlin-classes
    libs
    nodes
      CentralBankofCorda
      CreditSuisse
        plugins
          kotlin-source-0.1.jar
        corda.jar
        corda-webserver.jar
        node.conf
      HSBC
      Notary
      runnodes
      runnodes.bat
      runnodes.jar
    reports
    resources
    test-results
    tmp
  src
```

- Local deployment (business option) – Demobench is a great business tool for no-wifi demos with our clients. You can take your cordapp.jar file output and upload them to the dynamic building of nodes
  - Each node can run the node explorer and webserver so you can also demo within a local build

## Demobench – for Business / No wifi

**[https://www.corda.net/downloads/](https://www.corda.net/downloads/) | [https://docs.corda.net/demobench.html](https://docs.corda.net/demobench.html)**

**Starts local nodes on your computer with the ability to attach and simulate your own CorDapps as well as review the Command Line Client, Node Explorer, H2 database explorer, and custom Web Interfaces**



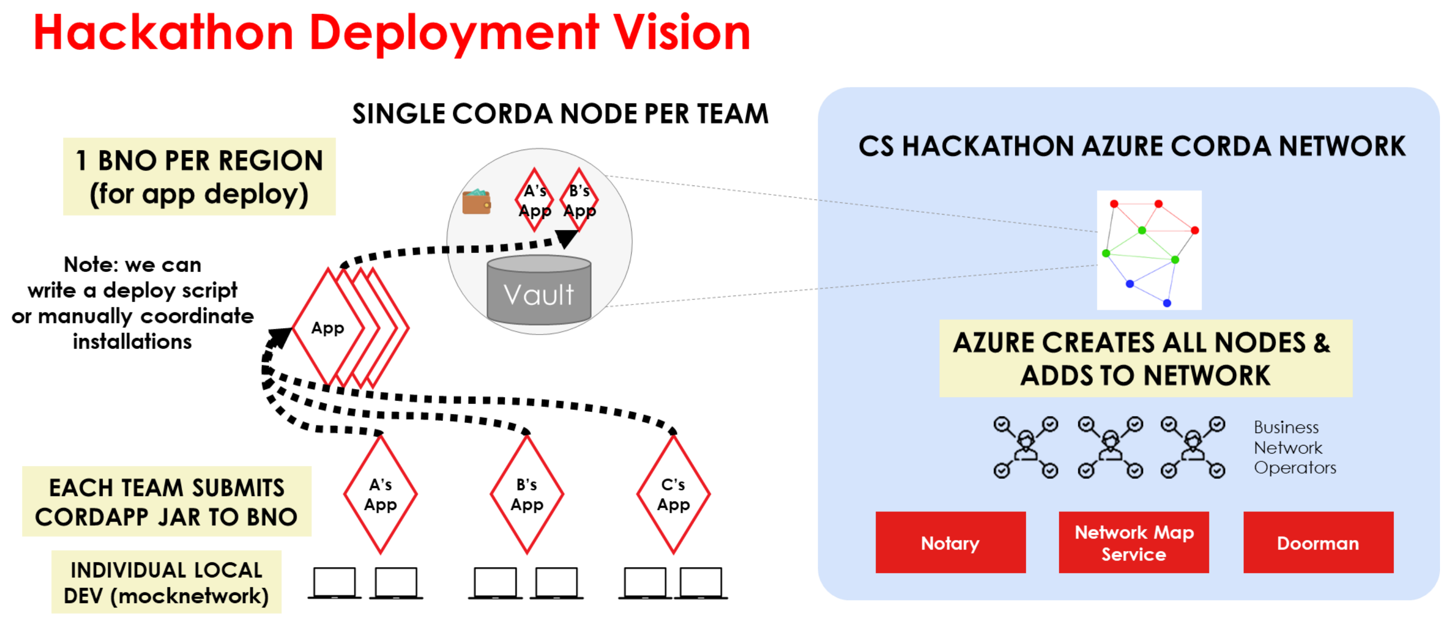
States in vault: 0
Known transactions: 0
Balance: 0

```
I met my bank manager yesterday and asked
to check my balance ... he pushed me over!

--- Corda Open Source 0.13.0 (17401e9) -----------------------------------------------

New! Training now available worldwide, see https://corda.net/corda-training/

Logs can be found in                    : C:\Users\cleme\demobench\20170822102852\bankofbreakfasttea\logs
Database connection url is              : jdbc:h2:tcp://192.168.1.176:10007/node
Incoming connection address             : [2001:0:5ef5:79fb:3486:1acc:9bda:6ec0]:10004
Listening on port                       : 10004
RPC service listening on port           : 10005
[quasar] WARNING: Method not found in class - assuming suspendable: jersey/repackaged/com/google/common/collect/ImmutableSet#of(Ljava/lang/Object;)Ljersey/repackaged/com/google/common/collect/ImmutableSet; (at com/wildfire/flow/BuyerFlow#call)
Node for "Bank of Breakfast Tea" started up and registered in 34.45 sec


Welcome to the Corda interactive shell.
Useful commands include 'help' to see what is available, and 'bye' to shut down the node.

Tue Aug 22 10:29:44 EDT 2017>>>
```

- Single server deployment (you own the VM)
    - Same as a local deployment, but the VM is owned by a single party. The single server can more easily upgrade cordapp.jar and restart the node.
    - It is also possible to horizontally scale the web server piece. In this case, a separate series of scripts would be required to ensure the cordapp API deployment is synchronized with the web server upgrades.
    - Previous corda projects have also developed Continuous Integration / Continuous Deployment processes for pulling code repositories and scripting compiling and moving of files
- Multi-server deployment (you own all nodes)
    - This is the ideal Azure deployment for creating a separate private network. Note that the doorman service and process is currently maintained with r3. Technically, any party can deploy their own mechanism for adding nodes to the network map service.
    - Azure Marketplace ARM template: [https://azuremarketplace.microsoft.com/en-us/marketplace/apps/corda.r3corda?tab=Overview](https://azuremarketplace.microsoft.com/en-us/marketplace/apps/corda.r3corda?tab=Overview)
    - Ansible is more generic way to create internal test Corda Compatibility Zone. [https://github.com/corda/corda-ansible](https://github.com/corda/corda-ansible)
    - The corda docker is updated: [https://github.com/corda/corda-docker](https://github.com/corda/corda-docker)
    - Known nodes needed: IP address for a network map and IP address for a notary
    - Separate participant nodes: separate coordination of creating node.conf files and registration through a doorman to a network
    - Deployment service: this is usually the role of a “Business Network Operator” which will run a service that keeps track of CorDapp installations and which Corda node has which “role” (i.e. central bank issuer, regulatory node listener, etc)
- Multi-server Deployment (Corda Connect Testnet)

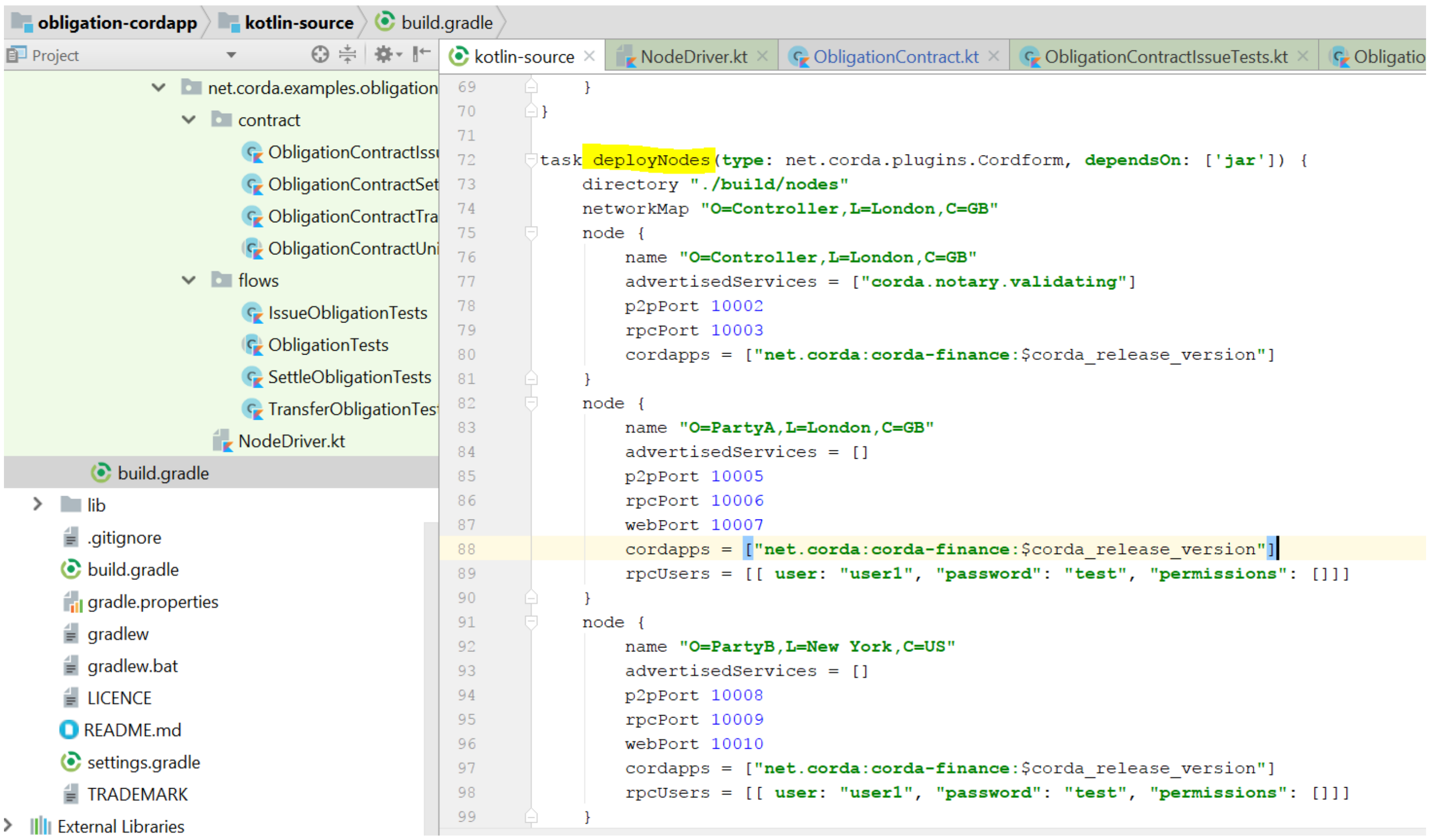
  - R3 runs the Notary, Network Map Service, and Doorman
  - Banks individually are responsible for their own node – you can only control what you have installed on your node (Firewall exceptions take a long time)
  - As a member of Testnet, the legal entity is responsible for configuring the node.conf file and using –-initial-registration command to be added to the Compatibility Zone
  - Each project would create a deployment service to coordinate between participants – this would install the cordapp across multiple nodes
  - Access to Oracle nodes – Thompson Reuters and Accuity have nodes on Testnet that allow you to connect to their oracle data provider services
- Multi-server Deployment (Corda Connect Staging / Production)
  - Still under development, but we're looking for the best way to make seamless progression with proper testing for upgrades to support a production-level SLA

Specifically for the Hackthon, we've defined the following one-time deployment and perspective of the operational corda network:


Hackathon Deployment Vision
SINGLE CORDA NODE PER TEAM
1 BNO PER REGION (for app deploy)
Note: we can write a deploy script or manually coordinate installations
A's App
B's App
Vault
App
EACH TEAM SUBMITS CORDAPP JAR TO BNO
INDIVIDUAL LOCAL DEV (mocknetwork)
A's App
B's App
C's App
CS HACKATHON AZURE CORDA NETWORK
AZURE CREATES ALL NODES & ADDS TO NETWORK
Business Network Operators
Notary
Network Map Service
Doorman

## 5.5 CORDAPP OBLIGATION EXAMPLE WALK THROUGH

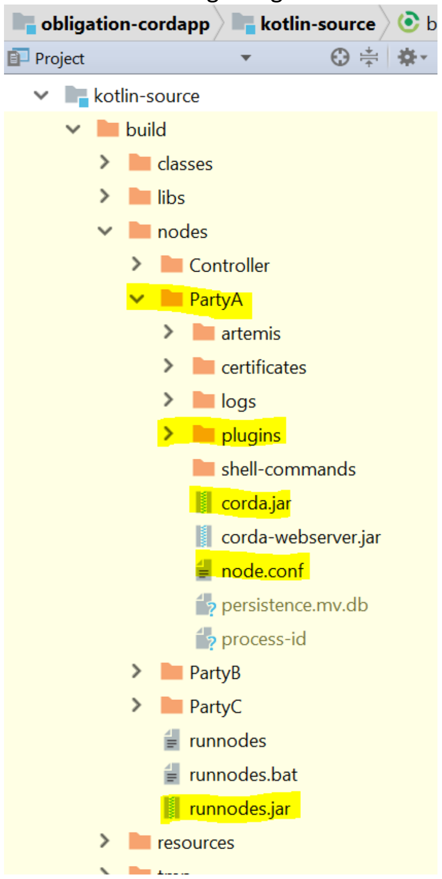
This will be covered in a recorded training. The code for the Obligation CorDapp is open source and available on github written by Roger Willis, R3's head of Developer Relations.
Following the README, we can **clone the source**, compile via **gradlew deploynodes** and deploy locally via **runnodes** command. The runnodes command will start all corda nodes in the given directory and create a private network map on your machine as defined by the build.gradle file:



```
net.corda.examples.obligation
    contract
        ObligationContractIss
        ObligationContractSet
        ObligationContractTra
        ObligationContractUni
    flows
        IssueObligationTests
        ObligationTests
        SettleObligationTests
        TransferObligationTes
    NodeDriver.kt
build.gradle
lib
.gitignore
build.gradle
gradle.properties
gradlew
gradlew.bat
LICENCE
README.md
settings.gradle
TRADEMARK
External Libraries
```

```
    }
}

task deployNodes(type: net.corda.plugins.Cordform, dependsOn: ['jar']) {
    directory "./build/nodes"
    networkMap "O=Controller,L=London,C=GB"
    node {
        name "O=Controller,L=London,C=GB"
        advertisedServices = ["corda.notary.validating"]
        p2pPort 10002
        rpcPort 10003
        cordapps = ["net.corda:corda-finance:$corda_release_version"]
    }
    node {
        name "O=PartyA,L=London,C=GB"
        advertisedServices = []
        p2pPort 10005
        rpcPort 10006
        webPort 10007
        cordapps = ["net.corda:corda-finance:$corda_release_version"]
        rpcUsers = [[ user: "user1", "password": "test", "permissions": []]]
    }
    node {
        name "O=PartyB,L=New York,C=US"
        advertisedServices = []
        p2pPort 10008
        rpcPort 10009
        webPort 10010
        cordapps = ["net.corda:corda-finance:$corda_release_version"]
        rpcUsers = [[ user: "user1", "password": "test", "permissions": []]]
    }
```

Note that executing this gradle command will create the same directories:



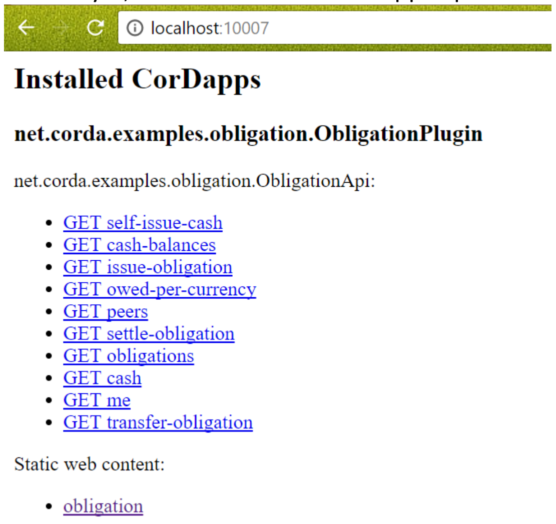
```
kotlin-source
    build
        classes
        libs
        nodes
            Controller
            PartyA
                artemis
                certificates
                logs
                plugins
                shell-commands
                corda.jar
                corda-webserver.jar
                node.conf
                persistence.mv.db
                process-id
            PartyB
            PartyC
            runnodes
            runnodes.bat
            runnodes.jar
        resources
```

The web servers started can be accessed:

1. PartyA: `http://localhost:10007`

2. PartyB: `http://localhost:10010`
3. PartyC: `http://localhost:10013`

For Party A, note the installed cordapp exposed APIs and available web content:



localhost:10007

Installed CorDapps

net.corda.examples.obligation.ObligationPlugin

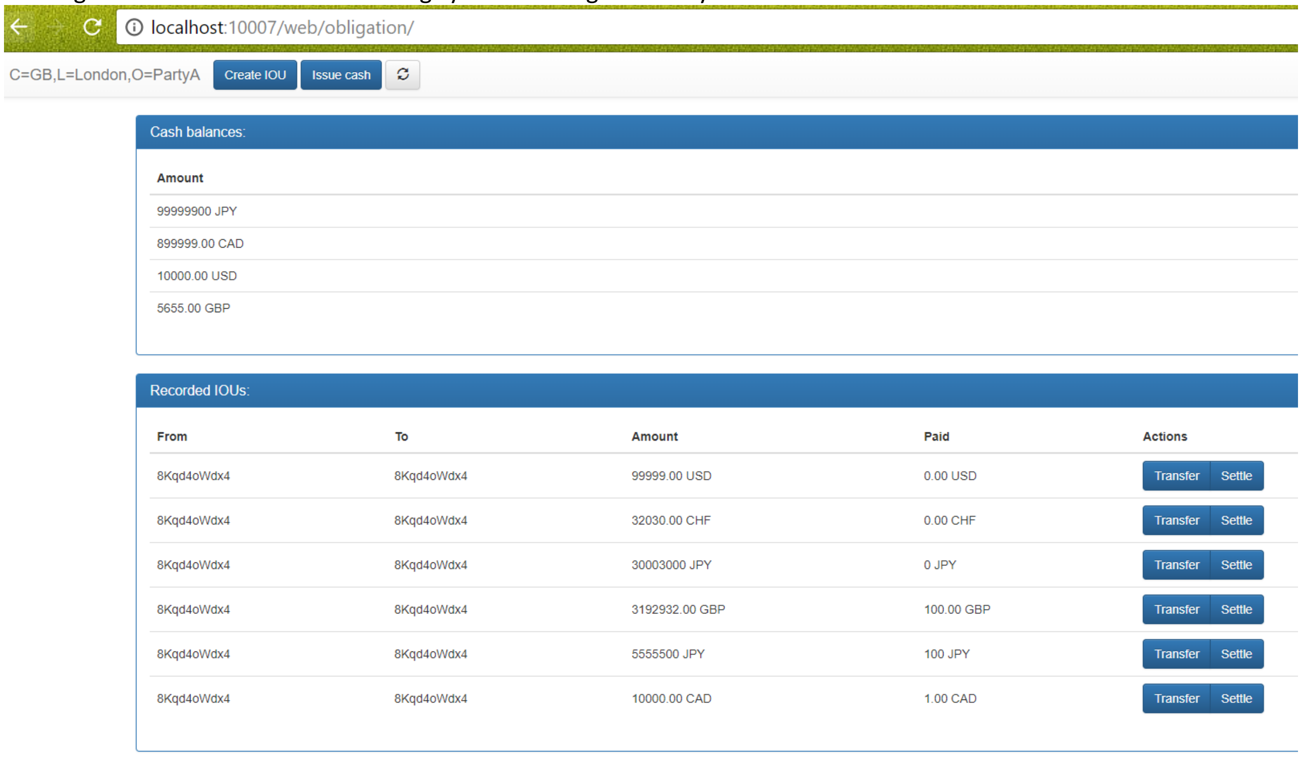
net.corda.examples.obligation.ObligationApi:

- GET self-issue-cash
- GET cash-balances
- GET issue-obligation
- GET owed-per-currency
- GET peers
- GET settle-obligation
- GET obligations
- GET cash
- GET me
- GET transfer-obligation

Static web content:

- obligation

Clicking on the static web content brings you to the login of Party A:



localhost:10007/web/obligation/

C=GB,L=London,O=PartyA

Cash balances:

| Amount |
|---|
| 99999900 JPY |
| 899999.00 CAD |
| 10000.00 USD |
| 5655.00 GBP |

Recorded IOUs:

| From | To | Amount | Paid | Actions |
|---|---|---|---|---|
| 8Kqd4oWdx4 | 8Kqd4oWdx4 | 99999.00 USD | 0.00 USD | Transfer Settle |
| 8Kqd4oWdx4 | 8Kqd4oWdx4 | 32030.00 CHF | 0.00 CHF | Transfer Settle |
| 8Kqd4oWdx4 | 8Kqd4oWdx4 | 30003000 JPY | 0 JPY | Transfer Settle |
| 8Kqd4oWdx4 | 8Kqd4oWdx4 | 3192932.00 GBP | 100.00 GBP | Transfer Settle |
| 8Kqd4oWdx4 | 8Kqd4oWdx4 | 5555500 JPY | 100 JPY | Transfer Settle |
| 8Kqd4oWdx4 | 8Kqd4oWdx4 | 10000.00 CAD | 1.00 CAD | Transfer Settle |

Note: This is the specific view pulling from Party A's corda node. Since Corda does not replicate transaction data, the view from Party A is unique to what's shown in the database. This does not represent any additional layer of UI filtered subset of data.
Walking through the Obligation CorDapp scenarios shows many of the different design patterns that may be relevant to other real-life use cases:

## Obligation CorDapp Patterns / Scenarios

1) **Self-issue cash –** pattern creates a corda state representing a "type" (e.g. USD) – fungible
2) **Issue an IOU between two parties –** borrower creates the IOU shared with other party
3) **Settle a subset amount of the IOU –** pattern allows you to create a transaction that consumes cash of a given "type". Contract code complex to cover all parameters. Partial settles possible. Only the borrower can update/settle.
4) **Settle a full amount of IOU –** pattern includes a threshold "exit" of the IOU when all the amount is paid off
5) **Transfer the IOU to another party –** another party pays for the IOU of set amount and now can settle values

**A lot of other samples: https://www.corda.net/samples/**

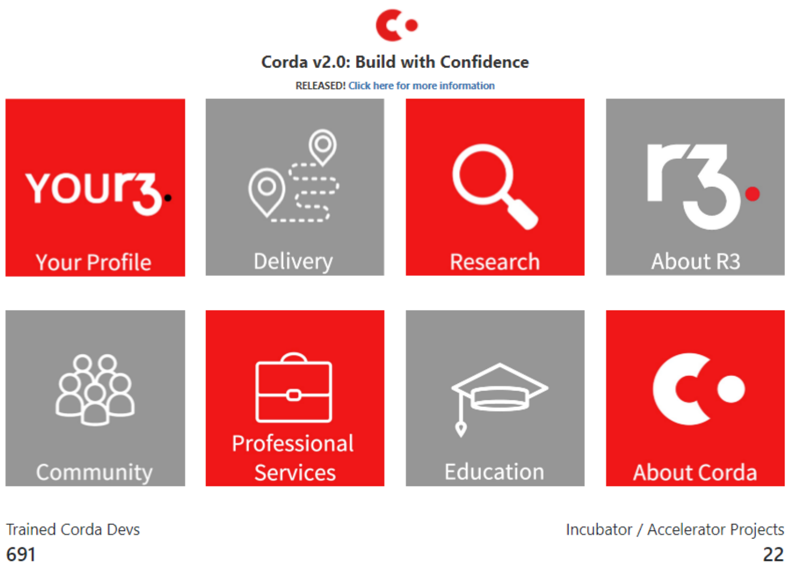
## 5.6 PRE-HACKFEST DEVELOPER CHECKLIST

These should be the bare minimum starting point prior to the start of the hackathon.

- ❑ **Environment Setup (on your laptop) -** https://docs.corda.net/getting-set-up.html
- ❑ **Review Corda Key Concepts -** https://docs.corda.net/key-concepts.html
- ❑ **Review Wildfire Design Doc for Designing on Corda inspiration**
- ❑ **Review available Corda Samples -** https://www.corda.net/samples/
- ❑ **Join our Slack and StackOverflow –** https://slack.corda.net

## 5.7 FURTHER ENGAGING WITH R3

As Credit Suisse is a member of r3, we also have a personalized r3 portal confluence wiki. Your key contacts can provide you with more information on this to see previous lists of completed and successful r3 projects with Credit Suisse and other r3 members.


Corda v2.0: Build with Confidence

RELEASED! Click here for more information

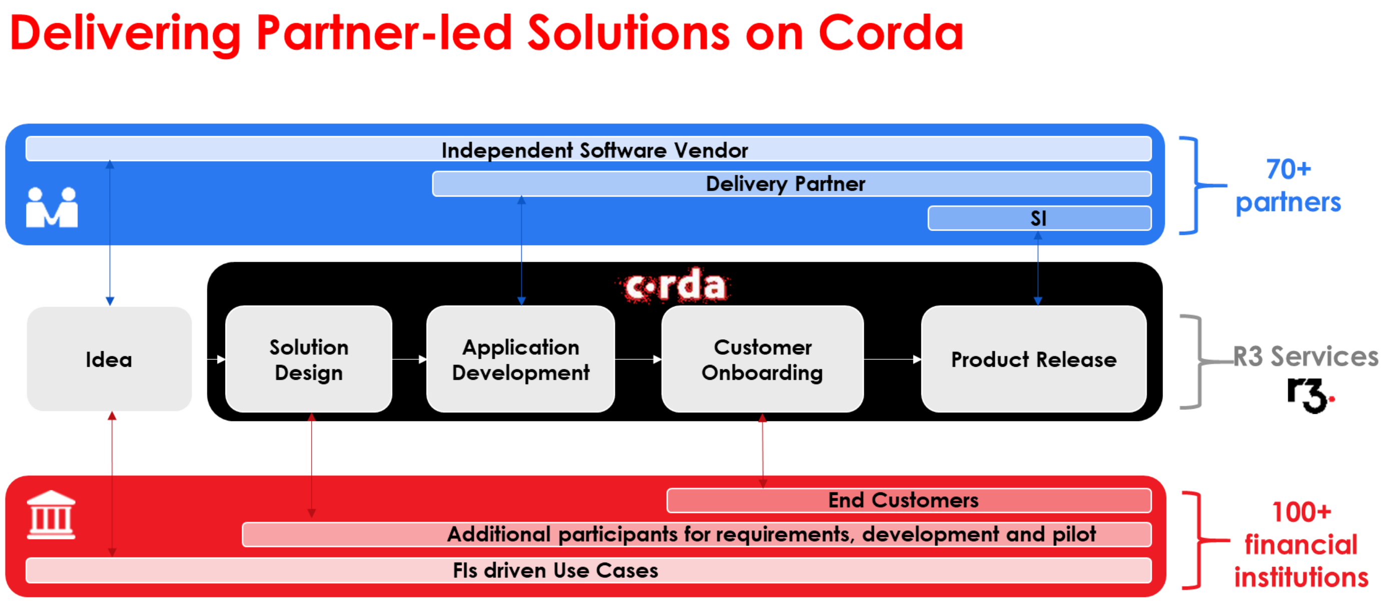
Trained Corda Devs

691

Incubator / Accelerator Projects

22

R3 as an organization has committed to helping the enterprise community and ecosystem better understand the industry through shared POCs and leveraged work. R3 Services leverages previously completed work and mutualizes developer resources between multiple organizations (sometimes competitors) to achieve a clearer opinion on the use case.
We realize that the operational effort to move to production will require a lot more resources than the r3 workforce (~160 global employees), therefore we've grown our partner group to connect the resources towards production installations. A full list of our partners is available on [http://www.r3.com/partner/](http://www.r3.com/partner/)


Delivering Partner-led Solutions on Corda
Independent Software Vendor
Delivery Partner
SI
70+ partners
corda
Idea
Solution Design
Application Development
Customer Onboarding
Product Release
R3 Services
r3.
End Customers
Additional participants for requirements, development and pilot
FIs driven Use Cases
100+ financial institutions

## 5.8 CORDA TRAINING MATERIALS

Please download all Corda training materials here:

**File:** “CS MSFT Hackathon Corda Material Final.zip”
**Link:** [https://r3share.mohso.com/dl/jirN1qU7hu](https://r3share.mohso.com/dl/jirN1qU7hu)
**Password:** LW4ogeP5