Chapter 13: Installing your network on Bluemix/Kubernetes

Learning Bluemix & Blockchain

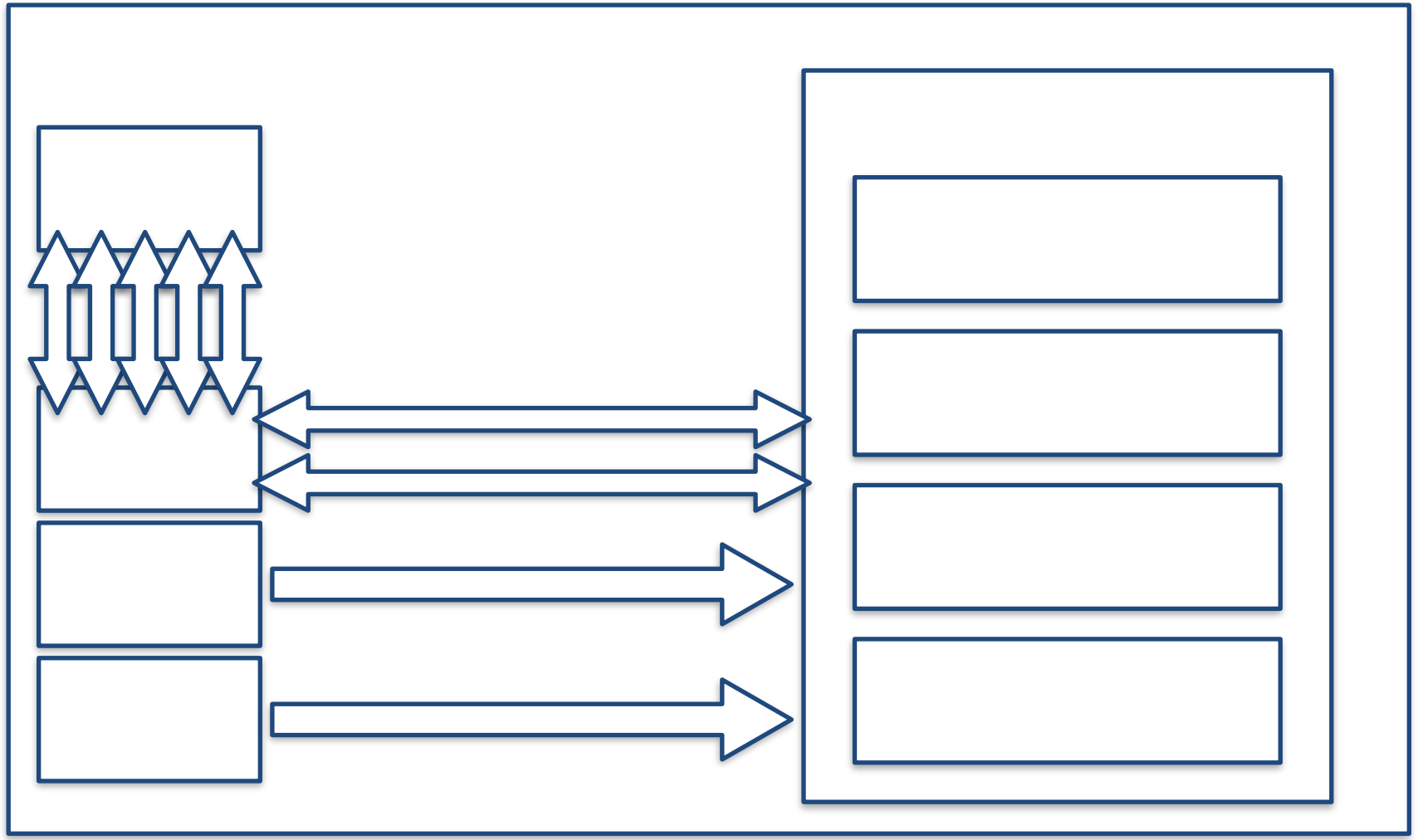
Bob Dill, IBM Distinguished Engineer, CTO Global Technical Sales
David Smits, Senior Certified Architect, IBM Blockchain

The Plan: 30 minute Chapters with an hour or two of practice

Chapter 1:	What is Blockchain? Concept and Architecture overview
Chapter 2:	What's the story we're going to build
Chapter 2.1:	Architecture for the Story
Chapter 3:	Set up local HyperLedger V1 development environment
Chapter 4:	Build and test the network
Chapter 5:	Administration User Experience
Chapter 6:	Buyer Support and User Experience
Chapter 7:	Seller Support and User Experience
Chapter 8:	Shipper Support and User Experience
Chapter 9:	Provider Support and User Experience
Chapter 10:	Finance Company Support and User Experience
Chapter 11:	Combining for Demonstration
Chapter 12:	Events and Automating for Demonstration
Chapter 13:	Running your Blockchain on the IBM Cloud Free Cluster
Chapter 14:	Debugging in Docker

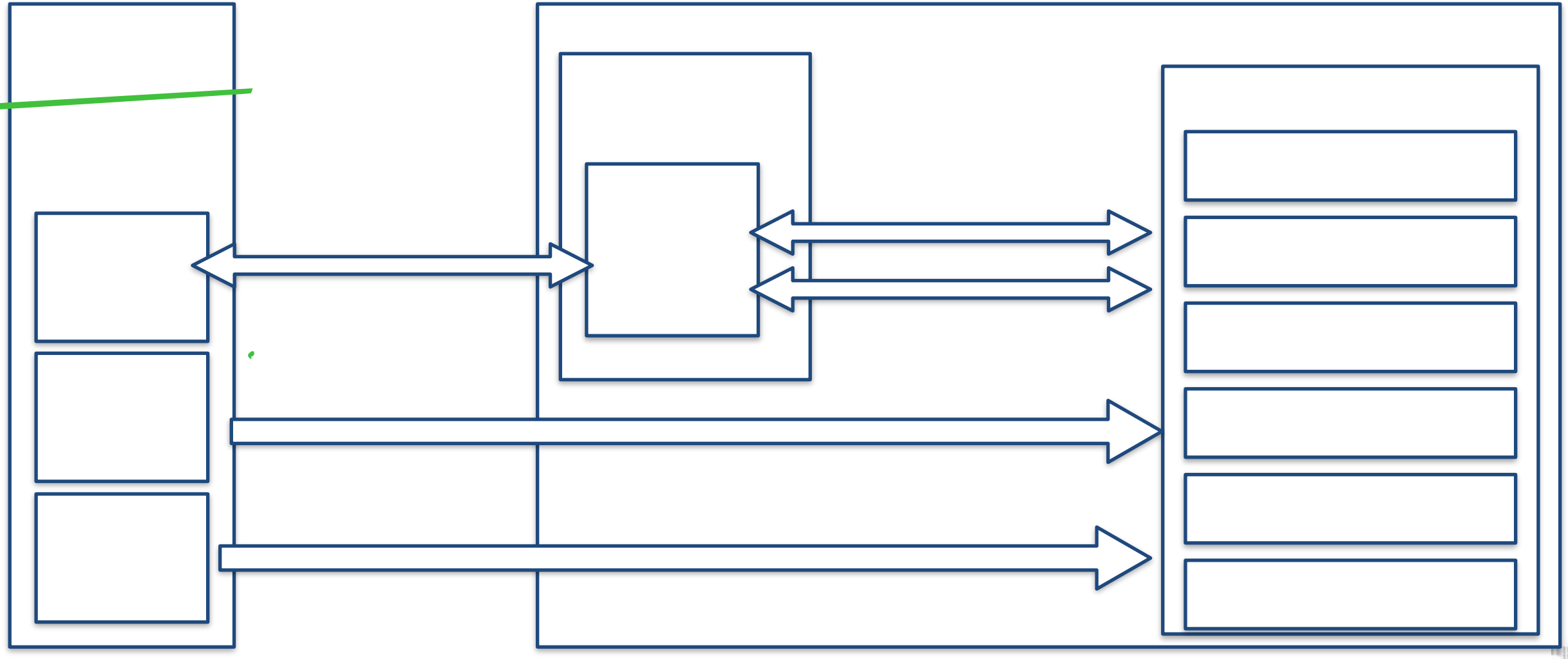
© 2017 IBM Corporation

Chapters 1-12



© 2017 IBM Corporation

Chapter 13



© 2017 IBM Corporation

Where do we make changes?
==> All of this is documented in the README.md file in Chapter 13

network information

fabric information:

Web Sockets

Bluemix application start-up

© 2017 IBM Corporation

Network Information

- Docker and the Kubernetes deploy use different names for the CA and for the channel
  - Docker
    - CA name: ca.example.org
    - channel name: composer
  - Kubernetes
    - CA name: CA
    - channel name: channel1

© 2017 IBM Corporation

HyperLedger Fabric information

- HyperLedger fabric always uses ~/.hfc-key-store to hold keys for direct access to the fabric.
  - This means that this folder must also exist in the IBM Cloud deployment
  - And must have keys made just for this deployment.
  - That requires creating the PeerAdmin information and storing it for later load to ~/.hfc-key-store
  - and then creating an exec to create that folder during deploy
  - the same is required for PeerAdmin.card and admin.card, which have to be stored in appropriate sub-folders in ~/.composer

© 2017 IBM Corporation

Web Sockets

- the web socket implementation was brute force for Chapters 1-12. In IBM Cloud, this approach does not work, as we cannot specify ports in the app.
  - --> Everything in the app must run through a single port.
  - This requires that the port creation be done in index.js while setting up http server and then shared with other modules in app.
  - app.locals provides that sharing ability
- Because multiple browsers can now attach to the app, must manage multiple client connect requests and keep them active until device departs.
- host_address for web socket connection from browser is no longer 'localhost:xxx', it is the URL for the bluemix app and needs to be dynamically acquired once the web app starts.

© 2017 IBM Corporation

IBM Cloud application start-up

- Up until this point, any application we have loaded into bluemix has been started with a simple 'node index' command.
  - --> When an application starts in Bluemix which will interact with Hyperledger and Composer, we need to set up two folders in the IBM Cloud for our app before it can run. Those folders are:
    - ~/.hfc-key-store
      - This folder will store the credential information necessary to gather blockchain events
    - ~/.composer/cards
      - This folder initially has the PeerAdmin and admin cards
    - ~/.composer/client-data
      - This folder initially has the credentials for the PeerAdmin and admin cards

© 2017 IBM Corporation

Code Changes: Server

- **index.js**:
  - The web socket creation and management code is removed from Z2B_Services and implemented in index.js, as it must share the httpserver service. It is also necessary to implement better socket management, since this will now run with access for multiple concurrent browsers. Because we are managing communications with multiple browsers, we need a way to send messages to all of them, which is done through the clients array. We also need to remove clients from that array when the browser session ends. This is done through the on-close process using information provided to the application in the client browser url & port.
  - Because we want to use the message sending ability from any module in our application, we extend the support to use app.locals to share the processMessages function.
- **Z2B_admin.js**
  - Administrative services are disabled so that the network is not accidentally taken off line.
- **Z2B_Services.js**
  - Routines for creating and managing sockets are removed completely and replaced with a single, very short, function
- **autoLoad.js**
  - During member creation, profile information needs to be loaded so that the member is able to connect with the correct network. This data is now loaded from a new connections.json file which is generated during the new kubernetes-deploy.sh script execution.
- **queryBlockchain.js**
  - getChainInfo is updated to pull the address of your kubernetes cluster and use that during the connection process.
- **hlcClient.js**
  - function calls which previously included the web socket to be used now reference req.app.locals, which is how we get to the processMessages function.

© 2017 IBM Corporation

Code Changes: Browser

- **z2b_events.js**
  - Earlier chapters in the tutorial had a web socket connection being created for each of the 6 roles (admin + 5 participant types). In this cloud-compatible version, we only create a single socket and then parse out the content to each participant based on the inbound message type, of which there are three (Message, Alert, BlockChain):
- **z2b_buyer.js** (and other participants)
  - all 'port' information is removed from the code as it is no longer relevant
  - calls to wsDisplay are removed as that service is now replaced by the wscConnect function

© 2017 IBM Corporation

Script Changes

- buildAndDeploy replaced by kuebernetes-deploy.sh
  - kubernetes-deploy uses set up information from the [ibm-container-service](https://github.com/IBM-Blockchain/ibm-container-service) git repo. This chapter includes the scripts from the cs-offerings folder in that repo.
- There are a number of functions which are used in the kubernetes-deploy.sh script built for the Zero To Blockchain tutorial:
  - ./createArchive.sh -n $NETWORK_NAME (this is the same version as before)
  - getContext (discovers your kube cluster ip address and sets the environment info)
  - clearOldCards (gets rid of now obsolete identity cards)
  - setupCluster (sets up the kube cluster - this is the code from ibm-containers-service)
  - pauseForCard (wait for you to access the automatically-loaded playground, launch it and then make a local copy of your PeerAdmin card)
  - updateCard (update the PeerAdmin card with the correct ip address)
  - ./getPEM.sh (credential management)
  - installNetwork (using the PeerAdmin card, install your network into your new kube cluster)
  - start-up command in package.json was node index and is replaced with ./init.sh

© 2017 IBM Corporation

Let’s go look at the code

© 2017 IBM Corporation

Putting it all together

- first create a free cluster in your organization
- install the bluemix cli and kubectl **./install-bx-cli.sh**
- make sure that you've completed the code updates in this chapter, then run:
  - **./kubernetes-deploy.sh -k name-of-your-cluster**
  - This will build your archive, populate your kubernetes cluster with hyperledger fabric V1 and hyperledger composer v0.16. It will then start the composer playground so that you can download the PeerAdmin card. The script will update that card with the correct address for your kubernetes cluster and then deploy your network.
- log in to bluemix using the bx login command
  - **bx login —apikey @{path-to-your-apikey-file}**
    - (creating an api key for IBM Cloud: https://console.bluemix.net/docs/iam/apikeys.html#manapikey)
  - update your manifest.yml file with an appropriate name for your nodejs app. The route must be globally unique. I am using, for example, Z2B.mybluemix.net, so no one else can use that route. The part you get to control is everything to the left of `.mybluemix.net`
  - **bx target —cf** to point cloud foundry to the correct organization and space
  - **bx cf push** to upload your application to the IBM Cloud

© 2017 IBM Corporation

The Plan: 30 minute Chapters with an hour or two of practice

Chapter 1:	What is Blockchain? Concept and Architecture overview
Chapter 2:	What's the story we're going to build
Chapter 2.1:	Architecture for the Story
Chapter 3:	Set up local HyperLedger V1 development environment
Chapter 4:	Build and test the network
Chapter 5:	Administration User Experience
Chapter 6:	Buyer Support and User Experience
Chapter 7:	Seller Support and User Experience
Chapter 8:	Shipper Support and User Experience
Chapter 9:	Provider Support and User Experience
Chapter 10:	Finance Company Support and User Experience
Chapter 11:	Combining for Demonstration
Chapter 12:	Events and Automating for Demonstration
Chapter 13:	Running your Blockchain on the IBM Cloud Free Cluster
Chapter 14:	Debugging in Docker

© 2017 IBM Corporation