Prerequisites

For this block chain, I'll assume that you have the following prerequisites:

* You've heard of blockchain or the Hyperledger Fabric framework
* You have intermediate Java programming knowledge and experience with both the language and platform
* You are familiar with or (ideally) proficient in using:
  + Eclipse IDE
  + Docker and Docker Compose
  + Gradle
  + Linux command line
  + SoapUI or another HTTP client software, such as Postman

You should also be comfortable installing software on your computer with minimal guidance. In the interest of space, I'm not including detailed installation instructions for all the software you need; the websites where you can obtain the software provide installation instructions.

Blockchain basics

A lot of hype surrounds blockchain, but it's well deserved. Not only is the technology itself cool, it's disruptive and has the potential to revolutionize the way business is done on the Internet.

Recommended content for blockchain developers

Sharpen your skills with developerWorks tutorials, courses, blogs, and community support in the [Blockchain Developer Center](http://developer.ibm.com/blockchain).

How? Well, let's think about the fundamental attributes of a successful business deal:

* Trust: We have a deal, but can I really trust you to honor it (or you me)?
* Transparency: Allows a look "behind the curtain" (which can both build trust and lessen the need for it).
* Accountability: Defines criteria to determine whether or not all parties agree that the deal has been honored.

The health of any business relationship between two or more parties will mean varying levels of the three attributes above (for example, more trust means less transparency required, and vice versa), but some of all of them must be present, or there will be problems.

Set up your development environment

In order to run chaincode, you first need to set up your development environment.

Once you've completed this section, you'll be ready to run one of the Hyperledger Java chaincode examples, where you'll deploy and invoke transactions on real chaincode. After that, I'll show you how to write a new chaincode program from scratch (well, almost).

In this section you will:

1. Set up the network environment — to run your local blockchain network.
2. Install the build software — to build your chaincode.
3. Install an HTTP client — to invoke transactions on your chaincode.
4. Start the blockchain network.
5. Build the Java shim client JAR.

### Set up the network environment

#### Install Docker

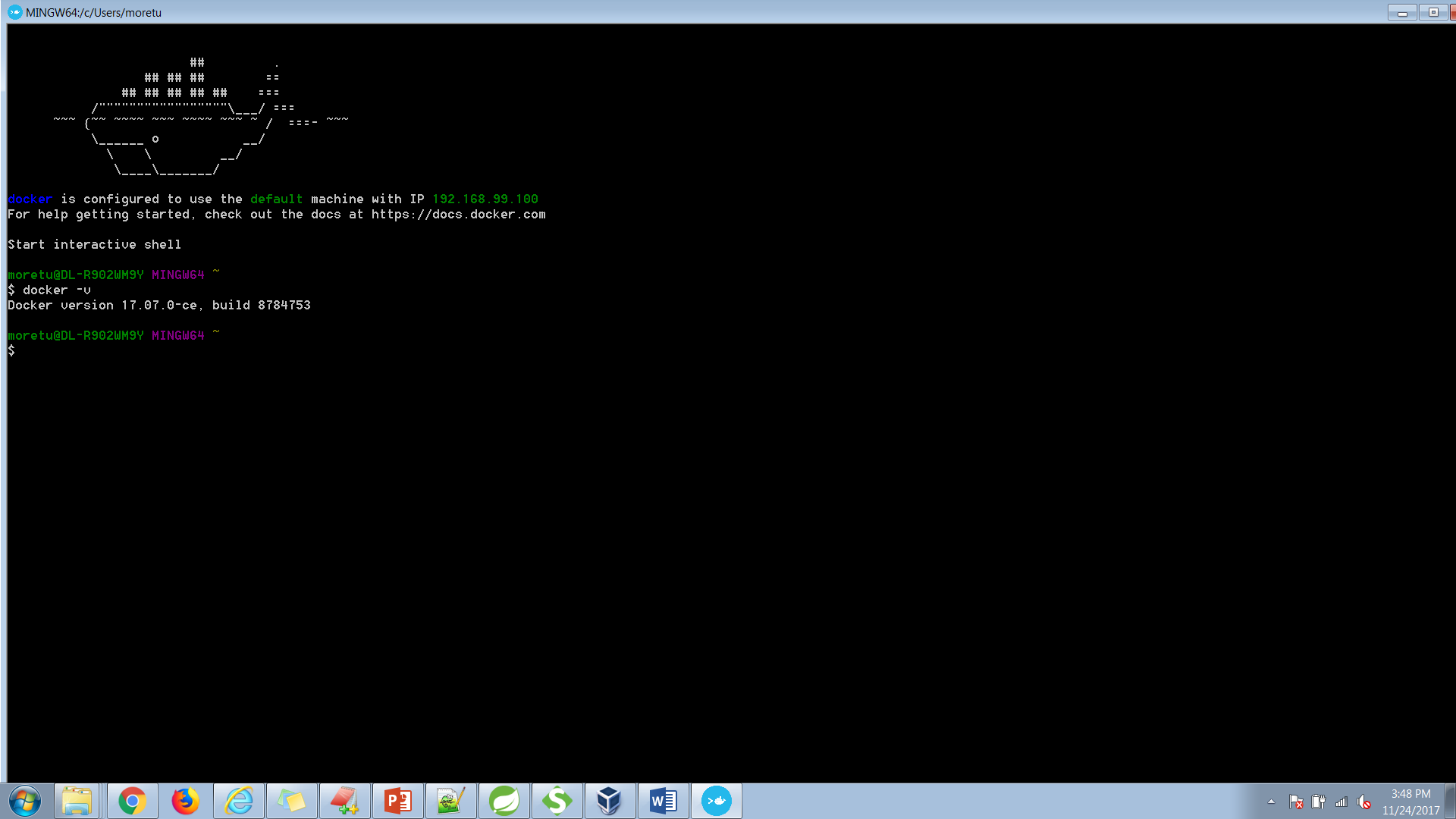
Installation instructions for Mac, Windows, and Linux can be found here:

<https://docs.docker.com/engine/getstarted/step_one>

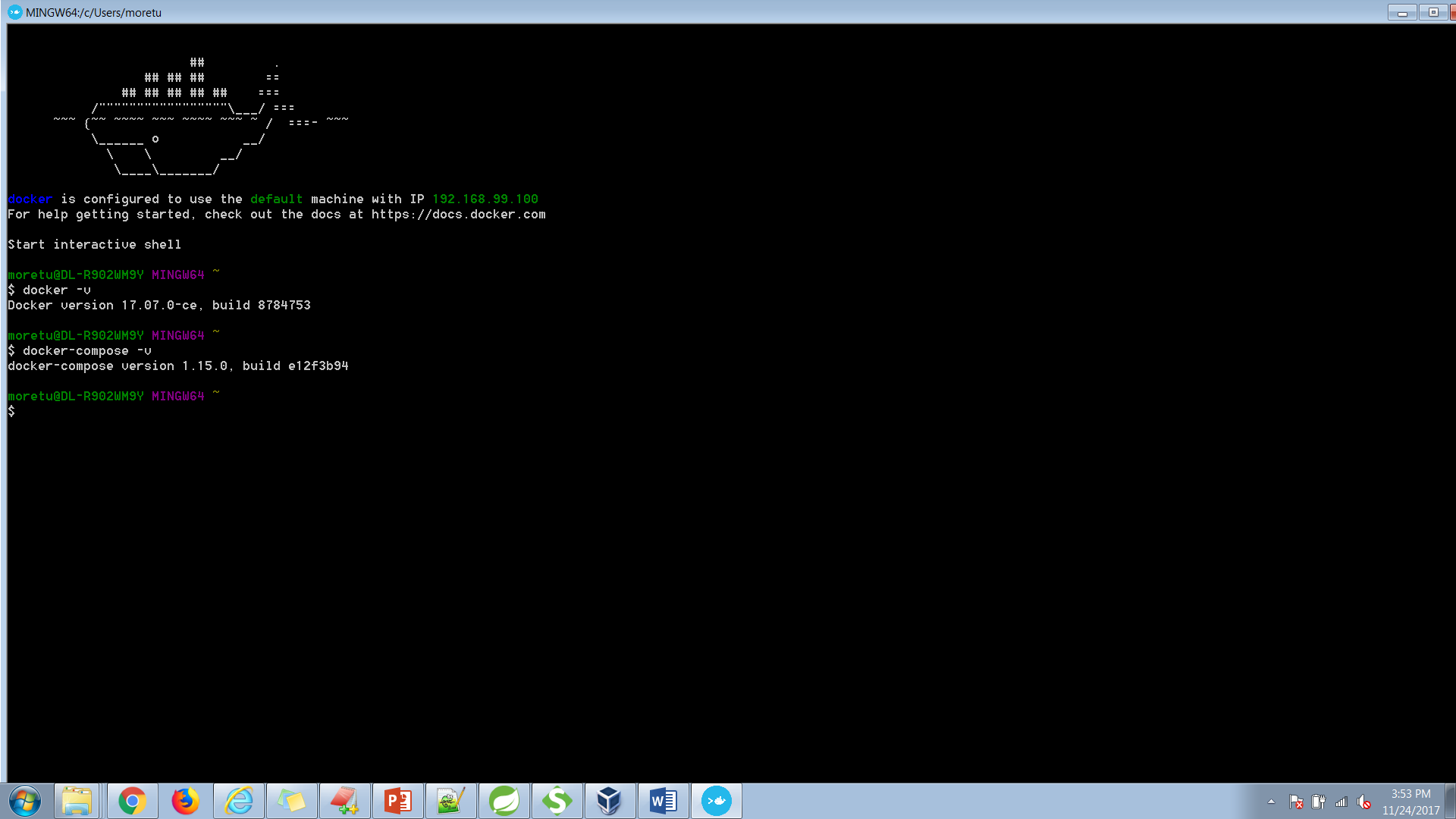
#### Verify Docker installation

To test your Docker installation, open a terminal window (or command prompt on Windows) and type the following commands:

docker -v



docker-compose -v



If you want to see Docker in action, you can run the hello-world image like this:

|  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- |
| $ docker run hello-world   Install the build software For the build system, the Hyperledger Fabric uses Gradle, and you will use that for this. Gradle is a build automation system that combines simple syntax to specify build components, along with the best features of Apache Ant and Apache Maven to create a powerful build system that's easy to use. It's no wonder so many developers are switching their projects over to Gradle. Install Gradle To install Gradle, follow the instructions here:  <https://gradle.org/install> Verify Gradle installation To verify your Gradle installation, open a terminal window and execute this command:   |  |  |  |  |  |  |  | | --- | --- | --- | --- | --- | --- | --- | | gradle -v   Install SoapUI To install SoapUI, follow the instructions here:  <https://www.soapui.org/getting-started/installing-soapui.html> Start the blockchain network Now that you have installed the necessary software to develop and test your chaincode, it's time to start your local blockchain network. The first step is to define the network's configuration.  First, create a directory that will serve as the root of all of the source code you'll use for chaincode development. In this tutorial I'll use ~/home/mychaincode (or C:\home\chaincode on Windows).  Next, set the GOPATH environment variable to this path. We won't be compiling any Go code, nor building Golang packages or other binaries, but Golang terminology is baked into Hyperledger, so it's a good idea to get comfortable with thinking in terms of the Go language and GOPATH.  On Linux, you would execute this command:   |  |  |  |  |  |  | | --- | --- | --- | --- | --- | --- | | export GOPATH=/c/IAD/mychaincode  you must tell Docker Compose how to compose and run the blockchain peer network. The definition of the network is in [YAML](http://www.yaml.org/), and you should name it docker-compose.yml.  membersrvc:  image: hyperledger/fabric-membersrvc  ports:  - "7054:7054"  command: membersrvc  vp0:  image: hyperledger/fabric-peer:x86\_64-0.6.0-preview  ports:  - "7050:7050"  - "7051:7051"  - "7053:7053"  environment:  - CORE\_PEER\_ADDRESSAUTODETECT=true  - CORE\_VM\_ENDPOINT=unix:///var/run/docker.sock  - CORE\_LOGGING\_LEVEL=DEBUG  - CORE\_PEER\_ID=vp0  - CORE\_PEER\_PKI\_ECA\_PADDR=membersrvc:7054  - CORE\_PEER\_PKI\_TCA\_PADDR=membersrvc:7054  - CORE\_PEER\_PKI\_TLSCA\_PADDR=membersrvc:7054  - CORE\_SECURITY\_ENABLED=false  - CORE\_SECURITY\_ENROLLID=test\_vp0  - CORE\_SECURITY\_ENROLLSECRET=MwYpmSRjupbT  links:  - membersrvc  command: sh -c "sleep 5; peer node start --peer-chaincodedev"  vp1:  image: hyperledger/fabric-peer:x86\_64-0.6.0-preview  ports:  - "8050:7050"  - "8051:7051"  - "8053:7053"  environment:  - CORE\_PEER\_ADDRESSAUTODETECT=true  - CORE\_VM\_ENDPOINT=unix:///var/run/docker.sock  - CORE\_LOGGING\_LEVEL=DEBUG  - CORE\_PEER\_ID=vp1  - CORE\_PEER\_PKI\_ECA\_PADDR=membersrvc:7054  - CORE\_PEER\_PKI\_TCA\_PADDR=membersrvc:7054  - CORE\_PEER\_PKI\_TLSCA\_PADDR=membersrvc:7054  - CORE\_SECURITY\_ENABLED=false  - CORE\_SECURITY\_ENROLLID=test\_vp1  - CORE\_SECURITY\_ENROLLSECRET=MwYpmSRjupbT  - CORE\_PEER\_DISCOVERY\_ROOTNODE=vp0:7051  links:  - membersrvc  - vp0  command: sh -c "sleep 5; peer node start --peer-chaincodedev"  notice that the CORE\_SECURITY\_ENABLED value is false. This means the fabric will not require you to send any type of end-user credentials  Now that the blockchain network definition is in place, you're ready to start your local blockchain network. To do this, run Docker Compose.  Navigate to your $GOPATH and execute this command:   |  |  |  |  |  | | --- | --- | --- | --- | --- | | docker-compose up   Build the Java shim client JAR Before you can run the chaincode examples, you need to get the latest source code from Hyperledger's GitHub repository.  Execute this command to create the directory structure the fabric build scripts expect:   |  | | --- | | mkdir -p $GOPATH/src/github.com/hyperledger |   Next, navigate to the bottom of the new directory structure you created:   |  | | --- | | cd $GOPATH/src/github.com/hyperledger |   There are two ways to access the Hyperledger source.   * **Without** git:   Navigate to the [Hyperledger GitHub mirror](https://github.com/hyperledger/fabric) and click on the **Clone or download** button, then **Download ZIP** . A ZIP file called fabric-master.zip will be downloaded to your computer,  which you can extract to $GOPATH/src/github.com/hyperledger. **Note:** make sure to change the name of the root directory from **fabric-master** to **fabric** when you extract the file.   * **With** git:   Navigate to $GOPATH/src/github.com/hyperledger, copy the URL from the text field in the "Clone with HTTPS" box , and execute this command using the copied URL:   |  |  | | --- | --- | | git clone [https://github.com/hyperledger/fabric.git -b v0.6](https://github.com/hyperledger/fabric.git%20-b%20v0.6)  Now you're ready to build the Java chaincode shim client JAR. Navigate to$GOPATH/src/github.com/hyperledger/fabric/core/chaincode/shim/java and run these two commands:   |  | | --- | | gradle -b build.gradle clean  gradle -b build.gradle build |   The Gradle build output should look like this:  $ cd $GOPATH/src/github.com/hyperledger/fabric/core/chaincode/shim/java  $ **gradle -b build.gradle clean**  Starting a Gradle Daemon (subsequent builds will be faster)  :core:chaincode:shim:java:clean  BUILD SUCCESSFUL  Total time: 5.422 secs  $ **gradle -b build.gradle build**  :core:chaincode:shim:java:copyProtos UP-TO-DATE  :core:chaincode:shim:java:extractIncludeProto  :core:chaincode:shim:java:extractProto UP-TO-DATE  :core:chaincode:shim:java:generateProto UP-TO-DATE  :core:chaincode:shim:java:compileJava  :core:chaincode:shim:java:processResources  :core:chaincode:shim:java:classes  :core:chaincode:shim:java:jar  :core:chaincode:shim:java:assemble  :core:chaincode:shim:java:extractIncludeTestProto  :core:chaincode:shim:java:extractTestProto UP-TO-DATE  :core:chaincode:shim:java:generateTestProto UP-TO-DATE  :core:chaincode:shim:java:compileTestJava UP-TO-DATE  :core:chaincode:shim:java:processTestResources UP-TO-DATE  :core:chaincode:shim:java:testClasses UP-TO-DATE  :core:chaincode:shim:java:test UP-TO-DATE  :core:chaincode:shim:java:check UP-TO-DATE  :core:chaincode:shim:java:build  :core:chaincode:shim:java:copyToLib  :core:chaincode:shim:java:generatePomFileForMavenJavaPublication  :core:chaincode:shim:java:publishMavenJavaPublicationToMavenLocal  :core:chaincode:shim:java:publishToMavenLocal    BUILD SUCCESSFUL  Total time: 4.521 secs Deploy and run a Java chaincode exampleClone the ChaincodeTutorial project from GitHub you have your Eclipse IDE configured to work with Gradle, you will clone the ChaincodeTutorial code from GitHub and import the code into Eclipse. Open a command prompt or terminal window, navigate to your $GOPATH, and execute this command:  git clone <https://github.com/makotogo/ChaincodeTutorial.git>  Your command output should look something like this:  $ export GOPATH=/c/IAD/mychaincode  $ cd $GOPATH  $ git clone https://github.com/makotogo/ChaincodeTutorial.git  Cloning into 'ChaincodeTutorial'...  remote: Counting objects: 133, done.  remote: Compressing objects: 100% (90/90), done.  remote: Total 133 (delta 16), reused 118 (delta 1), pack-reused 0  Receiving objects: 100% (133/133), 9.39 MiB | 1.95 MiB/s, done.  Resolving deltas: 100% (16/16), done.  $ cd ChaincodeTutorial  $ pwd  /c/IAD/mychaincode/ChaincodeTutorial Import the project into Eclipse In Eclipse, go to **File > Import... > Gradle > Existing Gradle** project. A wizard dialog box should open   **Build the Java chaincode**  Deploy and run your Java chaincode You will need to extract the build/distributions/ChaincodeTutorial.zip file and run the chaincode script      Now your Java chaincode is registered with your local blockchain network, and you're ready to deploy and test your chaincode   **Deploy your Java chaincode** Open SoapUI. If you are comfortable doing so, you can create a new REST project      Now that your Java chaincode is deployed and initialized, you can invoke transactions on it. In this section, you will invoke the log and query functions as transactions.    Pushing data from different Peer-    Output Log:-    \*\*IMP:-  Up get second peer up and running you need to run below docker command  docker run --rm -it -e CORE\_VM\_ENDPOINT=unix:///var/run/docker.sock -e CORE\_PEER\_ID=vp1 -e CORE\_PEER\_ADDRESSAUTODETECT=true -e CORE\_PEER\_DISCOVERY\_ROOTNODE=192.168.99.100:7051 hyperledger/fabric-peer:x86\_64-0.6.0-preview peer node start | | | | |
|  |