Instructions on how to deploy the network

1. Inorder to successfully deploy the business network the “teardownFabri.sh” script was excuted followed by the command “rm – Rf ~/.composer/” to clear any business network that was already being run and delete any identities on the network, respectively. The command “./startFabir.sh” was used to start up the network.As shown below in Figure 1.

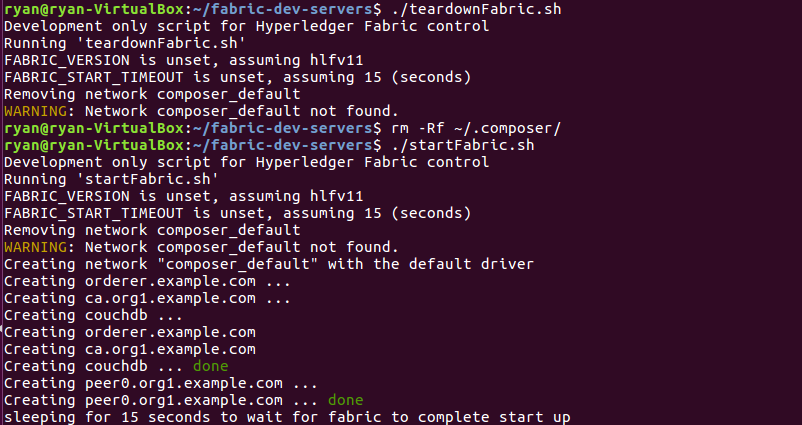


Figure 1: Resetting the network

1. The network that you want to deploy must be inside the “~/fabric-dev-servers” directory and must be extracted into a directory from that point of your choosing as shown in Figure 2.



Figure 2: directory setup

1. Once inside of the directory were you extracted the file execute the command “composer archive create –t dir –n .” to create a “.bna” file as shown in Figure 3.

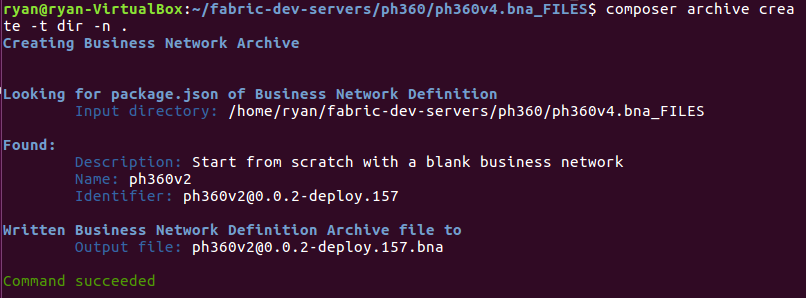


Figure 3: Creating a “.bna” file

1. Go back to the “fabric-dev-servers” directory and create a Peer Admin Card using the command “./createPeerAdminCard.sh” as shown in Figure 4.

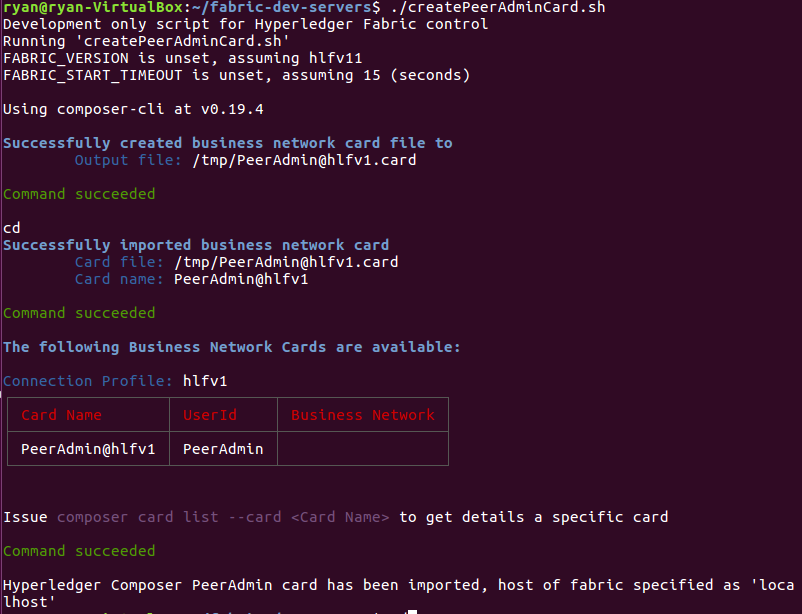


Figure 4: Creating a Peer Admin Card

1. Now a network card must be installed and the network need to be started as shown in Figure 5 back in the network directory.

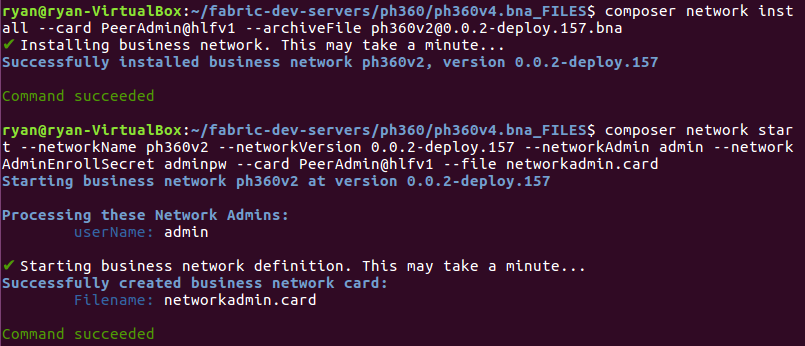


Figure 5: Install Admin Card and starting network

1. The Admin card needs to be imported and the network should be pinged to make sure it is running smoothly as shown in Figure 6.

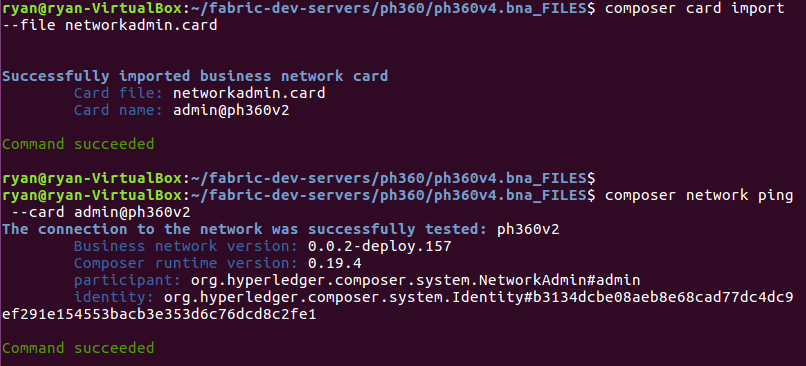


Figure 6: Card Import and Network Ping

1. Participant need to be added and identities now need to be issued to give users access to the block chain. An example of adding a Customer to a registry as well as issuing an identity is shown in Figure 7.

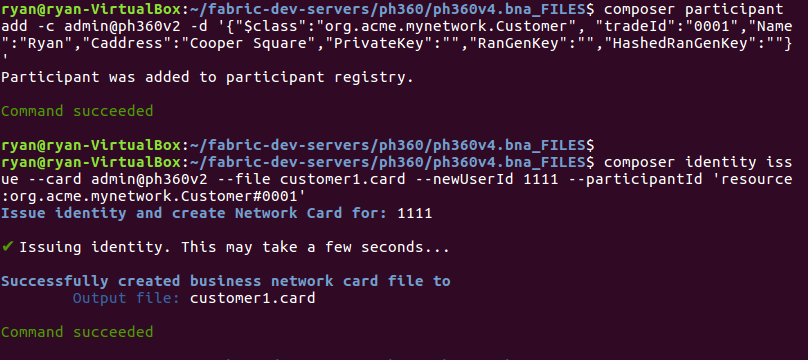
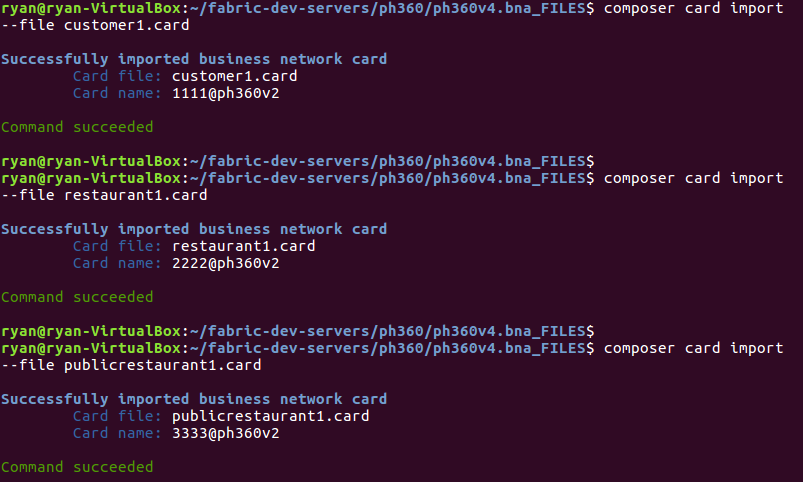


Figure 7: Adding Participant and Issuing Identities

1. For each participant that was added to a registry the corresponding card must be imported as shown in Figure 8.



1. To view the participants that were added a REST API can be generated by opening another terminal in the same directory an entering the input shown in Figure 9 with an appropriate output shown in Figure 10. The REST API will run on local port 3000.

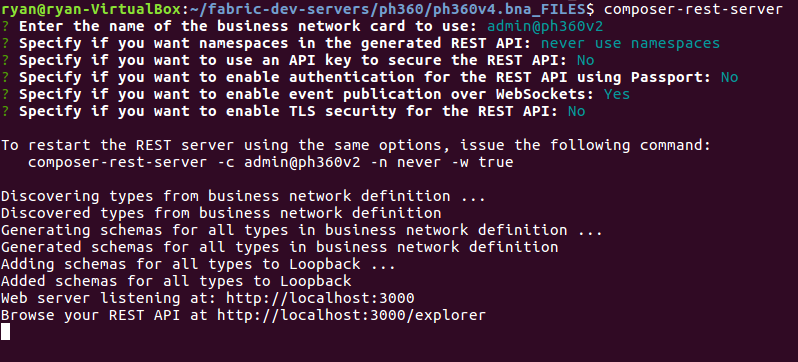


Figure 9: Generating a REST API

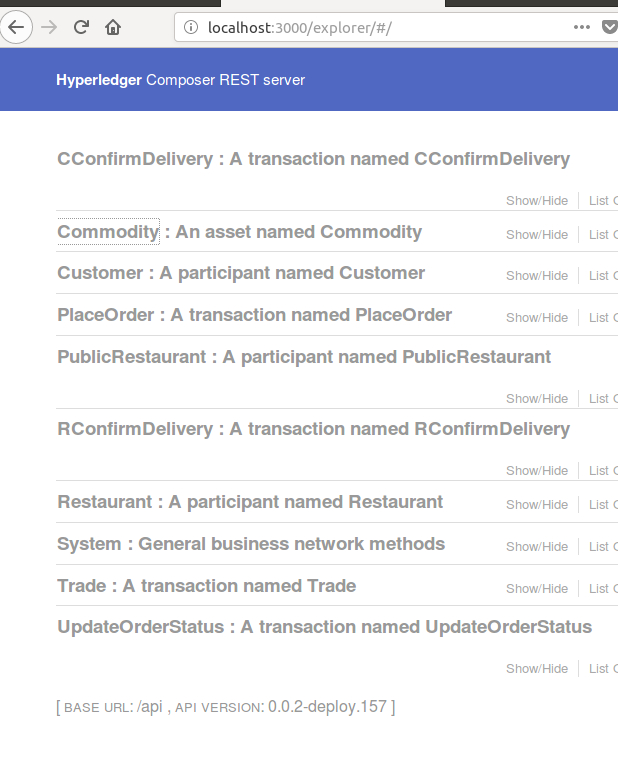


Figure 10: REST API Server

1. In order have a successful transaction between two parties the following input needs to occur as shown in Figure 11. In Figure 11 only the “Place Order” and “UpdateOrderStatus transaction are submitted. This generate key and hashes for the participant involved which can be seen by accessing the “get” function in the REST API. This also generate a commodity asset as shown in Figure 12.

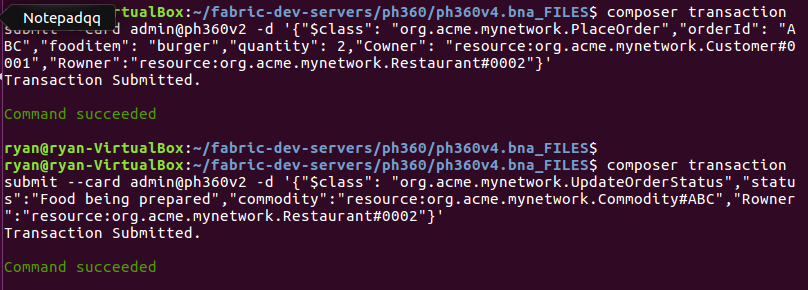


Figure 11: PlaceOrder and UpdateOrderStatus transactions

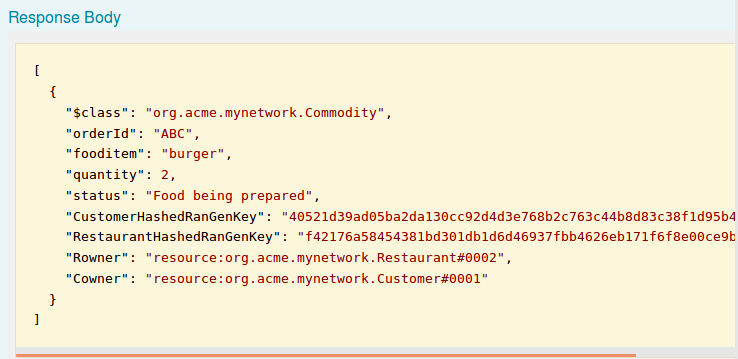


Figure 12: REST API output for a commodity

1. In order to submit the “CConfirmDelivery” and “RConfirmDelivery” the following input needs to occur. The customer and the restaurant need to give each other their respective keys so that the other can verify they are dealing with the correct party. This is the definition of a successful transaction. Appropriate output is shown in Figure 14.

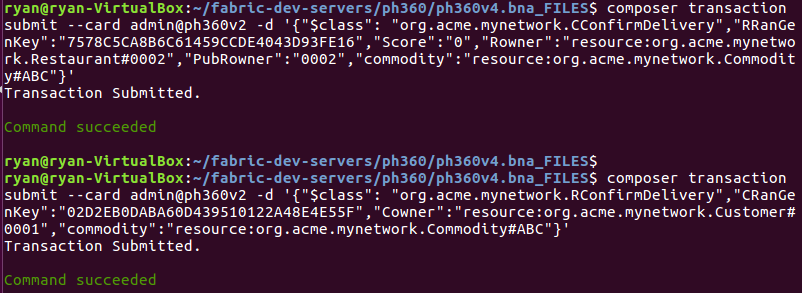


Figure 13: “CConfirmDelivery” and “RConfirmDelivery” transactions

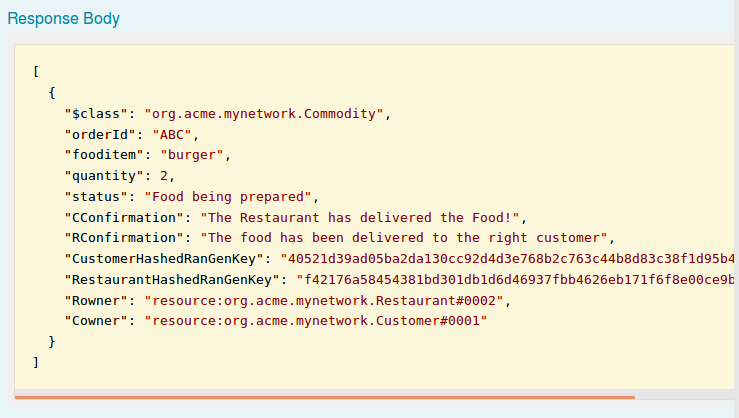


Figure 14: A successful transaction

1. To start angular one can run the command “yo hyperledger-composer:angular” as shown in Figure 15 and then go into the angular app-directory and run “npm start” . In a browser you can go to port <http://localhost:4200/> to view the angular app. An appropriate output for angular is shown in Figure 16.

Figure not shown

Figure 15: starting angular

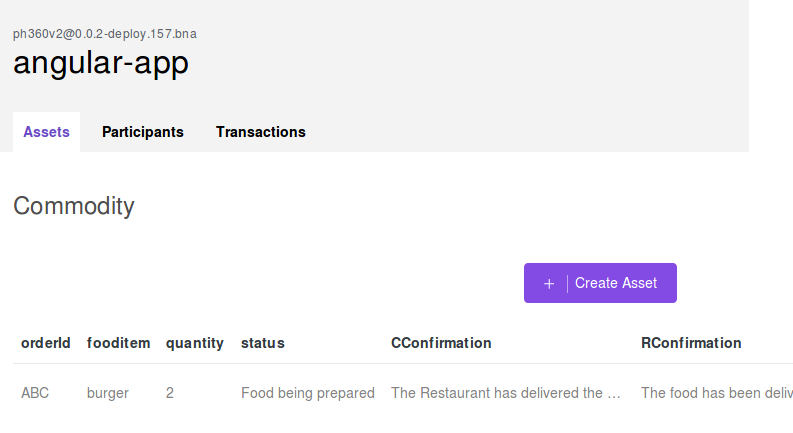


Figure 16: Angular Framework