A Beginner guide to Quorum Blockchain

Salman Dabbakuti

May 31 · 3 min read

So, What is Quorum?

1. Quorum is a permissioned implementation of Ethereum and includes fork of **go-ethereum** Developed by global financial services firm **JP Morgan Chase.**
2. Specifically Designed for private Transactions between nodes to ensure Privacy.
3. Quorum Uses **Raft** and **Istanbul** consensus Algorithms.
4. Quorum Uses **Tessara** and **Constellation** for encrypting and sharing data across nodes.
5. Quorum network doesn't cost ether for transactions. basically transaction verification is done by voting Consensus mechanism.
6. Quorum has similarities like ethereum. except, there is a parameter "**privateFor**". it makes Quorum really standout. this "privateFor" parameter plays vital role in doing private

transactions. every node has its own Privatekey-Publickey pair generated by quorum network manager. if we wanted particular node only to be part of the transaction, we will pass public key of participating node with privateFor parameter.

Getting Started With Quorum-examples

The best way of getting to know about Quorum is Running 7nodes example in their examples Repository. even there are several ways of setting up 7nodes, I prefer setting up with docker. because it seems to be the easiest method. For this, I'm using **Ubuntu 18.04** on **Google Cloud.** if you have Ubuntu machine on AWS EC2 or local Ubuntu machine, you can go with it. please make sure that docker and docker-compose are installed.

In the demonstration, we are going to setup 7 quorum nodes and see how Private transactions work between nodes.

Steps

1. **Setting Up 7nodes.**

```
git clone https://github.com/jpmorganchase/quorum-examples.git

cd quorum-examples

QUORUM_CONSENSUS=raft docker-compose up -d
```

By default, Quorum Network is created using Tessera transaction manager and Istanbul BFT consensus. so in order to change consensus configuration to Raft, we set environment variable **QUORUM_CONSENSUS=raft** and then started docker containers in the background with 7nodes running.

Run `docker ps` to check the status of each container and their IDs. wait till the status of Container become healthy.

2. Deploying Simple SmartContract

For Deploying smart contract on any node, we need to enter into geth javascript console of particular node. run below command to enter into console.

```
docker exec -it quorum-examples_node1_1 geth attach
/qdata/dd/geth.ipc
```

Here `quorum-examples_node1_1` is the container id of node1. above command opens up node1 geth console. now we can do operations from this console alone.

Now we are about Run javascript file that deploys simple smartcontract on node1. if you could observe in your examples repository, there are files `private-contract.js` and `simplestorage.sol` in `quorum-examples/examples/7nodes` directory. from your node1 geth console, Run

```
> loadScript('/examples/private-contract.js')
```

Be patient, it will take sometime. once deployed, it will return contract address. please copy it. it deploys a simple storage contract on node1 with a value of 42 and transaction is only private for node1 and node7(have a look at code). this means that, other nodes can't see this value.

To verify, go to geth javascript console of node2, Create contract instance with ABI and contract address.

```
>const abi=<Paste ABI here>;
>const address="<paste Contract Address here>";
>const contract=eth.contract(abi).at(address);
```

```
>contract.get() //calling get() method
0
```

Go to geth javascript console of node7, Create contract instance with ABI and contract address as we done for node2 above. now calling get() method on node7, returns a value of 42 which is added during deployment.

```
>contract.get()
42
```

3. Sending Private Transactions

Now you can create a new private transaction between node7 and any of your favorite node by setting new number with set() method and passing your node's public key. you can find public keys of all 7 nodes in `quorum-examples/examples/7nodes/keys/` directory.

```
contract.set(4,{from:eth.coinbase,privateFor:["<your
preferred node public key>"]});
```

Above script will set a value of 4 and only private for particular node. So, go to geth javascript console of your preferred node and initiate contract instance (if you didn't initiated previously) and call **get()** method. it will returns 4. do the same thing on other nodes too.

With quorum, we can ensure privacy and fast transactions across nodes. for further reading, please have a look at their official site and Github.