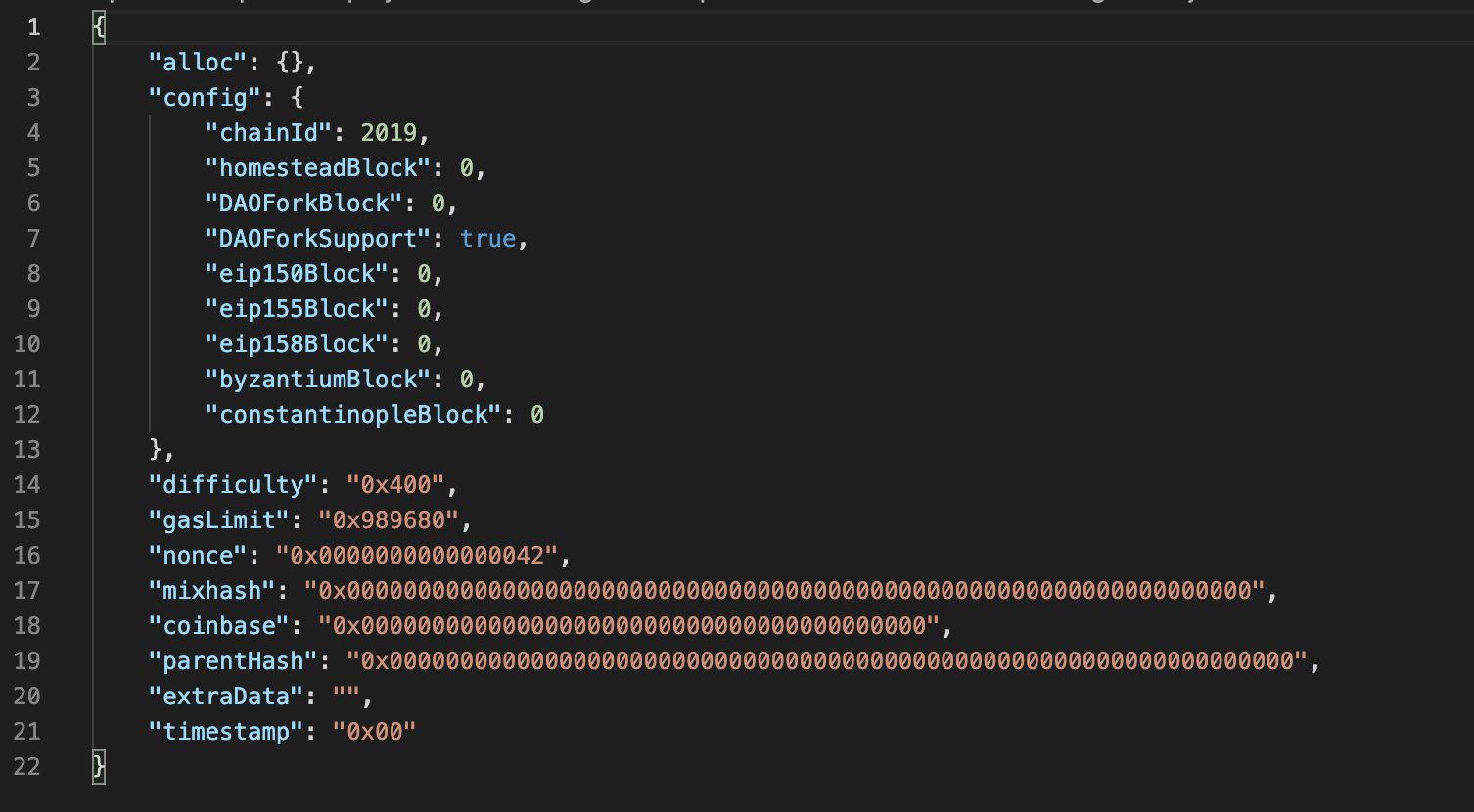
**Instructions for starting a private network**

* Start by creating a folder for the network, call it *ethereumNetwork*
* This folder must contain the *genesis.json* file.  
    
  
* In your terminal go to the *ethereumNetwork,* initialize the blockchain chain data folder with the following command -

| geth --datadir ./datadir init ./genesis.json |
| --- |

This will create a *datadir* folder inside the *ethereumNetwork* folder that will contain the chain data of the blockchain.

* To start the private blockchain network use the following command -

| geth --datadir ./datadir/ --networkid 2019 console |
| --- |

This will start up a console that exposes APIs that are available with this network.

* Then create a few accounts in this network using the following command -

| personal.newAccount('your password') |
| --- |

* The next step is to start the mining process. By default, the mining rewards will go to the first account that was created. To start mining -

| miner.start() |
| --- |

* To stop mining, use *miner.stop().* Press *ctrl + d*, to exit the network.

| miner.stop() |
| --- |

# **Deploying the contract**

* Go to the *KYC-SC* folder. This will have a truffle project. It will also contain the KYC.sol contract. This contract has to be compiled and deployed to the private Ethereum network. Before that ensure that the *truffle-config.js* file has the following geth network defined -  
    
  
* In a new terminal, go to the folder where you have defined the *ethereumNetwork* and type the following command -

| geth --datadir ./datadir --networkid 2019 --rpc --rpcport 30303 --allow-insecure-unlock console |
| --- |

This will restart the network and also expose the network APIs to an HTTP interface.

* Unlock the first account that was created earlier for an infinite amount of time using the command -

| personal.unlockAccount('Address of 1st Account', 'password of the account', 0) |
| --- |

* Start the mining process again.

| miner.start() |
| --- |

* In a separate terminal, go to the *KYC-SC*. Compile the smart contracts -

| truffle compile |
| --- |

* After a successful compilation deploy the contract to the geth network using the following command -

| truffle migrate --network geth |
| --- |

* Use the following command to open a console that can be used to call the KYC contract functions and interact with them -

| truffle console --network geth |
| --- |

* Get an instance of the deployed KYC contract -

| const instance = await KYC.deployed() |
| --- |

* Get the address of the deployed instance -

| instance.address |
| --- |

* Use the functions of the contract like *addBank()* and interact with the contract.

| instance.addBank("bank1", address of the bank, "reg1) |
| --- |

Other functions can be called in a similar fashion