A Project Report

On

Integration of Multiple Smart Contracts over Ethereum

BY

Surya Garikipati

2019AAPS1221H

Under the supervision of

Dr. Subhrakanta Panda

Soumya Prakash Otta

SUBMITTED IN PARTIAL FULLFILLMENT OF THE REQUIREMENTS OF

CS F366-67: LABORATORY PROJECT

BIRLA INSTITUTE OF TECHNOLOGY AND SCIENCE PILANI (RAJASTHAN)

HYDERABAD CAMPUS

(May 2022)

Birla Institute of Technology and Science-Pilani,

Hyderabad Campus

Certificate

This is to certify that the project report entitled “**Integration of multiple smart contracts over Ethereum**” submitted by Surya Garikipati (ID No. 2019AAPS1221H) in complete fulfillment of the requirements of the course CS F366-67, Laboratory Project Course, embodies the work done by him under my supervision and guidance.

Date: 04/05/2022

Dr. Subhrakanta Panda

Soumya Prakash Otta

BITS- Pilani, Hyderabad Campus

ABSTRACT

We research the way 2 different smart contracts on the same blockchain would interact with each other by integrating them together.

A smooth and efficient creation of smart contract 1 and 2 for Identity and Access Management. A test network which is a secure network to hold our database in – A unique id to identify each account along with face id.

We test various methods to figure out an efficient way for the integration of these two smart contracts.

ACKNOWLEDGMENTS

This project owes its existence to the support of innumerable people. Dr. Subhrakanta Panda and Soumya Prakash Otta Sir have been an ideal teacher, mentor and supervisor. Their advice and constant encouragement have been invaluable. I am grateful for their time and for allowing me the opportunity to work under them.

Many thanks also go out all my colleagues and mentors at BPHC who have contributed enormously to bring me where I am today.

CONTENTS

LIST OF FIGURES

Abbreviations

IAM – Identity and Access Management

SC – Smart Contract

Geth – Go-Ethereum software

JWT – JSON Web Token

Introduction

The objective is to develop a multifactor authentication approach for IAM solution for a Private Cloud using Smart Contract with a Private Blockchain. We propose to implement the same solution for a graduated federated/multi cloud environment.

The creation of smart contract 1 and 2 is completely done using solidity. We Implement these contracts by creating a private blockchain using Go-Ethereum. These contracts are then uploaded onto the test network, getting them ready to interact with each other using RFC 7519 Based Intercommunication with JWT.

Method and Challenges

The goal of this project is to use a Blockchain to build a typical IAM capability for a cloud by encrypting and storing user ID databases using a hash function and using an intelligent security engine to process it on a blockchain.

Smart Contracts are being used to securely access the user ID and associated authentication for cloud-based resources. Off the blockchain, a computationally demanding cryptographic policy enforcement mechanism is presented.

A separate smart contract, as well as its integration for multifactor authentication, will be used to combine comparable capabilities.

The identification attributes and access control restrictions will be saved and retrieved on the blockchain using smart contracts, while the encrypted data will be kept off the network.

We have already heard of creating smart contracts and deploying them on a blockchain. But there is a gray area in the implementation of two smart contracts together and implemented on one blockchain.

Analysis

We are to create two smart contracts. Each smart contract has its own features and quite different from each other. But the most difficult part of this project is the integration of both smart contracts.

Smart Contract 1 focuses on Authentication and Authorisation. The smart contract should be able to identify each unique user. This means no user can create two accounts and each user should have a secure private account. A face id is the second step of authentication which will allow users to access their account.

Smart Contract 2 focuses on Access Control. Its task is the management of all the cloud-based resources. The smart contract has access to the entire server and database. It tracks all resources and monitors all servers. It can delete or add servers where it deems necessary.

To integrate these two smart contracts on the same blockchain we use a software called RFC 7519 Based Intercommunication with JWT.

Literature Survey

Reference 1: This paper talks about the management of cloud databases using a smart contract on a blockchain

Reference 2: This paper brings light to the reason we are using smart contracts and how the play a major role on the blockchain

Reference 3: This paper is a survey that highlights the advantages of IAM on the blockchain.

Reference 4: This paper helps shows how to implement smart contract 2.

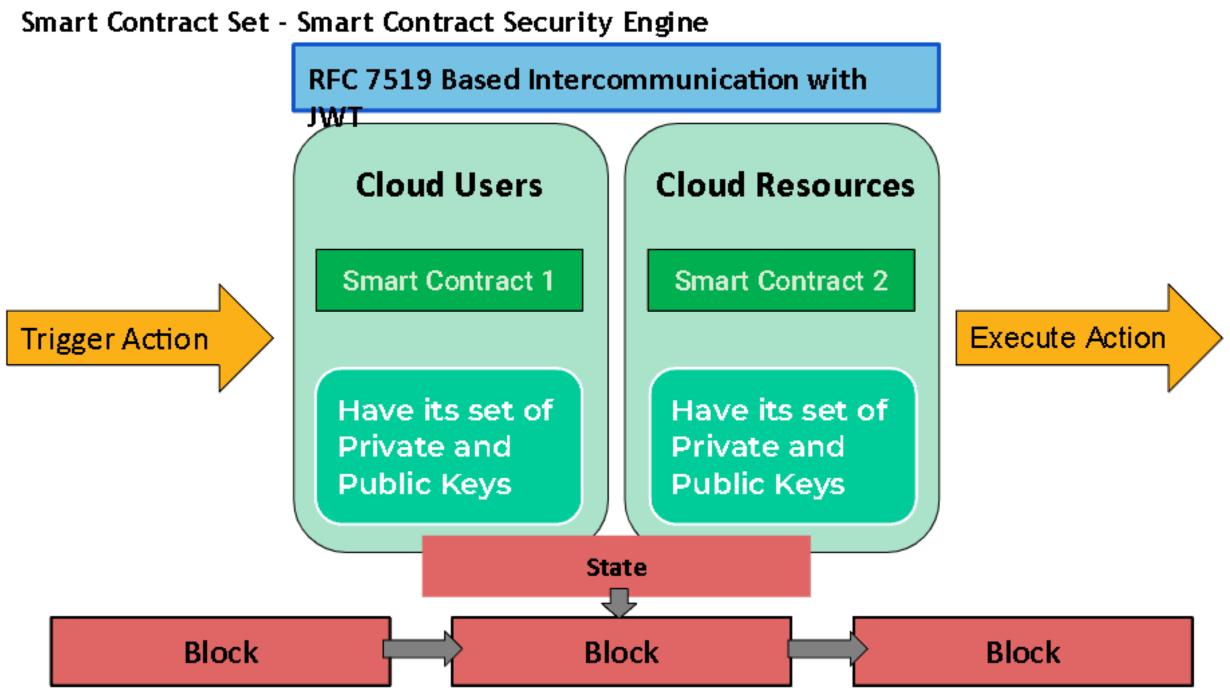
Reference 5: This paper talks about the various challenges and improvements that can be made on a cloud management smart contract

Reference 6: This paper talks about Smart Contracts in general and how we can integrate them onto our blockchains.

Reference 7: This paper sheds light on IAM for a smart contract. How to identify each user and make each user unique.

Plan of Action

Proposed Approach


Smart Contract Set - Smart Contract Security Engine
RFC 7519 Based Intercommunication with JWT
Cloud Users
Smart Contract 1
Have its set of Private and Public Keys
Cloud Resources
Smart Contract 2
Have its set of Private and Public Keys
Trigger Action
Execute Action
State
Block
Block
Block

Creation of Smart Contract 1 which manages the cloud-based users.

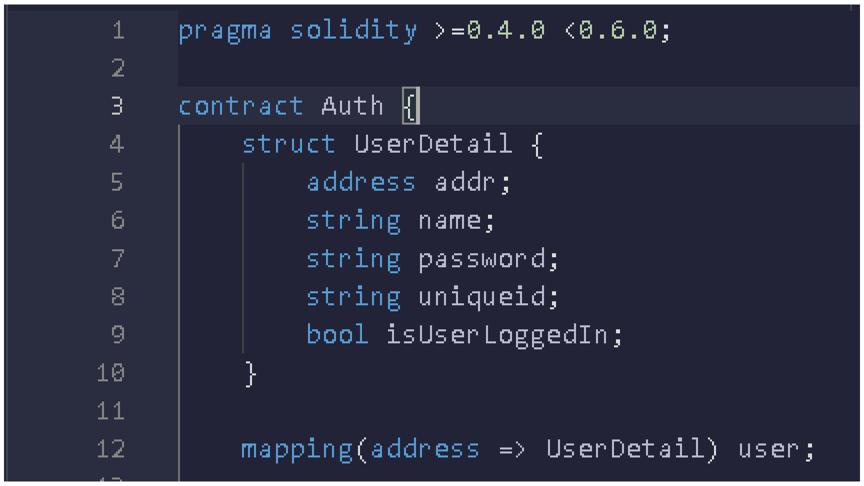
Creation of Smart Contract 2 which manages the cloud-based resources.

Run Smart Contract 1 and 2 on a single blockchain.

Create JSON Web Tokens and have the two smart contracts communicate with each other via JWT.

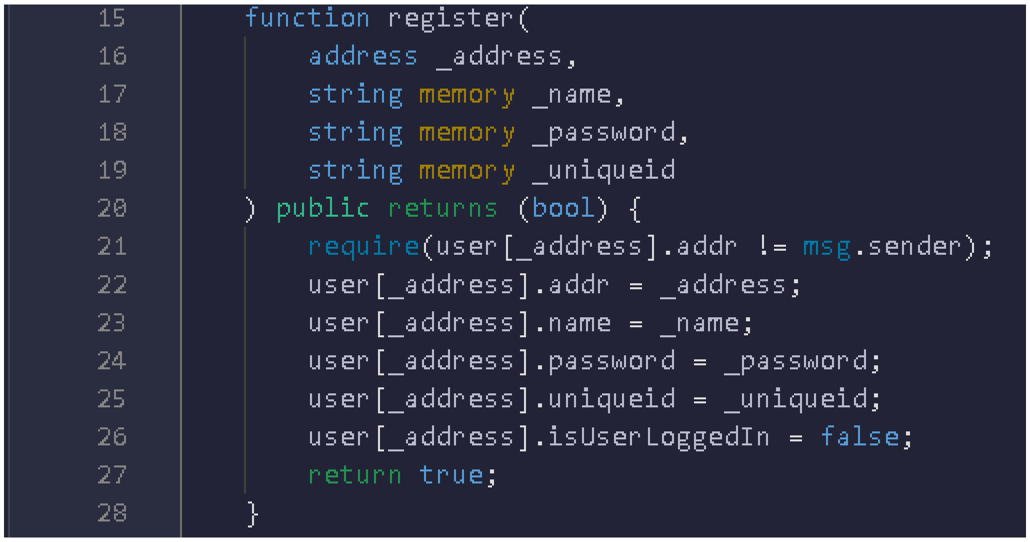
Implementation

Creation of smart contract 1



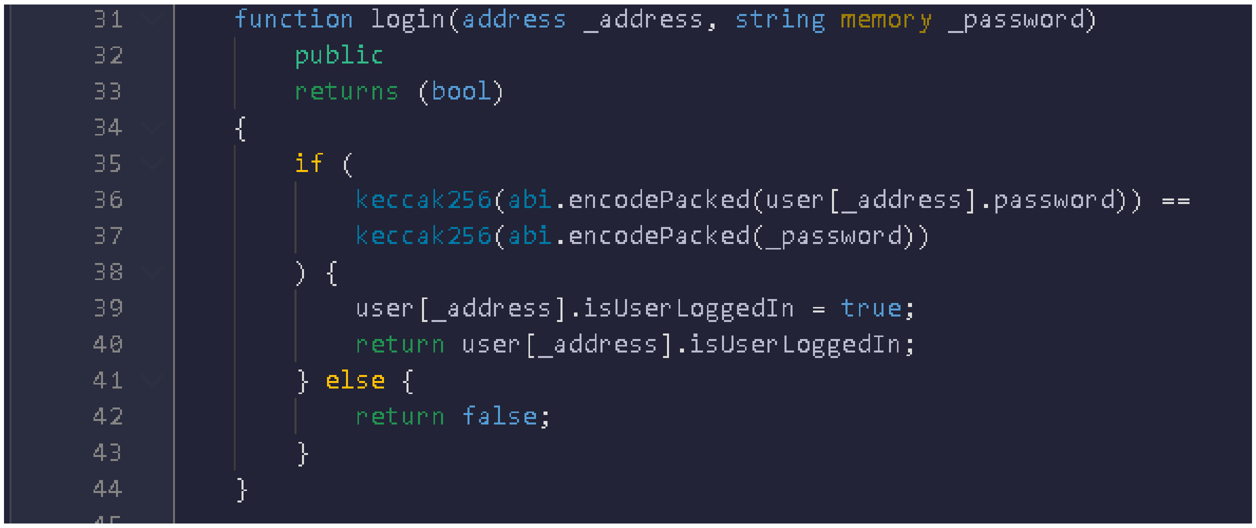
```
pragma solidity >=0.4.0 <0.6.0;

contract Auth {
    struct UserDetail {
        address addr;
        string name;
        string password;
        string uniqueid;
        bool isUserLoggedIn;
    }

    mapping(address => UserDetail) user;
```

We are using a version of solidity that can use any version between 0.4.0 and 0.6.0. We create a contract called Auth and declare each variable and map the address to user.



```
    function register(
        address _address,
        string memory _name,
        string memory _password,
        string memory _uniqueid
    ) public returns (bool) {
        require(user[_address].addr != msg.sender);
        user[_address].addr = _address;
        user[_address].name = _name;
        user[_address].password = _password;
        user[_address].uniqueid = _uniqueid;
        user[_address].isUserLoggedIn = false;
        return true;
    }
```

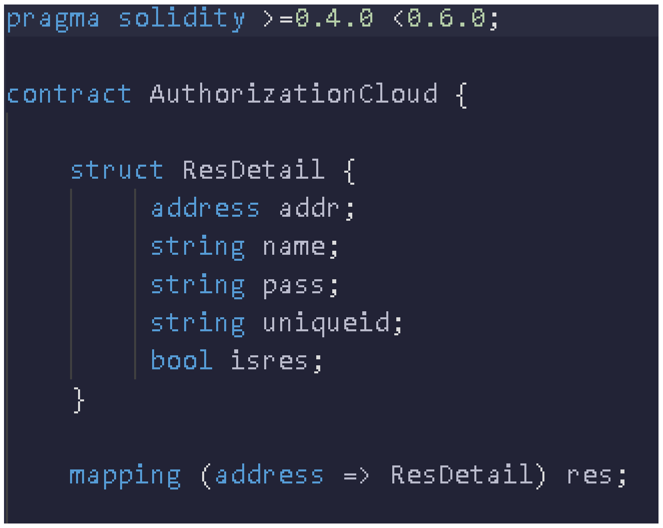
We create a function called register. It has 4 attributes. We need to specify an address, name of account, password for account, and unique id for each account.



```
function login(address _address, string memory _password)
    public
    returns (bool)
{
    if (
        keccak256(abi.encodePacked(user[_address].password)) ==
        keccak256(abi.encodePacked(_password))
    ) {
        user[_address].isUserLoggedIn = true;
        return user[_address].isUserLoggedIn;
    } else {
        return false;
    }
}
```

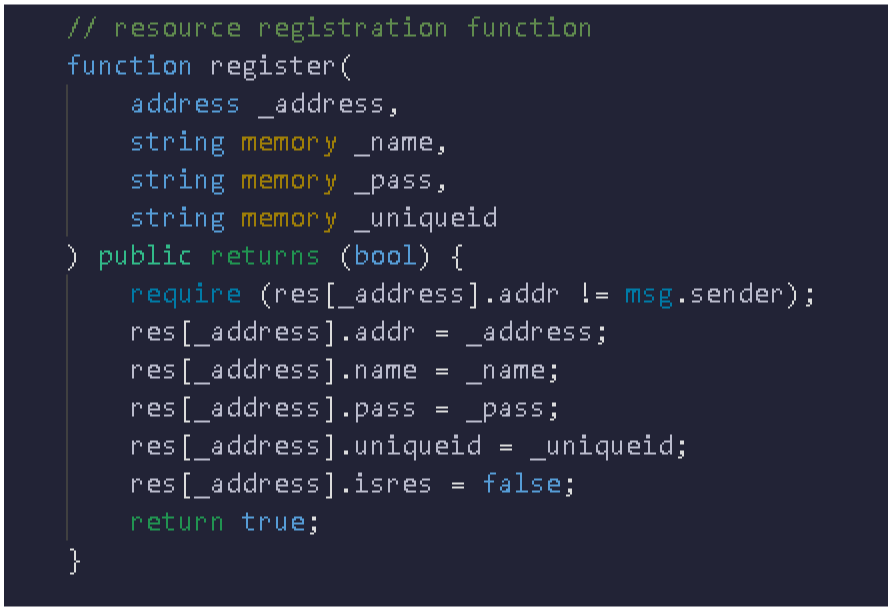
The function login. Takes the address and password as parameters. After passing them through, the function checks if the account is valid and if its logged in.

Creation of smart contract 2



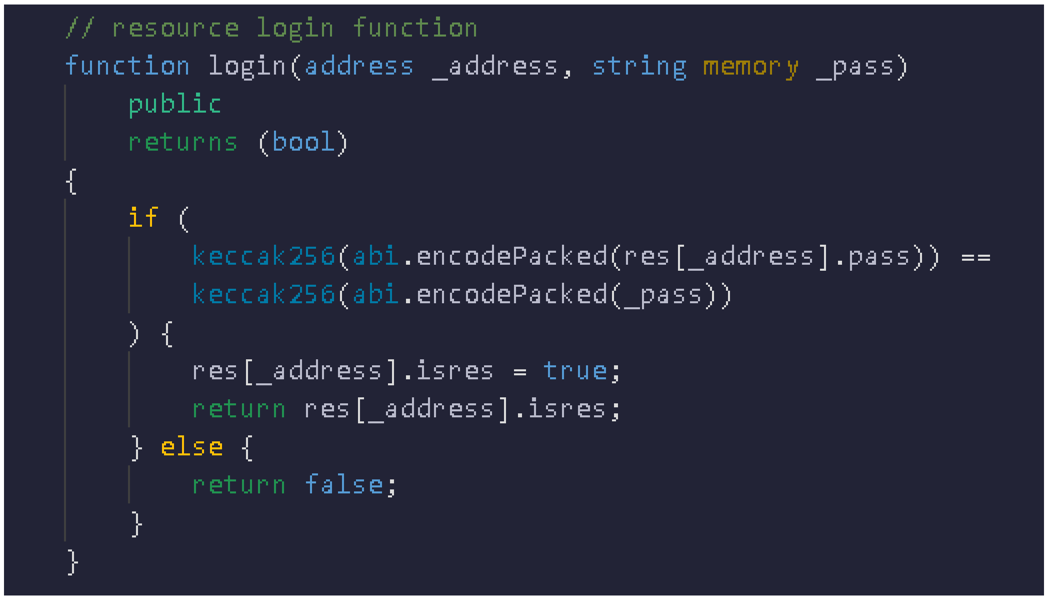
```
pragma solidity >=0.4.0 <0.6.0;

contract AuthorizationCloud {

    struct ResDetail {
        address addr;
        string name;
        string pass;
        string uniqueid;
        bool isres;
    }

    mapping (address => ResDetail) res;
```

Smart Contract 2 is very similar to Smart Contract 1. They must run on the same blockchain so they must be using the same version of solidity. The contract is called AuthorizationCloud.



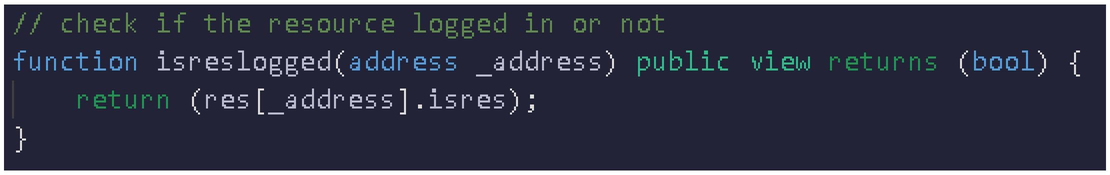
```
// resource registration function
function register(
    address _address,
    string memory _name,
    string memory _pass,
    string memory _uniqueid
) public returns (bool) {
    require (res[_address].addr != msg.sender);
    res[_address].addr = _address;
    res[_address].name = _name;
    res[_address].pass = _pass;
    res[_address].uniqueid = _uniqueid;
    res[_address].isres = false;
    return true;
}
```

The function register will attach the user in smart contract 1 to an account in smart contract 2. So first a user must create an account in smart contract 1 and use the same uniqueid to map the resource management contract with the authorization contract.



```
// resource login function
function login(address _address, string memory _pass)
    public
    returns (bool)
{
    if (
        keccak256(abi.encodePacked(res[_address].pass)) ==
        keccak256(abi.encodePacked(_pass))
    ) {
        res[_address].isres = true;
        return res[_address].isres;
    } else {
        return false;
    }
}
```

This function allows us to access their account to look through the resource management of the user.

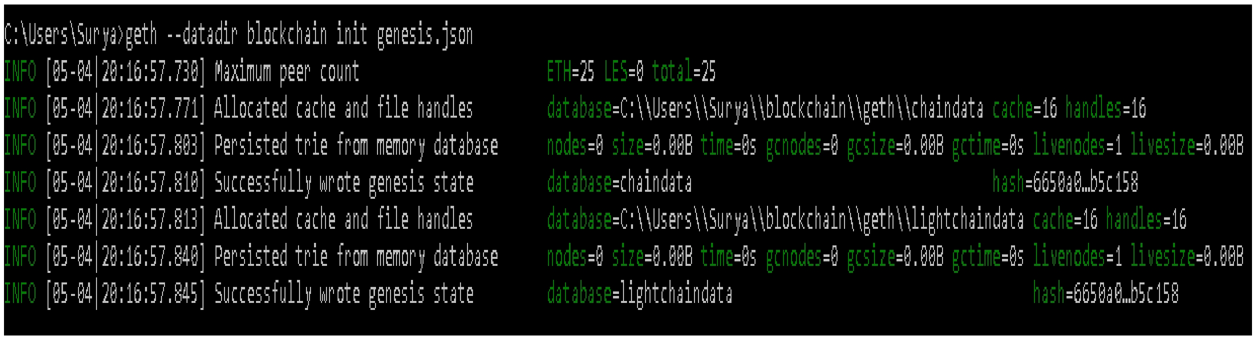


```
// check if the resource logged in or not
function isreslogged(address _address) public view returns (bool) {
    return (res[_address].isres);
}
```

This function checks whether the account already exists and doesn’t allow you to create duplicate accounts.

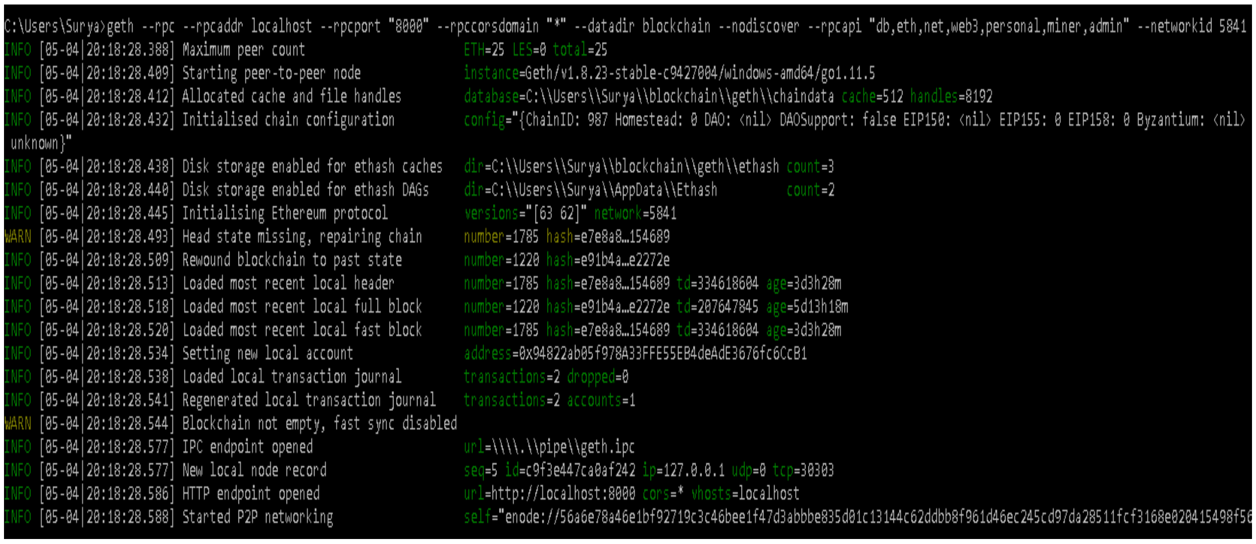
We can then implement this smart contract on a private test net to check if it’s working smoothly. We do this by downloading geth. By using the command lines:

geth --datadir blockchain init genesis.json



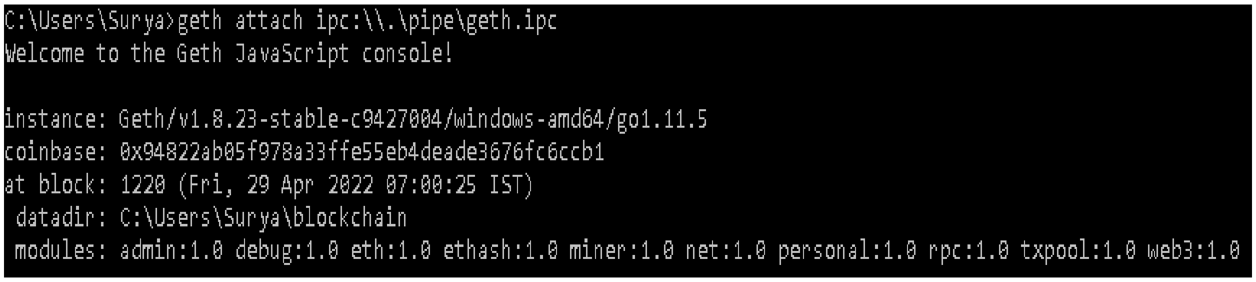
```
C:\Users\Surya>geth --datadir blockchain init genesis.json
INFO [05-04|20:16:57.730] Maximum peer count                       ETH=25 LES=0 total=25
INFO [05-04|20:16:57.771] Allocated cache and file handles         database=C:\\Users\\Surya\\blockchain\\geth\\chaindata cache=16 handles=16
INFO [05-04|20:16:57.803] Persisted trie from memory database      nodes=0 size=0.00B time=0s gcnodes=0 gcsize=0.00B gctime=0s livenodes=1 livesize=0.00B
INFO [05-04|20:16:57.810] Successfully wrote genesis state         database=chaindata                                    hash=6650a0…b5c158
INFO [05-04|20:16:57.813] Allocated cache and file handles         database=C:\\Users\\Surya\\blockchain\\geth\\lightchaindata cache=16 handles=16
INFO [05-04|20:16:57.840] Persisted trie from memory database      nodes=0 size=0.00B time=0s gcnodes=0 gcsize=0.00B gctime=0s livenodes=1 livesize=0.00B
INFO [05-04|20:16:57.845] Successfully wrote genesis state         database=lightchaindata                                    hash=6650a0…b5c158
```

geth --rpc --rpcaddr localhost --rpcport "8000" --rpccorsdomain "*" --datadir blockchain --nodiscover --rpcapi "db,eth,net,web3,personal,miner,admin" --networkid 5841



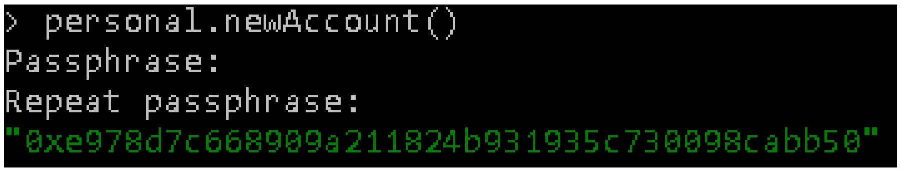
```
C:\Users\Surya>geth --rpc --rpcaddr localhost --rpcport "8000" --rpccorsdomain "*" --datadir blockchain --nodiscover --rpcapi "db,eth,net,web3,personal,miner,admin" --networkid 5841
INFO [05-04|20:18:28.388] Maximum peer count                       ETH=25 LES=0 total=25
INFO [05-04|20:18:28.409] Starting peer-to-peer node               instance=Geth/v1.8.23-stable-c9427004/windows-amd64/go1.11.5
INFO [05-04|20:18:28.412] Allocated cache and file handles         database=C:\\Users\\Surya\\blockchain\\geth\\chaindata cache=512 handles=8192
INFO [05-04|20:18:28.432] Initialised chain configuration          config="{ChainID: 987 Homestead: 0 DAO: <nil> DAOSupport: false EIP150: <nil> EIP155: 0 EIP158: 0 Byzantium: <nil>
 unknown}"
INFO [05-04|20:18:28.438] Disk storage enabled for ethash caches   dir=C:\\Users\\Surya\\blockchain\\geth\\ethash count=3
INFO [05-04|20:18:28.440] Disk storage enabled for ethash DAGs     dir=C:\\Users\\Surya\\AppData\\Ethash         count=2
INFO [05-04|20:18:28.445] Initialising Ethereum protocol           versions="[63 62]" network=5841
WARN [05-04|20:18:28.493] Head state missing, repairing chain      number=1785 hash=e7e8a8…154689
INFO [05-04|20:18:28.509] Rewound blockchain to past state         number=1220 hash=e91b4a…e2272e
INFO [05-04|20:18:28.513] Loaded most recent local header          number=1785 hash=e7e8a8…154689 td=334618604 age=3d3h28m
INFO [05-04|20:18:28.518] Loaded most recent local full block      number=1220 hash=e91b4a…e2272e td=207647845 age=5d13h18m
INFO [05-04|20:18:28.520] Loaded most recent local fast block      number=1785 hash=e7e8a8…154689 td=334618604 age=3d3h28m
INFO [05-04|20:18:28.534] Setting new local account                address=0x94822ab05f978A33FFE55EB4deAdE3676fc6CcB1
INFO [05-04|20:18:28.538] Loaded local transaction journal         transactions=2 dropped=0
INFO [05-04|20:18:28.541] Regenerated local transaction journal    transactions=2 accounts=1
WARN [05-04|20:18:28.544] Blockchain not empty, fast sync disabled
INFO [05-04|20:18:28.577] IPC endpoint opened                      url=\\\\.\\pipe\\geth.ipc
INFO [05-04|20:18:28.577] New local node record                    seq=5 id=c9f3e447ca0af242 ip=127.0.0.1 udp=0 tcp=30303
INFO [05-04|20:18:28.586] HTTP endpoint opened                     url=http://localhost:8000 cors=* vhosts=localhost
INFO [05-04|20:18:28.588] Started P2P networking                   self="enode://56a6e78a46e1bf92719c3c46bee1f47d3abbbe835d01c13144c62ddbb8f961d46ec245cd97da28511fcf3168e020415498f56
```

geth attach ipc:\\.\pipe\geth.ipc



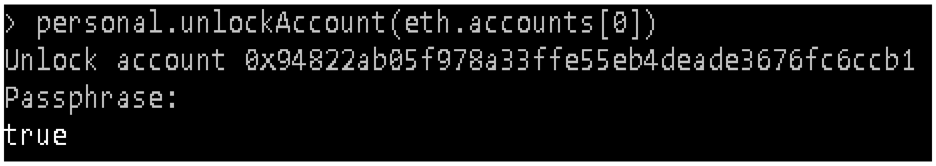
```
C:\Users\Surya>geth attach ipc:\\.\pipe\geth.ipc
Welcome to the Geth JavaScript console!

instance: Geth/v1.8.23-stable-c9427004/windows-amd64/go1.11.5
coinbase: 0x94822ab05f978a33ffe55eb4deade3676fc6ccb1
at block: 1220 (Fri, 29 Apr 2022 07:00:25 IST)
 datadir: C:\Users\Surya\blockchain
 modules: admin:1.0 debug:1.0 eth:1.0 ethash:1.0 miner:1.0 net:1.0 personal:1.0 rpc:1.0 txpool:1.0 web3:1.0
```

Now we create an admin account using the geth console commands. This account is used to mine and holds funds so we can deploy our smart contracts.



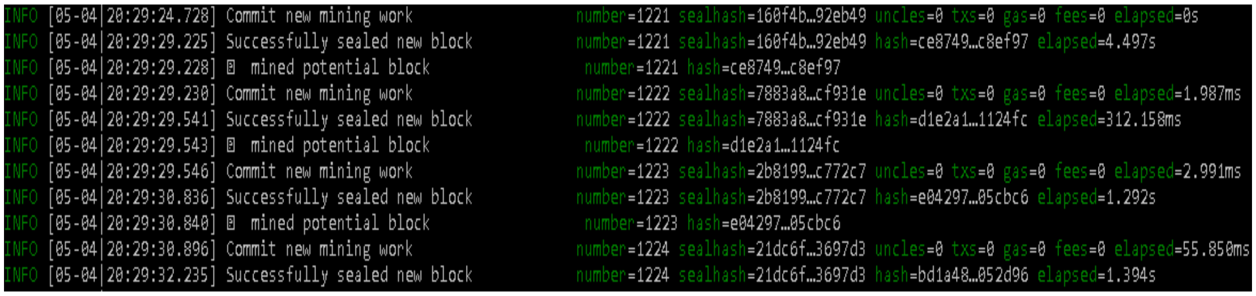
```
> personal.newAccount()
Passphrase:
Repeat passphrase:
"0xe978d7c668909a211824b931935c730098cabb50"
```

To actually deploy the smart contract, we must unlock the account by entering the password.



```
> personal.unlockAccount(eth.accounts[0])
Unlock account 0x94822ab05f978a33ffe55eb4deade3676fc6ccb1
Passphrase:
true
```

Deploying smart contracts require gas so we can mine on the admin account to counteract gas fees. We use the function miner.start().

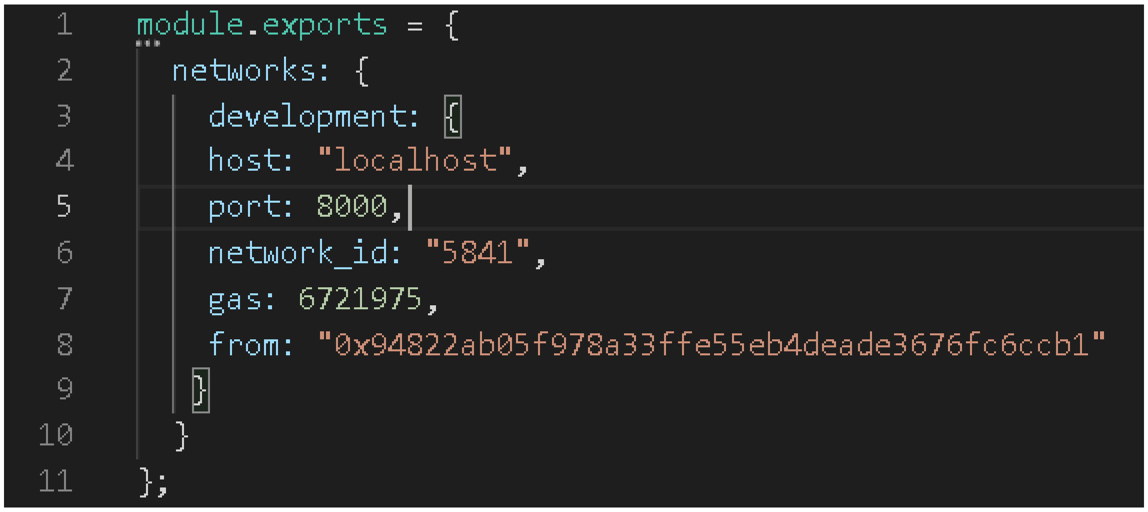


```
INFO [05-04|20:29:24.728] Commit new mining work                   number=1221 sealhash=160f4b…92eb49 uncles=0 txs=0 gas=0 fees=0 elapsed=0s
INFO [05-04|20:29:29.225] Successfully sealed new block            number=1221 sealhash=160f4b…92eb49 hash=ce8749…c8ef97 elapsed=4.497s
INFO [05-04|20:29:29.228] ⍰  mined potential block                  number=1221 hash=ce8749…c8ef97
INFO [05-04|20:29:29.230] Commit new mining work                   number=1222 sealhash=7883a8…cf931e uncles=0 txs=0 gas=0 fees=0 elapsed=1.987ms
INFO [05-04|20:29:29.541] Successfully sealed new block            number=1222 sealhash=7883a8…cf931e hash=d1e2a1…1124fc elapsed=312.158ms
INFO [05-04|20:29:29.543] ⍰  mined potential block                  number=1222 hash=d1e2a1…1124fc
INFO [05-04|20:29:29.546] Commit new mining work                   number=1223 sealhash=2b8199…c772c7 uncles=0 txs=0 gas=0 fees=0 elapsed=2.991ms
INFO [05-04|20:29:30.836] Successfully sealed new block            number=1223 sealhash=2b8199…c772c7 hash=e04297…05cbc6 elapsed=1.292s
INFO [05-04|20:29:30.840] ⍰  mined potential block                  number=1223 hash=e04297…05cbc6
INFO [05-04|20:29:30.896] Commit new mining work                   number=1224 sealhash=21dc6f…3697d3 uncles=0 txs=0 gas=0 fees=0 elapsed=55.850ms
INFO [05-04|20:29:32.235] Successfully sealed new block            number=1224 sealhash=21dc6f…3697d3 hash=bd1a48…052d96 elapsed=1.394s
```

We can successfully stop mining by using the function miner.stop().

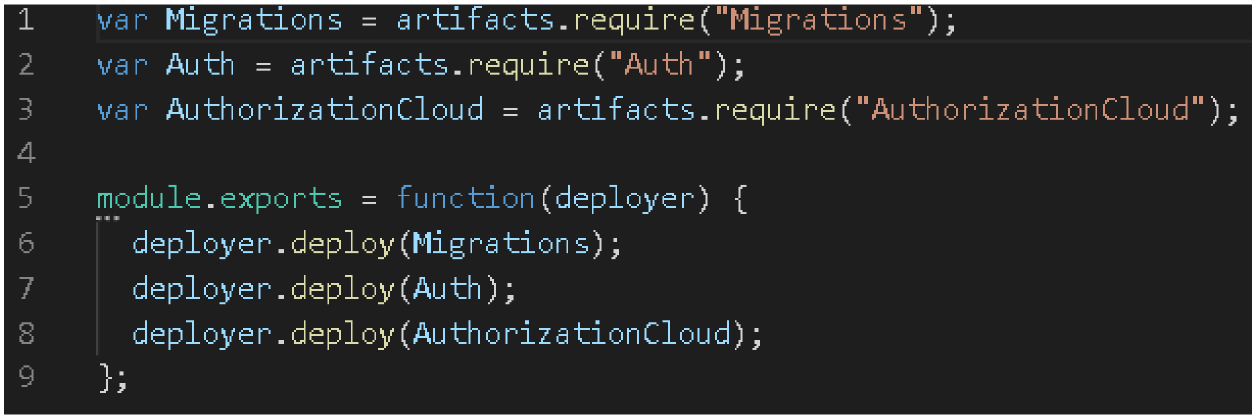
Now we must use truffle to initialize our smart contracts onto our blockchain. We run the following line on a new terminal to install the necessary truffle files. “truffle init”

By editing truffle-config.js we can connect truffle to our geth private blockchain.



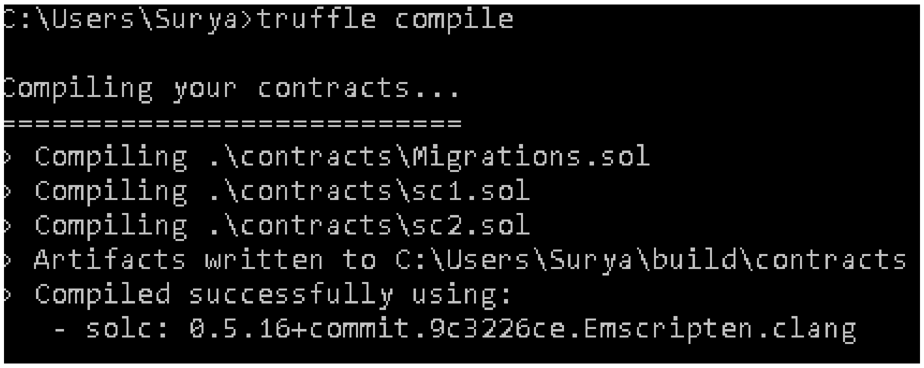
```
module.exports = {
  networks: {
    development: {
    host: "localhost",
    port: 8000,
    network_id: "5841",
    gas: 6721975,
    from: "0x94822ab05f978a33ffe55eb4deade3676fc6ccb1"
    }
  }
};
```

By editing 1_initial_migration.js we load the contracts into the blockchain.

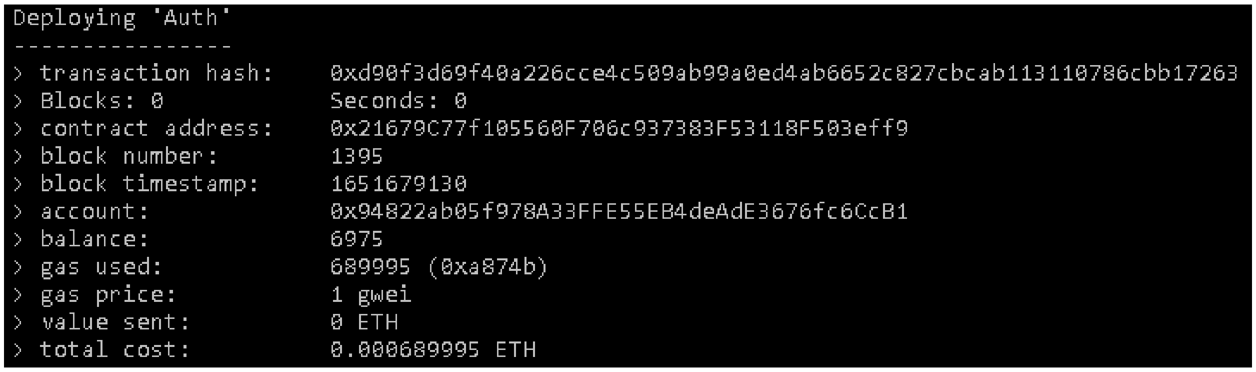


```
var Migrations = artifacts.require("Migrations");
var Auth = artifacts.require("Auth");
var AuthorizationCloud = artifacts.require("AuthorizationCloud");

module.exports = function(deployer) {
  deployer.deploy(Migrations);
  deployer.deploy(Auth);
  deployer.deploy(AuthorizationCloud);
};
```

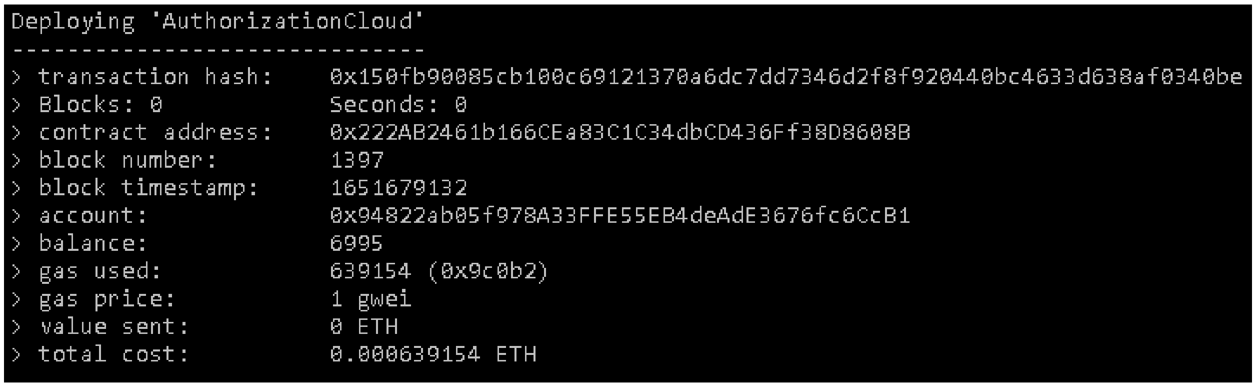
We must compile the contracts and then unlock the admin account to deploy the contracts and run the contracts.



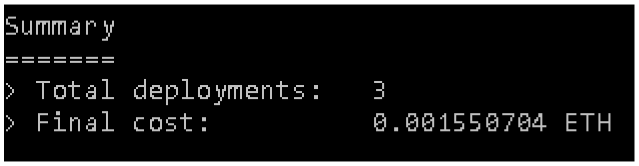
```
C:\Users\Surya>truffle compile

Compiling your contracts...
===========================
> Compiling .\contracts\Migrations.sol
> Compiling .\contracts\sc1.sol
> Compiling .\contracts\sc2.sol
> Artifacts written to C:\Users\Surya\build\contracts
> Compiled successfully using:
   - solc: 0.5.16+commit.9c3226ce.Emscripten.clang
```



```
Deploying 'Auth'
----------------
> transaction hash:    0xd90f3d69f40a226cce4c509ab99a0ed4ab6652c827cbcab113110786cbb17263
> Blocks: 0            Seconds: 0
> contract address:    0x21679C77f105560F706c937383F53118F503eff9
> block number:        1395
> block timestamp:     1651679130
> account:             0x94822ab05f978A33FFE55EB4deAdE3676fc6CcB1
> balance:             6975
> gas used:            689995 (0xa874b)
> gas price:           1 gwei
> value sent:          0 ETH
> total cost:          0.000689995 ETH
```

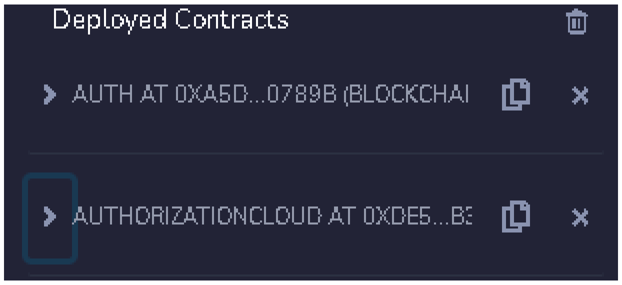


```
Deploying 'AuthorizationCloud'
------------------------------
> transaction hash:    0x150fb90085cb100c69121370a6dc7dd7346d2f8f920440bc4633d638af0340be
> Blocks: 0            Seconds: 0
> contract address:    0x222AB2461b166CEa83C1C34dbCD436Ff38D8608B
> block number:        1397
> block timestamp:     1651679132
> account:             0x94822ab05f978A33FFE55EB4deAdE3676fc6CcB1
> balance:             6995
> gas used:            639154 (0x9c0b2)
> gas price:           1 gwei
> value sent:          0 ETH
> total cost:          0.000639154 ETH
```

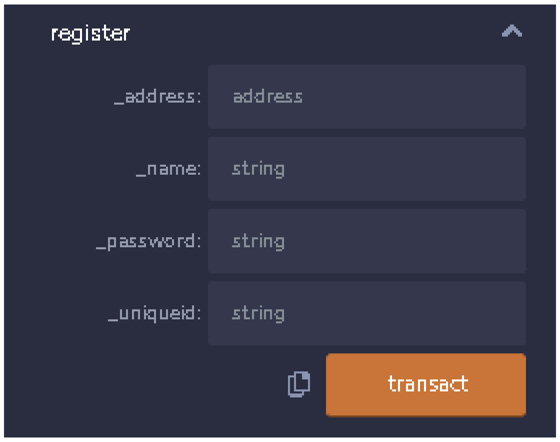


```
Summary
=======
> Total deployments:   3
> Final cost:          0.001550704 ETH
```

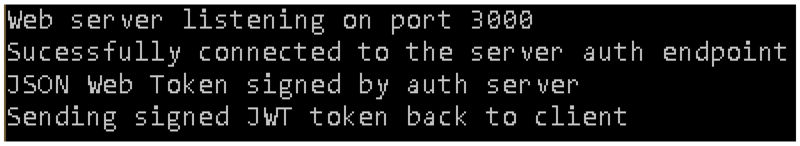
To run our contracts, we can use a Web3 Provider. Luckily the ide remix allows us to run a virtual host. As you can see, we have deployed both smart contracts.


Deployed Contracts
AUTH AT 0XA5D...0789B (BLOCKCHAI
AUTHORIZATIONCLOUD AT 0XDE5...B3

As you can see the function register has 4 parameters and we can run this to create accounts.



We use a third-party extension to run JSON web tokens so that smart contract 1 and 2 can communicate with each other.



```
Web server listening on port 3000
Sucessfully connected to the server auth endpoint
JSON Web Token signed by auth server
Sending signed JWT token back to client
```

Moving forward

We must create a more efficient way to implement JSON Web Tokens. Although we have a working model, this process of using a third-party supplier vastly impacts the security of the model. We also must run a test network with a small database so we can evaluate how well our model is working so we can deploy a working blockchain with a much larger database.

References:

[1] Deep, G.; Mohana, R.; Nayyar, A.; Sanjeevikumar, P.; Hossain, E. Authentication Protocol for Cloud Databases Using Blockchain Mechanism. *Sensors* **2019**, *19*, 4444. https://doi.org/10.3390/s19204444

[2] Ihle C., Sanchez O. (2019) Smart Contract-Based Role Management on the Blockchain. In: Abramowicz W., Paschke A. (eds) Business Information Systems Workshops. BIS 2018. Lecture Notes in Business Information Processing, vol 339. Springer, Cham. https://doi.org/10.1007/978-3-030-04849-5_30

[3] M. Kuperberg, "Blockchain-Based Identity Management: A Survey From the Enterprise and Ecosystem Perspective," in IEEE Transactions on Engineering Management, vol. 67, no. 4, pp. 1008-1027, Nov. 2020, doi: 10.1109/TEM.2019.2926471.

[4] D. C. Nguyen, P. N. Pathirana, M. Ding and A. Seneviratne, "Integration of Blockchain and Cloud of Things: Architecture, Applications and Challenges," in IEEE Communications Surveys

& Tutorials, vol. 22, no. 4, pp. 2521-2549, Fourthquarter 2020, doi: 10.1109/COMST.2020.3020092.

[5] S. Pavithra, S. Ramya and S. Prathibha, "A Survey On Cloud Security Issues And Blockchain," 2019 3rd International Conference on Computing and Communications Technologies (ICCCT), 2019, pp. 136-140, doi: 10.1109/ICCCT2.2019.8824891.

[6] Zibin Zheng, Shaoan Xie, Hong-Ning Dai, Weili Chen, Xiangping Chen, Jian Weng, Muhammad Imran,An overview on smart contracts: Challenges, advances and platforms,Future Generation Computer Systems,Volume 105,2020,Pages 475-491,ISSN 0167-739X,https://doi.org/10.1016/j.future.2019.12.019.

[7] Yang Liu, Debiao He, Mohammad S. Obaidat, Neeraj Kumar, Muhammad Khurram Khan, Kim-Kwang Raymond Choo, Blockchain-based identity management systems: A review, Journal of Network and Computer Applications,Volume 166,2020,102731,ISSN 1084-8045,https://doi.org/10.1016/j.jnca.2020.102731.