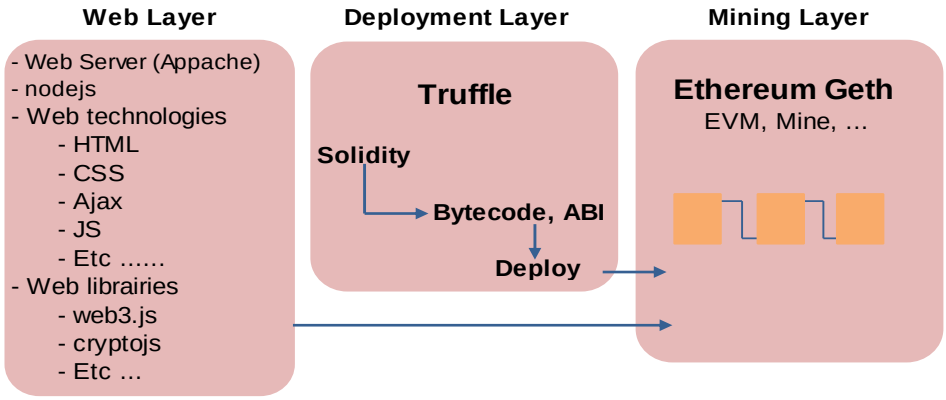
Mr Abdelhamid DEKHINET
University of BATNA 2
Department of Computer Science
Master 1 : Distributed Information Systems Engineering and Security

Lab : Deployment of Private Ethereum Blockchain

1. Introduction : By Ethereum private blockchain we mean an own blockchain installed locally. The main purpose of the lab is to setup a private blockchain development environment. the environment provides the necessary framework for the development of dApps (Distributed Applications) . The dApps include the creation of contract, its deployment in the blockchain, and the interaction with the contract via a front end web application. The target environment will contain a single node, for performance constraints, and will be deployed under linux Ubuntu.

2. The environment of dApps development : The environment of dApps development consists of setting up an own private Ethereum blockchain. As depicted in the figure below, the environment contains three layers. Web layer or the presentation layer intended for the end user. The compilation and the deployment layer for the development of smart contract. The mining layer or the blockchain itself, for the mining and running the smart contract. The blockchain software will be the Geth, it is the official implementation of the Ethereum written in Go.



3. Node characteristics

The node can be a simple PC :

- RAM 08 GB recommended or higher
- Virtual machine : VMware Workstation
- OS : Linux Ubuntu 64 bit (desktop-amd64)

4. Installation of the environment

The environment setup requires, mainly, **four** packages or applications, which are :

1- NodeJS 2-NPM 3-Geth 4-Truffle

NodeJS : Is an open-source, cross-platform, JavaScript runtime environment that executes JavaScript code outside a web browser : server-side scripting. Applications are written in JavaScript, and can be run within the Node.js runtime on OS X, Microsoft Windows, and Linux. It a platform built on Chrome's JavaScript runtime for easily building fast and scalable network applications. Node.js uses an event-driven, non-blocking I/O model that makes it lightweight and efficient, perfect for data-intensive real-time applications, JSON APIs based Applications, Data Streaming Applications and Single Page Applications.

To install Nodejs, run the following command:

```
ub2@ubuntu:~$ sudo apt-get install nodejs
```

To check the installation, issue the command:

```
ub2@ubuntu:~$ nodejs -v
v10.19.0
```

NPM : NPM stands for Node Package Manager is included as a recommended feature in Node.js installer, it is a package manager for the JavaScript programming language. It is the default package manager for the JavaScript runtime environment Node.js. It consists of a command line client, also called npm, and an online database of public and paid-for private packages, called the npm registry.
To install NPM, run the following command:

```
ub2@ubuntu:~$ sudo apt-get install npm
```

To check the installation, issue the command:

```
ub2@ubuntu:~$ npm -v
6.14.4
```

Geth : As mentioned before Geth is the official implementation of the Ethereum written in Go. The platform Geth is the Ethereum Blockchain itself. The mining process and the execution of transactions are done in Geth.
To install Geth, run the following commands:

```
ub2@ubuntu:~$ sudo apt-get install software-properties-common
```

```
ub2@ubuntu:~$ sudo add-apt-repository -y ppa:ethereum/ethereum
```

```
ub2@ubuntu:~$ sudo apt-get update
```

```
ub2@ubuntu:~$ sudo apt-get install ethereum
```

To check the installation, issue the command:

```
ub2@ubuntu:~$ geth version
Geth
Version: 1.9.15-stable
Git Commit: 0f77f34bb67b640bd8af22b215f3d279a1e21170
Architecture: amd64
Protocol Versions: [65 64 63]
Go Version: go1.14.2
Operating System: linux
GOPATH=
GOROOT=/build/ethereum-64QBnv/.go
```

Truffle : Truffle is framework and asset pipeline for Ethereum, aiming to facilitate the development process on Ethereum platform. With Truffle we get built-in smart contract compilation, linking, deployment and binary management, automated contract testing for rapid development, scriptable, extensible deployment & migrations framework and Network management for deploying to any number of public & private networks. The truffle package include, in addition to the solidity language, web3.js library.
To install Truffle, run the following commands:

```
ub2@ubuntu:~$ sudo npm install -g truffle
```

To check the installation, issue the command:

```
ub2@ubuntu:~$ truffle version
Truffle v5.1.33 (core: 5.1.33)
Solidity v0.5.16 (solc-js)
Node v10.19.0
Web3.js v1.2.1
```

5. Setup the Ethereum node - Miner
For performance reasons, our blockchain will contain a single node. Its extension follows the same procedure. To setup the node we can follow the steps:

Create the datadir folder : Each node (Miner) has its own datadir folder . It is used to store all the data needed by the node to mine normally: genesis block, accounts, ...

```
ub2@ubuntu:~$ mkdir node
ub2@ubuntu:~$ cd node
```

Create the Genesis file : Each blockchain starts with a genesis block that is used to initialize the blockchain and defines the terms and conditions to join the network. Our genesis block is called "***genesis.json***" and is stored under "***~/node***" folder.
The genesis.json file can be created using ***pupeth*** tools, included in the truffle suite, or any linux text editor like nano or vi.
The content of our genesis.json file is listed below:

```
{
 "config": {
          "chainId": 4224,
          "homesteadBlock": 0,
          "eip150Block": 0,
          "eip150Hash": "0x0000000000000000000000000000000000000000000000000000000000000000",
          "eip155Block": 0,
          "eip158Block": 0,
          "byzantiumBlock": 0,
          "constantinopleBlock": 0,
          "petersburgBlock": 0,
          "istanbulBlock": 0,
          "ethash": {}
 },
          "nonce": "0x0",
          "timestamp": "0x5ed53a49",
          "extraData": "0x0000000000000000000000000000000000000000000000000000000000000000",
          "gasLimit": "0x47b760",
          "difficulty": "0x200",
          "mixHash": "0x0000000000000000000000000000000000000000000000000000000000000000",
          "coinbase": "0x0000000000000000000000000000000000000000",
          "alloc": {
            "968cbb9bfbfc6ae46dbc732bb7d75ee077c35ef5": {
              "balance": "420000000000000000000000000000000000"
            },
            "0000000000000000000000000000000000000001": {
              "balance": "0x1"
            },
            "0000000000000000000000000000000000000002": {
              "balance": "0x1"
            },
            "00000000000000000000000000000000000000ff": {
              "balance": "0x1"
            }
          },
          "number": "0x0",
          "gasUsed": "0x0",
          "parentHash": "0x0000000000000000000000000000000000000000000000000000000000000000"
}
```

Initialize the Miner
At this time we initialize the private blockchain (one miner)with the genesis block. This operation will create the initial database stored under the data directory dedicated to the miner.
To initialize the miner, run the following commands :

```
ub2@ubuntu:~/node$ geth --datadir . init genesis.json
INFO [07-09|10:14:32.871] Maximum peer count                       ETH=50 LES=0 t
otal=50
INFO [07-09|10:14:32.897] Smartcard socket not found, disabling    err="stat /run
/pcscd/pcscd.comm: no such file or directory"
INFO [07-09|10:14:32.938] Allocated cache and file handles         database=/home
/ub2/node/geth/chaindata cache=16.00MiB handles=16
INFO [07-09|10:14:33.038] Writing custom genesis block
INFO [07-09|10:14:33.065] Persisted trie from memory database      nodes=7 size=1
.05KiB time="606.212µs" gcnodes=0 gcsize=0.00B gctime=0s livenodes=1 livesize=0.0
0B
INFO [07-09|10:14:33.066] Successfully wrote genesis state         database=chain
data                    hash="636d34…408d36"
INFO [07-09|10:14:33.067] Allocated cache and file handles         database=/home
/ub2/node/geth/lightchaindata cache=16.00MiB handles=16
INFO [07-09|10:14:33.093] Writing custom genesis block
INFO [07-09|10:14:33.094] Persisted trie from memory database      nodes=7 size=1
.05KiB time="295.018µs" gcnodes=0 gcsize=0.00B gctime=0s livenodes=1 livesize=0.0
0B
INFO [07-09|10:14:33.095] Successfully wrote genesis state         database=light
chaindata                    hash="636d34…408d36"
```

The initialization will create, mainly, two subfolders :

- ***geth***: contains the database of your private blockchain (chaindata).
- ***keystore***: location of your wallet used to store the accounts that you will create on the node.

```
ub2@ubuntu:~$ tree node/
node/
├── genesis.json
├── geth
│   ├── chaindata
│   │   ├── 000001.log
│   │   ├── CURRENT
│   │   ├── LOCK
│   │   ├── LOG
│   │   └── MANIFEST-000000
│   └── lightchaindata
│       ├── 000001.log
│       ├── CURRENT
│       ├── LOCK
│       ├── LOG
│       └── MANIFEST-000000
└── keystore

4 directories, 11 files
```

Create accounts : At this step we will create some accounts. This accounts will receive the reward Ether (mining reward) or to make transaction (smart contract). The first account created will be the default account.
To create account, run the following commands :

```
ub2@ubuntu:~$ cd node/
ub2@ubuntu:~/node$ geth --datadir . account new
INFO [07-09|10:50:43.624] Maximum peer count                       ETH=50 LES=0 t
otal=50
INFO [07-09|10:50:43.625] Smartcard socket not found, disabling    err="stat /run
/pcscd/pcscd.comm: no such file or directory"
Your new account is locked with a password. Please give a password. Do not forget
 this password.
Password:
Repeat password:

Your new key was generated

Public address of the key:   0x8e002B220e86fC5450A275aE2aCE48252a63a546
Path of the secret key file: keystore/UTC--2020-07-09T17-51-05.462765149Z--8e002b
220e86fc5450a275ae2ace48252a63a546

- You can share your public address with anyone. Others need it to interact with
you.
- You must NEVER share the secret key with anyone! The key controls access to you
r funds!
- You must BACKUP your key file! Without the key, it's impossible to access accou
nt funds!
- You must REMEMBER your password! Without the password, it's impossible to decry
pt the key!
```

We can create many accounts by issuing the same command. The list of accounts created can be displayed by running the command bellow :

```
ub2@ubuntu:~/node$ ls keystore/
UTC--2020-07-09T17-51-05.462765149Z--8e002b220e86fc5450a275ae2ace48252a63a546
UTC--2020-07-09T17-53-40.596317847Z--5a121f80e9202c08c42b958f5381be37b967776d
```

start running the node

To start running our node, we launch the ***geth*** command with a set of parameters, among these parameters :

- identity: name of our node
- networkid: this network identifier is an arbitrary value that will be used to pair all nodes of the same network.
- datadir: folder where our private blockchain stores its data

- rpc and rpcport: enabling HTTP-RPC server and giving its listening port number
- port: network listening port number, on which nodes connect to one another to spread new transactions and blocks
- nodiscover: disable the pair discovery mechanism
- mine: mine ethers and transactions
- unlock: id of the default account
- password: path to the file containing the password of the default account
- ipcpath: path where to store the geth.ipc for IPC socket/pipe, file created after starting the node

We store the geth command into a runnable script, named "*startnode.sh*". The parameters *--unlock 0 --password ./password.sec* unlock the default account and refers to the file *password.sec* containing the account[0] password.

```
ub2@ubuntu:~/node$ cat startnode.sh
geth --networkid 4224 --datadir "~/node" --nodiscover --rpc --ipcpath "~/node/geth.ipc"
 --rpccorsdomain "*" --rpcapi eth,web3,personal,net,miner,admin --unlock 0 --password .
/password.sec --allow-insecure-unlock
```

At this step we can start running the node, by launching the script file :

```
ub2@ubuntu:~/node$ ./startnode.sh
INFO [07-09|12:13:30.250] Maximum peer count                       ETH=50 LES=0 total=50
WARN [07-09|12:13:30.251] The flag --rpc is deprecated and will be removed in the future, please use --http
WARN [07-09|12:13:30.251] The flag --rpccorsdomain is deprecated and will be removed in the future, please u
se --http.corsdomain
WARN [07-09|12:13:30.251] The flag --rpcapi is deprecated and will be removed in the future, please use --ht
tp.api
INFO [07-09|12:13:30.265] Smartcard socket not found, disabling    err="stat /run/pcscd/pcscd.comm: no such
file or directory"
INFO [07-09|12:13:30.330] Starting peer-to-peer node               instance=Geth/v1.9.15-stable-0f77f34b/lin
ux-amd64/go1.14.2
INFO [07-09|12:13:30.331] Allocated trie memory caches             clean=256.00MiB dirty=256.00MiB
INFO [07-09|12:13:30.332] Allocated cache and file handles         database=/home/ub2/node/geth/chaindata ca
che=512.00MiB handles=524288
INFO [07-09|12:13:30.573] Opened ancient database                  database=/home/ub2/node/geth/chaindata/an
cient
INFO [07-09|12:13:30.678] Initialised chain configuration          config="{ChainID: 4224 Homestead: 0 DAO:
<nil> DAOSupport: false EIP150: 0 EIP155: 0 EIP158: 0 Byzantium: 0 Constantinople: 0 Petersburg: 0 Istanbul:
 0, Muir Glacier: <nil>, YOLO v1: <nil>, Engine: ethash}"
INFO [07-09|12:13:30.679] Disk storage enabled for ethash caches   dir=/home/ub2/node/geth/ethash count=3
INFO [07-09|12:13:30.679] Disk storage enabled for ethash DAGs     dir=/home/ub2/.ethash          count=2
INFO [07-09|12:13:30.708] Initialising Ethereum protocol           versions="[65 64 63]" network=4224 dbvers
ion=<nil>
WARN [07-09|12:13:30.709] Upgrade blockchain database version      from=<nil> to=7
INFO [07-09|12:13:30.745] Loaded most recent local header          number=0 hash="636d34…408d36" td=512 age=
1mo1w1d
INFO [07-09|12:13:30.746] Loaded most recent local full block      number=0 hash="636d34…408d36" td=512 age=
1mo1w1d
INFO [07-09|12:13:30.746] Loaded most recent local fast block      number=0 hash="636d34…408d36" td=512 age=
```

To interact with the miner and running command, we must attach to the node using the Geth attach console. The attachment can be done using IPC or RPC.

```
ub2@ubuntu:~/node$ geth attach geth.ipc
Welcome to the Geth JavaScript console!

instance: Geth/v1.9.15-stable-0f77f34b/linux-amd64/go1.14.2
coinbase: 0x8e002b220e86fc5450a275ae2ace48252a63a546
at block: 0 (Mon Jun 01 2020 10:26:33 GMT-0700 (PDT))
 datadir: /home/ub2/node
 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

> eth.coinbase
"0x8e002b220e86fc5450a275ae2ace48252a63a546"
> eth.accounts
["0x8e002b220e86fc5450a275ae2ace48252a63a546", "0x5a121f80e9202c08c42b958f538
1be37b967776d"]
> eth.getBalance(eth.coinbase)
0
> exit
```

```
ub2@ubuntu:~/node$ geth attach rpc:http://127.0.0.1:8545
Welcome to the Geth JavaScript console!

instance: Geth/v1.9.15-stable-0f77f34b/linux-amd64/go1.14.2
coinbase: 0x8e002b220e86fc5450a275ae2ace48252a63a546
at block: 0 (Mon Jun 01 2020 10:26:33 GMT-0700 (PDT))
 datadir: /home/ub2/node
 modules: admin:1.0 eth:1.0 miner:1.0 net:1.0 personal:1.0 rpc:1.0 web3:1.0

> eth.coinbase
"0x8e002b220e86fc5450a275ae2ace48252a63a546"
> eth.getBalance(eth.coinbase)
0
> miner.start()
null
> eth.getBalance(eth.coinbase)
2000000000000000000
> eth.getBalance(eth.coinbase)
10000000000000000000
> eth.getBalance(eth.coinbase)
42000000000000000000
> web3.fromWei(eth.getBalance(eth.coinbase))
52
> web3.fromWei(eth.getBalance(eth.coinbase))
54
>
```

You can initialize entirely the blockchain, before initialize the node, by deleting the geth folder in the node : **ub2@ubuntu:~/node$rm -rf geth/**

6. Create Ethereum Smart Contract

At this stage our private Blockchain is ready, we can create an Ethereum Smart Contract using Truffle. We will create a simple "Bank" Smart Contract. To start with this, first we create a new directory to store the Truffle project. And then in that directory, we will create a new Truffle project. It's done by **truffle init** command, which create all the necessary files required for a truffle project.

```
ub2@ubuntu:~$ mkdir truffle
ub2@ubuntu:~$ cd truffle/
ub2@ubuntu:~/truffle$ truffle init

Starting unbox...
=================

✔ Preparing to download box
✔ Downloading
✔ cleaning up temporary files
✔ Setting up box

Unbox successful, sweet!

Commands:

  Compile:        truffle compile
  Migrate:        truffle migrate
  Test contracts: truffle test
```

Write a "bank" Smart Contract

All the contracts should be written in the "contracts" directory. We will switch to this directory and create a contract with the name "bank.sol", using text editor :

```
  GNU nano 4.8                          bank.sol
pragma solidity >=0.4.22 <0.7.0;

contract bank {

    int cpt;

    constructor() public { cpt = 1; }

    function getbalance() view public returns(int) { return cpt; }
    function debit(int mnt) public { cpt = cpt - mnt ; }
    function credit(int mnt) public { cpt = cpt + mnt ; }

}

                        [ Read 14 lines ]
^G Get Help  ^O Write Out ^W Where Is  ^K Cut Text  ^J Justify   ^C Cur Pos
^X Exit      ^R Read File ^\ Replace   ^U Paste Text^T To Spell  ^  Go To Line
```

The smart contract cannot be executed by itself. We will have to make some configurations for it.

Configure Truffle Migration

To migrate our Smart Contract, we will have to add a file in the **"migrations"** directory. In this directory we will add a file named **"2_deploy_contracts.js"** with the following contents :

```
  GNU nano 4.8                  2_deploy_contracts.js
var Bank = artifacts.require("./bank.sol");
module.exports = function(deployer) {
   deployer.deploy(Bank);
};
```

Define the deployment parameters

The file named ***truffle.js*** or ***truffle-config.js*** must contains all the parameters needed to the deployment. The parameters include the network (Main net, Ropsten, Rinkebye, ...) in our case is local : 127.0.0.1:8545, the transaction Gas, the Gas price, the from account to cover transactions, ...

```
  GNU nano 4.8                    truffle-config.js

module.exports = {
  rpc: {
      host:"127.0.0.1",
      port:8545
      },

networks: {

     development: {
     host: "127.0.0.1",     // Localhost (default: none)
     port: 8545,            // Standard Ethereum port (default: none)
     network_id: "*",       // Any network (default: none)
     gas: 200000,           // Gas sent with each transaction (default: ~670000>
     gasprice: 2,
     from: "0x5a121f80e9202c08c42b958f5381be37b967776d",        // Account to s>
}
}
};
```

Compile the smart contract

```
ub2@ubuntu:~/truffle$ truffle compile

Compiling your contracts...
===========================
✔ Fetching solc version list from solc-bin. Attempt #1
✔ Downloading compiler. Attempt #1.
> Compiling ./contracts/Migrations.sol
> Compiling ./contracts/bank.sol
> Artifacts written to /home/ub2/truffle/build/contracts
> Compiled successfully using:
   - solc: 0.5.16+commit.9c3226ce.Emscripten.clang
```

Deploy the smart contract

For the deployment we used the account[1]. Before deploying the smart contract, we have to unlock the account and start the mining process, using the ***truffle migration*** command.

```
> personal.listAccounts
["0x8e002b220e86fc5450a275ae2ace48252a63a546", "0x5a121f80e9202c08c42b958f5381be
37b967776d"]
> personal.unlockAccount("0x5a121f80e9202c08c42b958f5381be37b967776d")

Unlock account 0x5a121f80e9202c08c42b958f5381be37b967776d
Passphrase:
true
> miner.start()
null
>
```

```
ub2@ubuntu:~/truffle$ truffle migrate

Compiling your contracts...
===========================
> Everything is up to date, there is nothing to compile.



Starting migrations...
======================
> Network name:    'development'
> Network id:      4224
> Block gas limit: 5618878 (0x55bcbe)


1_initial_migration.js
======================

   Deploying 'Migrations'
   ----------------------
   > transaction hash:    0x930d4c6d53eb128fdae9d408e0c600cb4e1ef90ab434570afa3c918d18a795a1
   > Blocks: 0            Seconds: 122
   > contract address:    0xDb9F49B66965D3296bC605EbEd1001E70BAa7746
   > block number:        186
   > block timestamp:     1594334146
   > account:             0x8e002B220e86fC5450A275aE2aCE48252a63a546
   > balance:             372
   > gas used:            164175 (0x2814f)
   > gas price:           20 gwei
   > value sent:          0 ETH
   > total cost:          0.0032835 ETH


   > Saving migration to chain.
   > Saving artifacts
   -------------------------------------
   > Total cost:          0.0032835 ETH

2_deploy_contracts.js
=====================

   Deploying 'bank'
   ----------------
   > transaction hash:    0xc62028597c2bb617759f05e13fddf7487a31a39d59107f6cede5b915a5878109
   > Blocks: 0            Seconds: 0
   > contract address:    0xc2aF83533Fc671880E7aF4E6b12Dd72Fb48AE8a8
   > block number:        191
   > block timestamp:     1594334340
   > account:             0x8e002B220e86fC5450A275aE2aCE48252a63a546
   > balance:             382
   > gas used:            131623 (0x20227)
   > gas price:           20 gwei
   > value sent:          0 ETH
   > total cost:          0.00263246 ETH


   > Saving migration to chain.
   > Saving artifacts
   -------------------------------------
   > Total cost:          0.00263246 ETH


Summary
=======
> Total deployments:   2
> Final cost:          0.00591596 ETH
```

Once the migration process is complete, all the deployment informations are displayed. The hash of the transaction relative to the bank smart contract, the block number hosting the transaction, the contract address in the blockchain, Etc...
The follow snapshot shows a part of the mining process.

```
INFO [07-09|15:38:59.095] 🔨 mined potential block                  number=189 hash="926a8b…21d028"
INFO [07-09|15:38:59.604] Submitted contract creation              fullhash=0xc62028597c2bb617759f05e13fd
487a31a39d59107f6cede5b915a5878109 contract=0xc2aF83533Fc671880E7aF4E6b12Dd72Fb48AE8a8
INFO [07-09|15:38:59.803] Successfully sealed new block            number=190 sealhash="317c64…252c68" ha
"c40161…d018e8" elapsed=712.998ms
INFO [07-09|15:38:59.803] 🔗 block reached canonical chain         number=183 hash="317bee…7b94a0"
INFO [07-09|15:38:59.803] Commit new mining work                   number=191 sealhash="86d703…734ff1" un
s=0 txs=0 gas=0      fees=0          elapsed="523.124µs"
INFO [07-09|15:38:59.804] Commit new mining work                   number=191 sealhash="33c41a…f328a9" un
s=0 txs=1 gas=131623 fees=0.00263246 elapsed=1.085ms
INFO [07-09|15:38:59.804] 🔨 mined potential block                 number=190 hash="c40161…d018e8"
INFO [07-09|15:39:01.305] Successfully sealed new block            number=191 sealhash="33c41a…f328a9" ha
"01c33f…380211" elapsed=1.500s
INFO [07-09|15:39:01.306] 🔗 block reached canonical chain         number=184 hash="9578db…51bd46"
INFO [07-09|15:39:01.309] Commit new mining work                   number=192 sealhash="70a4d7…e6bbd0" un
s=0 txs=0 gas=0      fees=0          elapsed=4.083ms
INFO [07-09|15:39:01.310] 🔨 mined potential block                 number=191 hash="01c33f…380211"
INFO [07-09|15:39:01.899] Submitted transaction                    fullhash=0x920139e81dbbbb7da6afb71ce92
313eb3bfec663fccb8c7c1af726583e0ec recipient=0xDb9F49B66965D3296bC605EbEd1001E70BAa7746
INFO [07-09|15:39:04.317] Commit new mining work                   number=192 sealhash="f82166…77c41d" un
s=0 txs=1 gas=27341  fees=0.00054682 elapsed="440.191µs"
INFO [07-09|15:39:07.625] Successfully sealed new block            number=192 sealhash="f82166…77c41d" ha
"24843c…e1620d" elapsed=3.308s
INFO [07-09|15:39:07.625] 🔗 block reached canonical chain         number=185 hash="b9fd9e…95beda"
INFO [07-09|15:39:07.626] Commit new mining work                   number=193 sealhash="e02ed1…a26d5e" un
s=0 txs=0 gas=0      fees=0          elapsed="697.113µs"
```

In addition to the deployment informations, a file named ***bank.json*** is created :

```
ub2@ubuntu:~/truffle/build/contracts$ ls
bank.json  Migrations.json
```

This file contains a lot of information, the most important are the bytecode of the contract, the contract address, the ABI. The last two are necessary to the Web front end application to interact with the blockcain.

7. Creating the Web front end application

Install Apache Web Server

To install Apache Web Server, run the commands :

```
ub2@ubuntu:~$ sudo apt-get update
ub2@ubuntu:~$ sudo apt-get install apache2
```

Verify the installation :

```
ub2@ubuntu:~$ systemctl status apache2
● apache2.service - The Apache HTTP Server
     Loaded: loaded (/lib/systemd/system/apache2.service; enabled; vendor pres
     Active: active (running) since Fri 2020-07-10 07:51:12 PDT; 1h 10min ago
```

Install web3 library

Web3 is a javascript library, it provides a set of functions for interacting with the blockchain. It can be considered as the middleware between the Web application and the blockchain. To install the library, run the following command :

```
ub2@ubuntu:~$ sudo npm install web3 --save
```

Normally, the installation process creates two files (package.json, package-lock.json) and a folder (node_modules). If the a problem occurs during the installaion, and the file package.json is not created, issue the command : ***npm init -y***

We can change the publication folder. In our case, instead of the default /var/www/html, we define /node_modules/web3/ as publication folder. So we must update the DocumentRoot parameter in the configuration file of apache.

Create the Web Front End

The first page of the Web front end application, e.g index.html, must inlude the web3 library, the contract address and the ABI.

```
<script src="https://cdn.jsdelivr.net/gh/ethereum/web3.js@1.0.0-beta.36/dist/web3.min.js"></script>
// or <script src="http://localhost/dist/web3.js"></script>
var address = "0x28BB1ba8B84Fe2eC0F6d1E0aA43F2Eb688BA881A";
var abi = [
        {           "inputs": [],
                    "payable": false,
                    "stateMutability": "nonpayable",
                    "type": "constructor"
         },
         ..............................]
```

The web application