Software Installation Guide

Viktor Vasylkovskyi

No Institute Given

The previous chapter explained in detail what how the proposed framework solve the problem stated in Chapter. Due to the broad nature of software artifact and involvement of a lot of different technologies arises the need for the detailed documentation about the software components installation. This chapter will present a full detailed Installation Guide for all the software that is required to be installed in order to execute the proposed application.

1 Pre-Installation Requirements

The following prerequisites and requirements must be satisfied in order to install artifact successfully.

1.1 Operational System

Current installation guide is dedicated for one of the following operational systems:

- Ubuntu 16.04 LTS (Xenial)
- Ubuntu 18.04 LTS (Bionic)

1.2 Database requirements

- MongoDB shell version v4.2 or higher - full installation guide available here `https://docs.mongodb.com/manual/tutorial/install-mongodb-on-ubuntu/#install-mongodb-community-edition-using-deb-packages`

1.3 Javascript runtime environment

- Nodejs v10.6.0 or higher.

  Installation via NVM available here: `https://tecadmin.net/install-nodejs-with-nvm/`

1.4 Angular

- Angular 7 and Angular CLI

  Installation available here `https://cli.angular.io/`

1.5 EOS Blockchain

EOS lightweight operational system is composed of the following items

- eosio 1.8.6+
- eosio.cdt 1.6.1+
- eosio.contracts 1.8.1+

1.6 ROS

- ROS Kinetic Kame
- ROS Melodic (only for Ubuntu 18.04 LTS bionic)

2 EOS blockchain Installation and Configuring Procedure

This section describes in detail the necessary steps in order to install eosio binaries and prepare blockchain to run locally.

The installation procedure consists of the following steps:

1. Install EOSIO
2. Install EOS Contract Development Kit (CDK)
3. Configure EOS Blockchain and run Boot Sequence

The full tutorial can be found on Eos developers community website:

- `https://developers.eos.io/eosio-home/docs/introduction`

2.1 Install EOSIO

All available versions of eosio binaries can be found here `https://github.com/EOSIO/eos/releases/`.

1. Choose the latest eosio debian for your ubuntu version and download it. Then install it executing the following command:
   - sudo apt install [Your previously downloaded eosio debian]
2. Setup a development directory:
   You're going to need to pick a directory to work from, it's suggested to create a contracts directory somewhere on your local drive.
   - mkdir contracts
   - cd contracts
3. Check the installation
   - open command line and type.
     - $ keosd

     The output should look like something in the next figure 5.1.
   - Check the Wallet.
     - $ cleos wallet list

     should return $ Wallets: []
   - Check nodeos endpoints.
     - $ curl http://localhost:8888/v1/chain/get_info

Fig. 1. EOSIO binaries keosd output example

2.2 Install the CDT

All available versions of eosio/cdt can be found here `https://github.com/EOSIO/eosio.cdt/releases/`. Installation of the CDT will take at least 30 minutes.

1. Choose the latest eosio debian for your ubuntu version and download it. Then install it executing the following command:
   - sudo apt install [Your previously downloaded eosio/cdt debian]
2. Clone eosio.cdt contracts as local binary into your contracts folder previously created
   - $ cd contracts
   - $ git clone –recursive https://github.com/eosio/eosio.cdt –branch v1.6.1 –single-branch. (or any other version higher that v1.6.1)
   - $ cd eosio.cdt
   - $ ./build.sh
   - $ sudo ./install.sh

   Installing eosio.cdt will make the compiled binary global so it can be accessible anywhere

2.3 Configure EOS Blockchain

Now since all the required eos binaries are successfully installed, it is time to configure the environment.

In order to configure EOSIO environment, the following steps are required:

1. Create Development Wallet
2. Boot Sequence

Create a development wallet Wallets are repositories of public-private key pairs. Private keys are needed to sign operations performed on the blockchain. Wallets are accessed using cleos.

1. Create a Wallet
   - $ cleos wallet create -n [my_wallet_name] –file wallet.txt

   This will create a wallet with a password and place it into wallet.txt file
2. Open a Wallet
   - $ cleos wallet open -n [my_wallet_name]
3. Unlock Wallet

   - $ cleos wallet unlock -n [my_wallet_name] –password [your password from wallet.txt]
4. Create Development KeyPair
   - Create a development keypair: $ cleos create key –to-console.
   - Now import development keypair: $ cleos wallet import –private-key [private-key] -n [my_wallet_name]
5. Check if the key was imported correctly:
   - $ cleos wallet private_keys –password [your password from wallet.txt] -n [my_wallet_name]

We are going to need development key pair for the boot sequence. We will refer to them as EOS_PUB_DEV_KEY and EOS_PRIV_DEV_KEY.

Blockchain Boot Sequence Boot sequence is the process of starting of a blockchain. Current project is already enabled with boot sequence setup, we are just going to adapt it to the development wallet created in previous subsection. For now we are ready to clone our development project and configure project's blockchain.

1. **Clone the project**
   - $ git clone `https://github.com/vvasylkovskyi/eos-web.git`
2. **Genesis**: in the project "src/blockchain" folder you can find our blockchain smart contracts and genesis point of a blockchain with main configuration files
   - Open genesis.json file and paste into EOS_PUB_DEV_KEY public development key stored your development wallet in previous step.
   - Open genesis directory and open genesis_start.sh and start.sh and substitute EOS_PUB_DEV_KEY and EOS_PRIV_DEV_KEY with your development key-pair.

   The scripts you can see in genesis folder are the blockchain boot control scripts. When starting a new blockchain from scratch, you need to run **./genesis_start.sh**. After that you shall see a new folder created "blockchain". In that folder you can find 2 folders and 2 files:
   - In **nodeos.log** you can see the logs of your blockchain producing blocks. To do so run in terminal $ tail -f nodeos.log
   - **eosd.pid** is where you can find the process id of the blockchain.
   - **data** folder is where the created blocks are stored.
   - **config** is a configuration folder which contains **config.ini** file where all the blockchain is configured. The main configuration points for the current projects are
     - http-server-address = 127.0.0.1:8888 (Blockchain RPC API host)
     - access-control-allow-origin = *
     - access-control-allow-headers = true
     - enable-stale-production = true
     - producer-name = eosio (the blocks producer account)
     - plugin = eosio::producer_plugin (enable eosio blocks producing)

Fig. 2. Smart Contract Deployment output

- plugin = eosio::chain_api_plugin
- plugin = eosio::http_plugin

This configuration is required to be set up in order to execute project successfully.

– **start.sh** script is used to start blockchain after it's database has been created. In other words, you can run genesis_start.sh only once, after that, to initiate blockchain you are going to have to run start.sh script.
– **stop.sh** script is used to stop blockchain.
– **clean.sh** script cleans all the blockchain data. Be careful, as using this script you may delete the whole blockchain folder and therefore loose config.ini file.
– **hard_start.sh** - Sometimes, restarting a nodeos requires the paramenter –hard-replay which is replaying all the transactions from the genesis. You will know when this is needed when you'll start nodeos and you'll notice an error in the log file mentioning exactly this "perhaps we need to replay"

3. **Smart Contracts**
After finishing the process of creation of the blockchain, run start.sh script and let blockchain be running so you can deploy the smart contracts which you can find in "src/blockchain/src" folder. Next follows the process of deploying the smart contracts to the blockchain.

– Open "src/blockchain" folder (blockchain root)
– In order to be able to deploy smart contract you need to create an account to which smart contract will be deployed. Each account has a private/public key pairs, so first thing to do is to create 2 key pairs. Repeat the following process two times (One key pair for each account):
  - $ cleos wallet unlock -n [mywalletname] password [your password from wallet.txt]
  - $ cleos create key to-console.
  - $ cleos wallet import private-key [private-key] -n [mywalletname]
– $ cleos create account eosio login [EOS_PUB_KEY] -p eosio@active
– $ cleos create account eosio storage [EOS_PUB_KEY] -p eosio@active
You can check if your accounts where created successfully by checking account info:
  - $ cleos get account [account_name]

Now that you have 2 accounts, build smart contracts:

- $ eosio-cpp src/login/login.cpp -o build/login/login.wasm –abigen –contract=login
- $ eosio-cpp src/storage/storage.cpp -o build/storage/storage.wasm –abigen –contract=storage

And finally, deploy each to it's account:

- $ cleos set contract login build/login -p login@active
- $ cleos set contract storage build/storage -p storage@active

After successfully deploying smart contracts you should see output similar to the one in the following Figure 5.2.

After finishing the previous steps, you have private blockchain network with one node producing.

3 Scatter Wallet

In order for users to perform transactions on blockchain, they need to have keys to sign the transactions. Scatter Wallet is a secure keys management software that each user is required to have installed in order to be able sign transaction be able to perform actions on blockchain. This section provides a guide to installation of **Scatter Wallet**.

- Download latest scatter version from `https://get-scatter.com/download`
- Open Scatter Wallet downloaded in previous step. You may need to give a file an executable permission $ chmod +x [your_Scatter_file]. After opening it, type your password so you can start using scatter.

For the first time of using scatter you are going to have an empty wallet. In order to start using scatter with your application, open networks tab and click add custom network. You will be present with a screen as in figure x. Fill the descriptions of your blockchain host, protocol and port and at the end, insert your blockchain chainId. If you don't know your chainId, run:

- $ cleos get info

And it will return a set of informations with the chainId among it.

Once you blockchain network is successfully added to your scatter, you are ready to import your private keys.

- To import keys to scatter, select wallet tab and click import key button. After that, select Text Option.
- At last, insert the private key that you have created in cleos previous steps. Beware you may want for the private key to be associated to the corresponding account, so before importing the private key into Scatter, make sure that you have created the account in eosio and associated the private key to it.
- Once you have successfully imported the key, scatter will automatically associate the account to it. You will see something like in the next figure

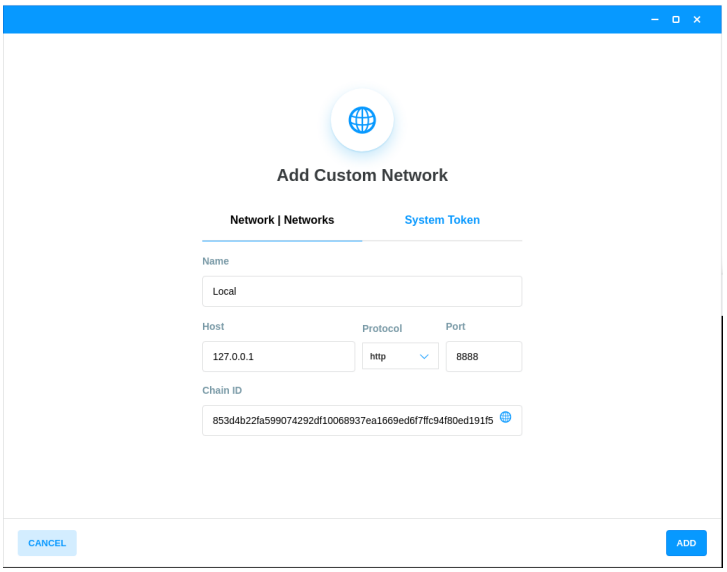

Fig. 3. Scatter Network Window

4 Application

In order to run the application, it is required to install the libraries before starting.

- Open project root folder
- $ npm install

npm install may take a while before installing all external libraries required to run the application. This libraries includes all the necessary assets for running the Angular frontend and Express Backend, alongside with the integration of the latest with the mongo database.

4.1 Angular

In angular FE you can find environment.ts file. Here is where all the environments are set. In order to start using application, you should change the following items according to your connections.

- **nodeos_chainId** - you can find it by executing *cleos get info*.
- **NodeOS** - set the general nodeos info, such as you blockchain service endpoint and port.
- **Mongo** - your mongo database url
- **Express** - express middleware url

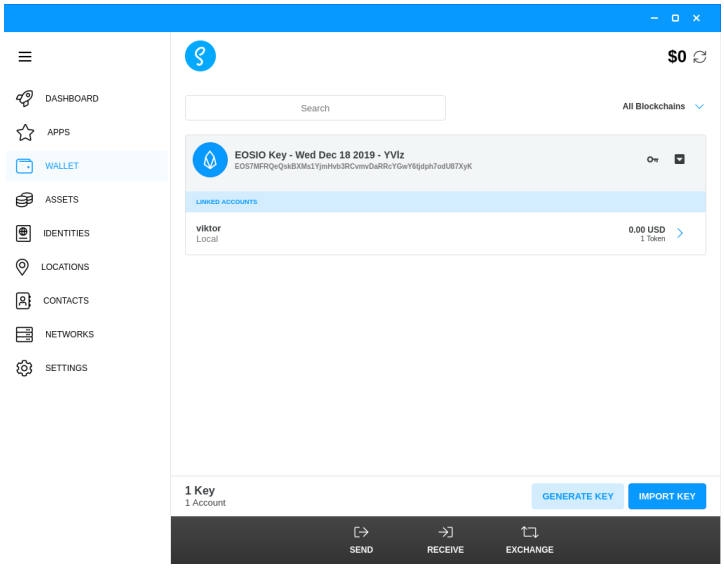

Fig. 4. Scatter After Importing an account

4.2 Express

In the Express BE you can find environment.js file where all the environments are set. Make sure that you change the environments accordingly to you workspace.

- **producer_private_key** - here you define the private key for the robot (the one who is going to create transactions to the blockchain. The key must be the same as it is in your blockchain wallet.)
- **Mongo** - Your mongo database endpoints
- **db**$_name$ – $yourmongodatabasename$.

5 Troubleshooting

5.1 Mongo Troubleshooting

- If connection fails, execute following steps:
  - $ sudo rm /var/lib/mongodb/mongod.lock
  - $ sudo service mongod start
  - $ mongo

5.2 NodeJS TroubleShooting

- Watcher failure Error: ENOSPC: no space left on device, watch 'src/server/index.js...
  - execute $ sudo sysctl fs.inotify.max_user_watches=582222 sudo sysctl -p or to make it permanent
  - $ echo fs.inotify.max_user_watches=582222 — sudo tee -a /etc/sysctl.conf sudo sysctl -p

5.3 EOS Troubleshooting

– https://developers.eos.io/eosio-nodeos/docs/troubleshooting