# Instructions on installing Rubix on Cloud VM – Ubuntu

Welcome to Rubix Community. This document gives you the next steps for setting up of your VM to make it ready to start transactions.

## Before you install:

We recommend:

a. Installing the Rubix Wallet software as a non-root user with sudo privileges and home folder option
b. Installing Java JDK 11 or Open JDK version 11
c. Open Ports: This is explained in detail in the section Checking the ports
d. IP V6 / IP V4: IPFS has some compatibility issues with IP V6. You need to enable IP V4.
e. Static Public IP can avoid many problems. In case you don't have a static Public IP then please refer to the section Adding Bootstrap nodes:
f. Additional utilities:
    a. Screen: This will allow you to run Java and IPFS in non-terminated terminals and allow you to access the terminal output.
    b. jq: This is a parser for JSON objects and can be used for formatted display of JSON outputs

## Installation:

a. Login as a non-root user with sudo privileges
b. Please follow the instructions at this link for installation of java, ipfs and Rubix wallet as given in the page. The latest version of Rubix is Rubix-1.6

## Creating a Decentralized ID (DID)

If this is a fresh wallet, then before you start using it, you need to create your DID key.

```
curl --location --request POST 'http://localhost:1898/create' --form 'data="your-passphrase"' --form 'image=@"imagepath"'
```

**your-passphrase** is a string literal which along with the image is used to create the cryptographic hash of DID Key. The image should be a 256x256 px PNG file. You can get this using image search or you can create your own PNG image with your favourite image editing tool.

If the creation of DID is successful, you will get a success message along with the DID details. The DID Details are stored in the file: ~/Rubix/DATA/DID.json

The key components of your DID are:

a. peerid: This is the id that ipfs uses to communicate with your node. This is a unique hash across the network
b. didHash: This is the DID Key. This is tied to your wallet and will be used for all transactions from and to your wallet.
c. walletHash: This is a unique hash of your wallet. This is used along with your didHash internally by Rubix Wallet

After creating a new Wallet, run the sync to synchronize the wallet with the details of other nodes in the network.

```
curl http://localhost:1898/sync
```

## Starting or Restarting the wallet

The wallet needs ipfs daemon (more details) and rubix.jar (more details) components to be running. This instruction can be followed if you are having screen utility. In case you are not having screen utility, run it using nohup and && to start them as background processes.

Before you start these processes, ensure that the earlier instances are completely removed. You can remove earlier instances by giving the commands:

```
killall -9 java
killall -9 ipfs
```

Start the ipfs daemon:

```
screen -dmS ipfs bash -c 'ipfs daemon'
```

Start the rubix.jar:

```
screen -S rubixJar -dm -L -Logfile ~/rubixConsole.log bash -c "java -jar
/home/$USER/rubix-work-dir/rubix-1.6.jar"
```

Calling the start API

```
curl http://localhost:1898/start
```

## Syncing with DID Server

Rubix wallet automatically syncs up with the DID server as and when necessary. In case you need to manually sync up with the DID server, give the command:

```
curl http://localhost:1898/sync
```

## Receive Tokens

When you are running Rubix, the wallet is automatically ready to receive tokens.

## Transfer Tokens

When you are running Rubix, you are ready to start transferring tokens to other wallets.

```
curl --header "Content-Type: application/json" --request POST
http://localhost:1898/initiateTransaction --data '{ "receiver":
"445f59c3d71c6769124470cf4b82ca0b9b1626aec4f14f50a8f1e6a13e1fc70d", "tokenCount":1,
"comment":"transaction comments", "type":1}'
```

The components of the commands are:
receiver: The DID Hash of the receiver Node
tokenCount: The number of tokens that need to be transferred.
comment: The transaction comments. This is a string. If any special characters are used, be sure to escape the characters in the command
type: This is defaulted to 1. Only in special cases of fixed quorum (in specific contexts of custom wallets tied with other applications, can this be different)

## Connections using IPFS:

Rubix uses IPFS for communication with Peers. The peer id is the key identifying each node. IPFS uses several ports for its communication. The ports necessary for IPFS are 4001 and 5001. To check if these ports are open, please refer to this section

## Running the Rubix Jar

You need the java 11.0.10 2021-01-19 LTS for running Rubix Wallet. Please don't use Open JDK since this is not fully tested with Rubix Wallet and there may be some compatibility issues. We are working on testing the Rubix Wallet on Open JDK. Once tested, a new JAR file will be released.

Rubix Wallet needs the following ports to be opened.
15010, 15011, 15030, 15031, 15032, 15033, 15034, 15035, 15036, 15040, 15080, 15081, 15091

To check if these ports are open, please refer to this section

## Adding Bootstrap nodes:

Rubix wallet is designed to run on nodes that have Public Ips as well as nodes that have only Private IPs. If your node is having a static Public IP then no further configuration is needed. The system will automatically be able to connect to the IPFS network.

If your node is running on Private IP then you need to connect with another Rubix node running on Public IP

By default, Rubix comes configured with 2 IPFS bootstrap nodes.

To list the bootstrap nodes already configured, please give the command:

```
curl http://localhost:1898/bootstrap
```

To add a bootstrap node, please give the following command:

```
curl --header "Content-Type: application/json" --request POST
http://localhost:1898/bootstrap?id=/ip4/115.124.117.37/tcp/4001/p2p/QmWXELAo
KJsCMFoW3j6pFmXEhouwKgWiK7wN6uLyuX6ULV
```

To add new bootstrap nodes to the system, give the following commands:

```
curl --header "Content-Type: application/json" --request POST
http://localhost:1898/bootstrap?id="/ip4/115.124.117.37/tcp/4001/p2p/QmWXELA
oKJsCMFoW3j6pFmXEhouwKgWiK7wN6uLyuX6ULV"
```

To check whether the bootstrap node is configured correctly, you can check the connection by giving the following command:

```
ipfs ping QmWXELAoKJsCMFoW3j6pFmXEhouwKgWiK7wN6uLyuX6ULV
```

## Checking the ports

Rubix wallets need specific ports to be opened for communication (both inbound and outbound). The ports necessary for each component is given in that section. To see if the specific port is opened or not, follow these steps:

Open a new terminal window:

Give the command
lsof -i -n -P |grep {Port number}

For example if you need to check if port 4001 is opened, then the command will be

lsof -i -n -P |grep 4001

If you get an output for the above command giving the list of processes using the port, then Congratulations!!! The port is opened and the operation is successful.

If you don’t get any output for this command, then it is very likely that the firewall need to be configured to allow this.

Caveat: Giving the sudo ufw command to open the port on the node may not work if the router or the firewall hardware is having overrides.