**MAP Validator Node Setup**

<https://www.maplabs.io/>

<https://bridge.maplabs.io/>

<https://makalu.mapscan.io/>

<https://medium.com/marcopolo-protocol/map-mainnet-goes-live-on-august-31-4d3b044fcd8c>

**About MAP Protocol**

MAP Protocol is the omnichain Network for An Interoperable Web3 that enables developers to build omnichain dApps in simple steps. MAP Protocol lets cryptocurrency, NFT, and data flow around all chains securely and seamlessly at minimum cost. MAP Protocol is the only infrastructure made for omnichain dApps that connects all chains, provides security finality, charges the lowest cross-chain gas fees, and offers complete dApp development service toolkits.

**PREREQUISITES:**

* **Server with: 16 GB of Ram**, **a Quad core 2.5 GHz (64-bit) CPU, 256GB SSD and a 100Mb/s Ethernet connection w/ fiber Internet (ideally redundant connection and HA switches)**
* [**Ubuntu 20.04 LTS**](https://releases.ubuntu.com/20.04/) +
* **Your account needs to have at least 1,000,000 MAP**

**Update Linux**

To start we need to ensure that the server is up to date and has all the dependencies required to operate the node.

This guide uses ***Ubuntu 20.04.5 LTS***

First thing is to add update and upgrade the Ubuntu system.

sudo apt update && apt upgrade -y

**Setup User and Firewall**

See Hank the Cranks guide here:

<https://medium.com/@htctimbo/1cfce50d9d2e>

**Setup Server with prerequisites**

Next we need to ensure the server has all the dependencies required to operate the node.

\*\*Building atlas requires \*\*

* Git
* Go (version 1.14 or later)
* C compiler
* Python (optional to use the tool)

apt install git

apt install python3

apt install python3-pip

apt install gcc

apt install make

**Download Go language and do checksum (find latest version at**[**https://go.dev/dl/**](https://go.dev/dl/)**):**

curl -OL <https://go.dev/dl/go1.19.linux-amd64.tar.gz>

**Untar and then remove the tar:**

tar -C /usr/local -xvf go1.19.linux-amd64.tar.gz  
rm go1.19.linux-amd64.tar.gz

**Set permanent path in profile:**

nano ~/.profile

**Add the following lines at the end of the file:**

export PATH=$PATH:/usr/local/go/bin  
export GOROOT=/usr/local/go  
export GOPATH=$HOME/atlas  
export PATH=$GOPATH/bin:$GOROOT/bin:$PATH

CTRL-X to quit, press Y and ENTER to confirm

**Refresh profile:**

source ~/.profile

**Clone atlas:**

git clone <https://github.com/mapprotocol/atlas.git> && cd atlas

**Make atlas:**

make atlas

**Make marker:**

make marker

**Download Tool (Optional)**

git clone <https://github.com/johnashu/pymap.git>

**Install Requirements**

pip3 install -r requirements.txt

**Create .Env file**

Edit example.env file and save it in the pymap dir as .env

nano /pymap/example.env

# location of the MAP binaries from Go installation

binaries=/home/atlas/build/bin

# RPC address of testnet

testnet=http://18.142.54.137:7445

# RPC address of Mainnet

rpcaddr=https://poc3-rpc.maplabs.io

# RPC port number - Not really used unless setting up a local network..

rpcport=False

# Password as a string, used for most CLI methods in Marker

password=password

# Password File location.  Same password as above but stored in a file - used for Atlas Start 'Node'

passwordFile=/home/maffaz/password

# Location of the KeyStore file for this env role

keystore=/home/maffaz/pymap/admin/keystore/UTC--2022-08-26T23-45-19.943014769Z--1234567890abcdef123456

# Type of role for creating new accounts - validator / voter / ...

namePrefix=validator

# Number Map to lock in a node.

lockedNum=10

# Private key of the authorised signer for the node

signerPriv=

# Address of Validator when asked

validator=0x

# Target Validator address. When not calling the node address this can be an External node (I.e. when voting)

target=0x

# number of MAP to vote for a validator

voteNum=5

# Rate of commission this validator will charge

commission=40000

# Directory of data, node, keystore, password etc for this env file

datadir=admin

# Main Validator address that is signing blocks

miner.validator=0x

# Signer Address of Node

unlock=0x

# Type of Sync (Full)

syncmode=full

# Communication Port for the Node

port=30321

# Working Dir for SystemD

working\_dir=/home/maffaz

Ctrl+X to save but rename as .env

Graphical user interface, application, website

Description automatically generated

Type ‘y’ then edit the name to .env

A screenshot of a computer

Description automatically generated with medium confidence >>> A screenshot of a computer

Description automatically generated with medium confidence

**Start Toolbox**

Start the toolbox

Python3 start\_tool.py

**Create Keystores for Account and Signer**

Option 2

Or

Command to Process:

/home/maffaz/atlas/build/bin/atlas --datadir /home/maffaz/account account new

**A picture containing text

Description automatically generated**

**Buy Some MAP!!**

**# Buy MAP**

Map is available on several chains and can be purchased with the contract addresses provided.

\* Ethereum - Uniswap / Kucoin

> 0x9E976F211daea0D652912AB99b0Dc21a7fD728e4

​

\* BNB Chain - Pancake Swap

> 0x8105ECe4ce08B6B6449539A5db23e23b973DfA8f

\* Polygon - Quickswap

> 0xBAbceE78586d3e9E80E0d69601A17f983663Ba6a

**# Bridge MAP to MAP protocol**

Goto the MAP bridge and send the MAP from Eth / Matic / BNB -> MAP Address created above.

> https://bridge.maplabs.io/#/home?sourceNetwork=ETH&destNetwork=MAP&ts=1661547728846

1. Select the chain from and the amount of map to bridge

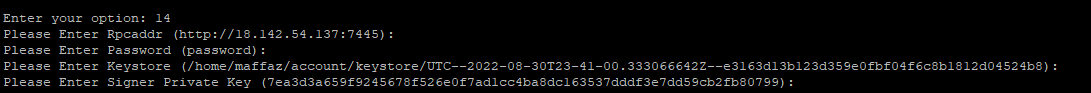
2. Approve contract

3. bridge assets

4. Add MAP protocol to Metamask by changing the 'From' Protocol to 'MAP'.  This will trigger auto adding the network to Metamask

You should now be able to see you Map in Metamask and also by selecting 'Get Balance' (22) in the Tool.

**Authorise Signer**

****

Command to Process:

/home/maffaz/atlas/build/bin/marker authorizeValidatorSigner --signerPriv 7ea3d3a659f9245678f526e0f7ad1cc4ba8dc163537dddf3e7dd59cb2fb80799 --rpcaddr http://18.142.54.137:7445 --password password --keystore /home/maffaz/account/keystore/UTC--2022-08-30T23-41-00.333066642Z--e3163d13b123d359e0fbf04f6c8b1812d04524b8

A picture containing text

Description automatically generated

LOCK MAP

