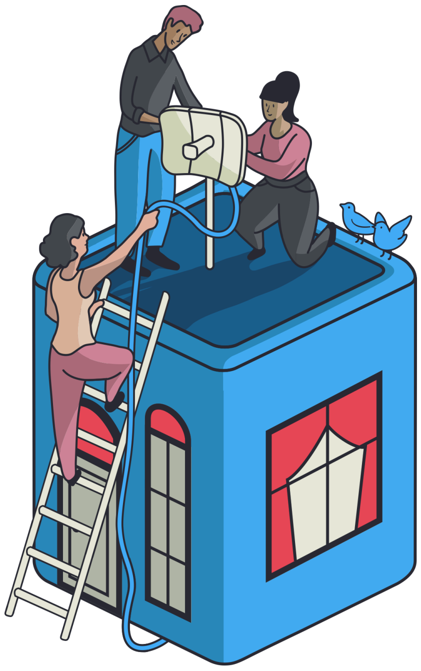
Peggy Testnet #1 Agenda

Buckle up, these tokens have places to be



PREPARED FOR

The Althea Community

PREPARED BY

Justin Kilpatrick, CTO

Why Testnet?

Testnet #1 will be the first long term test of the Peggy software in a distributed environment.

Like Brokennet this testnet will be launched with a four hour Zoom call with all participants online, from 9am to Noon PST and

The calendar event link is here, this includes the Zoom link.

The goal is to keep this testnet online for about two weeks, so that more extensive testing can be done and more problems found. Validators are advised to not concern themselves with any issues between the 23rd-27th. We will return with fixes to any breakage, or a celebratory status update if everything remains working on the 28th.

What are we testing?

Peggy is now functionally complete, so our plan is to send a lot of tokens back and forth until we break something. This test does not include slashing, key delegation, and genesis state saving for chain restarts.

1. Launch a Cosmos chain running the Peggy module (Stargate RC3 currently)
2. Deploy the Peggy Cosmos contract to the Rinkeby Ethereum testnet
3. Setup Peggy orchestrators on each validator and register them for use
4. Update the validator set on Rinkeby
5. Send ERC20 tokens to Cosmos and then back to Ethereum
6. (5) but at a larger scale
7. (5) but with significant validator state churn

What do I need?

A Linux server with any modern Linux distribution, 2gb of ram and at least 20gb storage. Requirements are very minimal.

Bootstrapping steps and commands

We're going to have a centralized start testnet. Where I will launch a chain, send everyone else tokens, and then each participant will come in and stake to become a validator.

In order to further simplify bootstrapping for this testnet we will be using pre-built binaries I am placing into a github release. These include ARM binaries for those of you on ARM platforms. Note that you will need to be running a 64bit ARM machine with a 64 bit operating system to use these binaries. In order to download ARM binaries change the names in the wget links from 'client' to 'arm-client'. Repeat for all binaries

Download the Peggy tools:

mkdir peggy-tools

cd peggy-tools

wget https://github.com/althea-net/peggy/releases/download/Testnet1/client
https://github.com/althea-net/peggy/releases/download/Testnet1/orchestrator
https://github.com/althea-net/peggy/releases/download/Testnet1/peggy
https://github.com/althea-net/peggy/releases/download/Testnet1/register-eth-key
https://github.com/althea-net/peggy/releases/download/Testnet1/relayer

chmod +x *

sudo mv * /usr/bin/

You may need to repeat this process if the release is updated

Actually joining the brokennet chain

So at this point you have everything that you'll need build and ready to go

Generate your keys:

Be sure to back up the phrase you get! You'll need it in a bit

cd $HOME

peggy init mymoniker --chain-id peggy-testnet1

The chain id is peggy-testnet1

peggy keys add validator

Copy the BrokenNet genesis file to:

wget
https://github.com/althea-net/peggy/releases/download/Testnet1/peggy-testnet1-genesis.json

cp peggy-testnet1-genesis.json $HOME/.peggy/config/genesis.json

Add persistent_peers:

Change the p2p.persistent_peers field in ~/.peggy/config/config.toml to contain the following:

```
persistent_peers =
“c074d2da2aad38907772e22e40a444c7e9ab3e2e@104.236.19.8:26656,737f401b6ed982bdd95568fd2232394a9c754a6a@peggy.technofractal.com:26657”
```

ed3125eb91d4e045ef030ca5

Start your full node:

Wait for it to sync up

```
peggy start
```

Request some tokens to be sent to your address / Paste in chat

Send your validator setup transaction:

```
peggy tx staking create-validator \
  --amount=1500000stake \
  --pubkey=$(peggy tendermint show-validator) \
  --moniker="put your validator name here" \
  --chain-id=peggy-testnet1 \
  --commission-rate="0.10" \
  --commission-max-rate="0.20" \
  --commission-max-change-rate="0.01" \
  --min-self-delegation="1" \
  --gas="auto" \
 --gas-adjustment=1.5 \
  --gas-prices="0.025stake" \
  --from=validator
```

Or if you need to change your stake **THIS IS OPTIONAL!**

```
peggy tx staking create-validator \
  --amount=1500000stake \
  --pubkey=$(peggy tendermint show-validator) \
  --moniker="put your validator name here" \
  --chain-id=peggy-testnet1 \
  --commission-rate="0.10" \
  --commission-max-rate="0.20" \
  --commission-max-change-rate="0.01" \
  --min-self-delegation="1" \
  --gas="auto" \
 --gas-adjustment=1.5 \
  --gas-prices="0.025stake" \
  --from=validator
```

Or to increase your stake **ALSO OPTIONAL!**

```
peggy keys show validator1 --bech val
```

```
peggy tx staking delegate <the valoperpub key from the first command> 99000000stake --from validator1 --chain-id peggy-testnet1 --fees 50stake --broadcast-mode block
```

Confirm that you are validating:

```
peggy query tendermint-validator-set | grep "$(peggy tendermint show-validator)"
```

Bootstrapping Peggy

Now that we've started a testnet we can get into the Peggy specific components. This guide does not include building or deploying the Peggy solidity contract simply because only one person needs to do it. On a real chain there would be a governance vote about what contract address to use, but that's not required here.

Edit your peggy config to enable the rpc:

```
vim $HOME/.peggy/config/app.toml
```

Go to the line for api configuration and set enable=true then restart your node

Register your Ethereum key:

Save the Ethereum key that this generates!

```
register-eth-key --cosmos-phrase="your phrase"
--cosmos-rpc="http://localhost:1317"--fees=footoken
```

Download Geth

```
wget
https://gethstore.blob.core.windows.net/builds/geth-linux-amd64-1.9.25-e7872729.tar.gz

tar -xvf geth-linux-amd64-1.9.25-e7872729.tar.gz

cd geth-linux-amd64-1.9.25-e7872729

./geth --syncmode "light" --rinkeby --http
```

Start your Orchestrator:

```
RUST_LOG=INFO orchestrator \

    --cosmos-phrase="{{COSMOS_MNEMONIC}}" \

    --ethereum-key="{{ETH_PRIV_KEY}}" \

    --cosmos-legacy-rpc="http://localhost:1317" \

    --cosmos-grpc="http://localhost:9090" \

    --ethereum-rpc="http://localhost:8545" \

    --fees=footoken \

    --contract-address="0xB411f2158e70414921BEA40bC3001F89F6595F22"
```

Testing Peggy

Run the peggy client:

This private key 0xb1bab011e03a9862664706fc3bbaa1b16651528e5f0e7fbfcbfdd8be302a13e7

Has millions of tokens in these ERC20 contracts on Rinkeby, have fun!

0x3A020A6A407d145de10De0367a767611F1652c06

0x95a76bC37Eca834143E61d9F8c8F32da01BdeA1B

0x8a0540d474E8D1a96D1c5e5a138232D83f19c6aF

Eth To Cosmos

```
RUST_LOG=info client eth-to-cosmos \
        --ethereum-key="your ethereum key" \
        --ethereum-rpc="http://localhost:8545" \
        --contract-address="0xB411f2158e70414921BEA40bC3001F89F6595F22" \
        --erc20-address="the erc20 contract your bridging" \
        --amount=10000000 \
        --cosmos-destination="your destination"
```

Cosmos to Eth

```
RUST_LOG=info client cosmos-to-eth \
        --cosmos-phrase="your phrase" \
        --cosmos-rpc="http://localhost:1317"  \
        --fees="peggy/<any of the above contract addresses>" \
        --erc20-address="<any of the above contract addresses>" \
        --amount=5000000 \
        --eth-destination="<your dest address>"
```

Systemd unit file for running peggy

To enable, copy the below to /etc/systemd/system/peggy.service :point_down:

```
[Unit]

Description=Peggy Node

After=network-online.target


[Service]

User=johnzampolin

ExecStart=/usr/bin/peggy start --pruning=nothing

Restart=always

RestartSec=3

LimitNOFILE=4096


[Install]

WantedBy=multi-user.target
```

Then you will need to reload systemd and start the service

```
Sudo systemctl daemon-reload

Sudo systemctl start peggy

# to see the logs

Journalctl -u peggy -f
```