Deploy a subnet

Tutorial to deploy your first custom IPC subnet Ready to test the waters with your first subnet? This guide will deploy a subnet with three local validators orchestrated byipc-cli . This subnet will be anchored to the publicCalibration testnet . This will be a minimal example and may not work on all systems. The full documentation provides more details on each step.

Several steps in this guide involve running long-lived processes. In each of these cases, the guide advises starting a newsession . Depending on your set-up, you may do this using tools likescreen ortmux , or, if using a graphical environment, by opening a new terminal tab, pane, or window.

Step 1: Prepare your system

Install the basic requirements for IPC:

Linux MacOS * Install system packages:sudo apt install build-essential clang cmake pkg-config libssl-dev protobuf-compiler git curl * . * Install Rust. Seeinstructions * . * Install cargo-make:cargo install --force cargo-make * . * Install Docker. Seeinstructions * . * Install Foundry. Seeinstructions * . *

Also install the following dependencies (details)

```

Copy sudo apt update && sudo apt install build-essential libssl-dev mesa-opencl-icd ocl-icd-opencl-dev gcc git bzr jq pkg-config curl clang hwloc libhwloc-dev wget ca-certificates gnupg -y

``` * Install Xcode from App Store or terminal:xcode-select --install * Install Homebrew. Seeinstructions * . * Install dependencies:brew install jq * Install Rust. Seeinstructions * . (if you have homebrew installed rust, you may need to uninstall that if you get errors in the build) * Install Cargo make:cargo install --force cargo-make * Install docker. Seeinstructions * . * Install foundry. Seeinstructions * . *

Building:

NOTE: this step may take a while to compile, depending on OS version and hardware build Linux MacOS ```

Copy

make sure that rust has the wasm32 target & use stable version of rustc

rustup target add wasm32-unknown-unknown rustup default stable

add your user to the docker group

sudo usermod -aG docker USER && newgrp docker

clone this repo and build

git clone https://github.com/consensus-shipyard/ipc.git cd ipc/contracts make gen cd .. cargo build --release

building will generate the following binaries

./target/release/ipc-cli --version ./target/release/fendermint --version

Copy

make sure that rust has the wasm32 target & use stable version of rustc

rustup target add wasm32-unknown-unknown rustup default stable

clone this repo and build

git clone https://github.com/consensus-shipyard/ipc.git cd ipc/contracts make gen cd .. cargo build --release

building will generate the following binaries

./target/release/ipc-cli --version ./target/release/fendermint --version

```

Step 2: Initialise your config

- Initialise the config
-

Linux/MacOS ```

Copy alias ipc-cli="cargo run -q -p ipc-cli --release --" ipc-cli config init

``` This should have populated a default config file with all the parameters required to connect to calibration at~/.ipc/config.toml . Feel free to update this configuration to fit your needs.

The IPC stack is changing rapidly. To make sure you use the latest contracts deployed on Filecoin Calibration:

- Runnano ~/.ipc/config.toml
- to see your configuration
-

```

Copy keystore_path = "~/.ipc"

[[subnets]] id = "/r314159"

[subnets.config] network_type = "fevm" provider_http = "https://api.calibration.node.glif.io/rpc/v1" gateway_addr = " registry_addr = ""

```

- Replace
- thegateway_addr
- andregistry_addr
- with the following values. Click on the badges below to take you to the source to copy and paste them or go to this link
- .
- ?
- ?
-

Step 3: Set up your wallets

Since we are setting up a subnet with multiple validators, we will create a set of wallets to spawn and interact within the subnet.

TIP: Note down wallet and subnet addresses and keys as you go along * Create four different wallets (we recommend a minimum of 4 for BFT security) *

```

Copy ipc-cli wallet new --wallet-type evm ipc-cli wallet new --wallet-type evm ipc-cli wallet new --wallet-type evm ipc-cli wallet new --wallet-type evm

```

- You can optionally set one of the wallets as your default so you don't have to use the--from
- flag explicitly in some of the commands:
-

```

Copy ipc-cli wallet set-default --address --wallet-type evm
```

```

- Go to theCalibration faucet
- and get some funds sent to each of your addresses
-

NOTE: you may hit faucet rate limits. In that case, wait a few minutes or continue with the guide and come back to this before step 9. Alternatively, you can send funds from your primary wallet to your owner wallets. TIP: If you'd like to import an EVM account into Metamask, you can use export the private key usingipc-cli wallet export --wallet-type evm --address

*. More information is available in theEVM IPC agent support docs .*

*Step 4: Create a child subnet*

- *The next step is to create a subnet under/r314159*
- *calibration. Remember to set a default wallet or explicitly specify the wallet from which you want to perform the action with the--from*
- *flag.*
-

```

Copy ipc-cli subnet create --parent /r314159 --min-validator-stake 1 --min-validators 4 --bottomup-check-period 300 --from-permission-mode collateral --supply-source-kind native

```

*This will output your subnet ID, similar to the following:*

```

Copy /r314159/t410fx2xy6x6idpy6yfywiilp6uitq4eerhpdr72wtmi

```

*Make a note of the address of the subnet you created because you will use it below.*

*Step 5: Join the subnet*

*Before we deploy the infrastructure for the subnet, we will have to bootstrap the subnet and join from our validators, putting some initial collateral into the subnet and giving our validator address some initial balance in the subnet. For this, we need to send ajoin command from each of our validators from their validator owner addresses providing their corresponding public key.*

- *Get the public key for all of your wallets and note it down. This is the public key that each of your validators will use to sign blocks in the subnet.*
-

```

Copy ipc-cli wallet pub-key --wallet-type evm --addressipc-cli wallet pub-key --wallet-type evm --address ipc-cli wallet pub-key --wallet-type evm --address ipc-cli wallet pub-key --wallet-type evm --address

```

- *Join the subnet with each validator.*
-

```

Copy ipc-cli subnet join --from=--subnet= --collateral=10 --public-key=--initial-balance 1 ipc-cli subnet join --from=--subnet= --collateral=10 --public-key= --initial-balance 1 ipc-cli subnet join --from=--subnet= --collateral=10 --public-key=--initial-balance 1 ipc-cli subnet join --from= --subnet= --collateral=10 --public-key=--initial-balance 1

```
```

Step 6: Deploy the infrastructure

First, we need to export the validator private keys for all wallets into separate files which we will use to set up a validator node.

```

*Copy ipc-cli wallet export --wallet-type evm --address--hex > ~/.ipc/validator_1.sk ipc-cli wallet export --wallet-type evm --address --hex > ~/.ipc/validator_2.sk ipc-cli wallet export --wallet-type evm --address --hex > ~/.ipc/validator_3.sk ipc-cli wallet export --wallet-type evm --address --hex > ~/.ipc/validator_4.sk*

```

Let's start our first validator which the rest of the validators will bootstrap from. Make sure you have docker running before running this command.

```

*Copy cargo make --makefile infra/fendermint/Makefile.toml \ -e NODE_NAME=validator-1 \ -e PRIVATE_KEY_PATH=\ -e SUBNET_ID= \ -e CMT_P2P_HOST_PORT=26656 \ -e CMT_RPC_HOST_PORT=26657 \ -e ETHAPI_HOST_PORT=8545 \ -e RESOLVER_HOST_PORT=26655 \ -e PARENT_GATEWAY=curl -s https://raw.githubusercontent.com/consensus-shipyard/ipc/cd/contracts/deployments/r314159.json | jq -r '.gateway_addr' \ -e PARENT_REGISTRY=curl -s https://raw.githubusercontent.com/consensus-shipyard/ipc/cd/contracts/deployments/r314159.json | jq -r '.registry_addr' \ -e FM_PULL_SKIP=1 \ child-validator*

```

Once the first validator is up and running, it will print out the relative information for this validator.

TIP: Highly recommend documenting that information which will be useful to bootstrap other validators, connect to the IPC subnet on MetaMask, etc. ```

Copy

Subnet node ready!

Subnet ID: /r314159/t410f6b2qto756ox3qfoonq4ii6pdrylxwyretgpixuy

Eth API: http://0.0.0.0:8545

Chain ID: 3684170297508395

Fendermint API: http://localhost:26658

CometBFT API: http://0.0.0.0:26657

CometBFT node ID: ca644ac3194d39a2834f5d98e141d682772c149b

CometBFT P2P: http://0.0.0.0:26656

IPLD Resolver Multiaddress: /ip4/0.0.0.0/tcp/26655/p2p/16Uiu2HAkwhrWn9hYFQMR2QmW5Ky7HJKSGVkT8xKnQr1oUGCkqWms

```

*You'll need the final component of theIPLD Resolver Multiaddress (thepeer ID ) and theCometBFT node ID for the next nodes to start.*

- *BOOTSTRAPS*
- *:@validator-1-cometbft:26656*
- ```
- *Copy*
- *// An example*
- *ca644ac3194d39a2834f5d98e141d682772c149b@validator-1-cometbft:26656*
```

- ```
- *RESOLVER_BOOTSTRAPS*
- *: /dns/validator-1-fendermint/tcp/26655/p2p/*
- ```
- *Copy*
- *// An example*
- */dns/validator-1-fendermint/tcp/26655/p2p/16Uiu2HAkwhrWn9hYFQMR2QmW5Ky7HJKSGVkT8xKnQr1oUGCkqWms*
- ```
-

Now, run the 2nd validator in a separate terminal.

```

*Copy cargo make --makefile infra/fendermint/Makefile.toml \ -e NODE_NAME=validator-2 \ -e PRIVATE_KEY_PATH=\ -e SUBNET_ID= \ -e CMT_P2P_HOST_PORT=26756 \ -e CMT_RPC_HOST_PORT=26757 \ -e ETHAPI_HOST_PORT=8645 \ -e RESOLVER_HOST_PORT=26755 \ -e BOOTSTRAPS=@validator-1-cometbft:26656 \ -e RESOLVER_BOOTSTRAPS=/dns/validator-1-fendermint/tcp/26655/p2p/ \ -e PARENT_GATEWAY=curl -s https://raw.githubusercontent.com/consensus-shipyard/ipc/cd/contracts/deployments/r314159.json | jq -r '.gateway_addr' \ -e PARENT_REGISTRY=curl -s https://raw.githubusercontent.com/consensus-shipyard/ipc/cd/contracts/deployments/r314159.json | jq -r '.registry_addr' \ child-validator*

```

Now, the 3rd:

```

*Copy cargo make --makefile infra/fendermint/Makefile.toml \ -e NODE_NAME=validator-3 \ -e PRIVATE_KEY_PATH=\ -e SUBNET_ID= \ -e CMT_P2P_HOST_PORT=26856 \ -e CMT_RPC_HOST_PORT=26857 \ -e ETHAPI_HOST_PORT=8745 \ -e RESOLVER_HOST_PORT=26855 \ -e BOOTSTRAPS=@validator-1-cometbft:26656 \ -e RESOLVER_BOOTSTRAPS=/dns/validator-1-fendermint/tcp/26655/p2p/ \ -e PARENT_GATEWAY=curl -s https://raw.githubusercontent.com/consensus-shipyard/ipc/cd/contracts/deployments/r314159.json | jq -r '.gateway_addr' \ -e PARENT_REGISTRY=curl -s https://raw.githubusercontent.com/consensus-shipyard/ipc/cd/contracts/deployments/r314159.json | jq -r '.registry_addr' \ child-validator*

```

And finally, the 4th:

```

*Copy cargo make --makefile infra/fendermint/Makefile.toml \ -e NODE_NAME=validator-4 \ -e PRIVATE_KEY_PATH=\ -e SUBNET_ID= \ -e CMT_P2P_HOST_PORT=26956 \ -e CMT_RPC_HOST_PORT=26957 \ -e ETHAPI_HOST_PORT=8845 \ -e RESOLVER_HOST_PORT=26955 \ -e BOOTSTRAPS=@validator-1-cometbft:26656 \ -e RESOLVER_BOOTSTRAPS=/dns/validator-1-fendermint/tcp/26655/p2p/ \ -e PARENT_GATEWAY=curl -s https://raw.githubusercontent.com/consensus-shipyard/ipc/cd/contracts/deployments/r314159.json | jq -r '.gateway_addr' \ -e PARENT_REGISTRY=curl -s https://raw.githubusercontent.com/consensus-shipyard/ipc/cd/contracts/deployments/r314159.json | jq -r '.registry_addr' \ child-validator*

```

NOTE:

- *Use full path to PRIVATE_KEY_PATH, don't path with "~"*
- *Do not change values of any port from the ones provided unless you have to*
- *If you are deploying all validators on a single server, ports will need to be different, as shown in above examples. If you are deploying them from different servers, the ports can be similar. **

Step 7: Interact with your subnet using the IPC CLI

- *Make sure~/.ipc/config.toml*
- *contains the configuration of your subnet in the "Subnet template" section. Uncomment the section and populate the*

corresponding fields

-

```

*Copy*

# *Subnet template - uncomment and adjust before using*

*[[subnets]] id =*

*[subnets.config] network_type = "fevm" provider_http = "http://localhost:8545/" gateway_addr = "0x77aa40b105843728088c0132e43fc44348881da8" registry_addr = "0x74539671a1d2f1c8f200826baba665179f53a1b7"*

```

*NOTE: The ETH addresses forgateway_addr andregistry_addr used when they are deployed in genesis in a child subnet by Fendermint are0x77aa40b105843728088c0132e43fc44348881da8 and0x74539671a1d2f1c8f200826baba665179f53a1b7 , respectively, so no need to change them. * Fetch the balances of your wallets using the following command. The result should show the initial balance that you have included for your validator address in genesis: **

```

*Copy ipc-cli wallet balances --wallet-type evm --subnet=*

```

Step 8: Run a relayer

IPC relies on the role of a specific type of peer on the network called the relayers that are responsible for submitting bottom-up checkpoints that have been finalized in a child subnet to its parent.

This process is key for the commitment of child subnet checkpoints in the parent, and the execution of bottom-up cross-net messages. Without relayers, cross-net messages will only flow from top levels of the hierarchy to the bottom, but not the other way around.

- *Run the relayer process passing the 0x address of the submitter account:*
-

```

*Copy ipc-cli checkpoint relayer --subnet--submitter*

```

*Previous Architecture Next Subnets Last updated1 day ago On this page *Step 1: Prepare your system * Step 2: Initialise your config * Step 3: Set up your wallets * Step 4: Create a child subnet * Step 5: Join the subnet * Step 6: Deploy the infrastructure * Step 7: Interact with your subnet using the IPC CLI* Step 8: Run a relayer*

Was this helpful? Edit on GitHub