

Deploy an OP Stack devnet

This guide will show you how to run your own OP Stack devnet.

Dependency setup

Environment setup and Golang installation

First, [install dependencies for Celestia software](#) and for [OP Stack](#) .

Clone repository

Next, clone the repo:

```
bash cd HOME git
```

```
clone
```

```
https://github.com/celestiaorg/optimism cd
```

```
optimism cd HOME git
```

```
clone
```

```
https://github.com/celestiaorg/optimism cd
```

```
optimism Check out to the version for either the stable version or upstream version:
```

```
v1.2.0-OP_v1.7.0-CN_v0.12.4
```

```
rebase-upstream bash git
```

```
checkout
```

```
tags/v1.2.0-OP_v1.7.0-CN_v0.12.4 git
```

```
submodule
```

```
update
```

```
--init
```

```
--recursive git
```

```
checkout
```

```
tags/v1.2.0-OP_v1.7.0-CN_v0.12.4 git
```

```
submodule
```

```
update
```

```
--init
```

```
--recursive bash git
```

```
checkout
```

```
rebase-upstream git
```

```
submodule
```

```
update
```

```
--init
```

```
--recursive git
```

```
checkout
```

```
rebase-upstream git
```

```
submodule
```

```
update
```

```
--init
```

```
--recursive
```

Build devnet

Build TypeScript definitions for TS dependencies:

```
bash cd HOME cd
```

```
optimism make cd HOME cd
```

```
optimism make Set environment variables to start network:
```

```
bash export SEQUENCER_BATCH_INBOX_ADDRESS = 0xff00000000000000000000000000000000 export L2OO_ADDRESS = 0x70997970C51812dc3A010C7d01b50e0d17dc79C8 export SEQUENCER_BATCH_INBOX_ADDRESS = 0xff00000000000000000000000000000000 export L2OO_ADDRESS = 0x70997970C51812dc3A010C7d01b50e0d17dc79C8
```

Start devnet

Start the network by running:

```
bash make
```

```
devnet-up make
```

devnet-up This starts up the layer 1 (ETH), layer 2 (op-geth), data availability layer (Celestia), the sequencer (op-node), batch submitter (op-batcher), state commitment service (op-proposer).

View the logs of the devnet

If you'd like to view the logs of the devnet, run the following command from the root of the Optimism directory:

```
bash make
```

```
devnet-logs make
```

devnet-logs Optional: Docker tips

Stop devnet

devnet-clean

To view the containers running, send:

```
bash docker
```

ps docker

ps Find the container ID of theops-bedrock_op-batcher_1 and run the following to follow the logs:

```
bash docker
```

logs

-f

< container-i d

docker

logs

-f

< container-i d

In a new terminal, find the container ID of `theghr.io/celestiaorg/local-celestia-devnet:main` and run the following to follow the logs:

```
bash docker
```

logs

-f

< container-i d

docker

logs

-f

```
< container-i d
```

You can do the same for other containers as you explore the stack.

Now, we'll check for a recent transaction on the L1 with:

bash cast

block

latest

```
--rpc-url
```

localhost:8545 cast

block

latest

```
--rpc-url
```

localhost:8545 Output of a block that contains a transaction will look like this:

[illegible]

```
bash export TX_HASH = 0x79a0a7a1b4936aafe7a37dbfb07a6a9e55c145a4ed6fd54f962649b4b7db8de7 export TX_HASH = 0x79a0a7a1b4936aafe7a37dbfb07a6a9e55c145a4ed6fd54f962649b4b7db8de7
```

Now read the transaction call data on the L1:

bash cast

tx TX HASH --rpc-url

localhost:8545 cast

tx TX HASH --rpc-url

localhost:8545 The output will look similar to below:

You are looking for a batcher transaction to the address `0xf0000000000000000000000000000000901`. First, remove the prefix `0xce`. Now, set the input as the `INPUT` variable and encode it as base64:

Remember to remove the 0xce prefix!

Clone thego-da repository:

```
console { "blobs": [ { "value":
  "AKUmhJ8FnuyVBs38FDKEIAAAAAZB42trhw/DDc4GFAlv4klkv5Zh4E16mmO5fpNOS1f5wzpbs8YK3S0Rvs4ULLj13euw+Ovddv6Q23zuV1ShROEvk5aptIT7bGmZunvc1OiKwJTXVbN0BiGm6k2zNM
  " } ] } { "blobs": [ { "value":
  "AKUmhJ8FnuyVBs38FDKEIAAAAAZB42trhw/DDc4GFAlv4klkv5Zh4E16mmO5fpNOS1f5wzpbs8YK3S0Rvs4ULLj13euw+Ovddv6Q23zuV1ShROEvk5aptIT7bGmZunvc1OiKwJTXVbN0BiGm6k2zNM
  " } ] }
```

The [Ethereum fallback mechanism](#) allows rollups to "fall back" to Ethereum or another EVM chain in the case of downtime or errors submitting data to Celestia.

The Ethereum fallback mechanism is implemented in the [celestiaorg/optimism](https://github.com/celestiaorg/optimism) v1.1.0 release.

Theop-batcher/batcher/driver.go andop-node/rollup/derive/calldata_source.go files are part of the Ethereum fallback mechanism in theop-batcher andop-node respectively.

`driver.go.thecaldataTxCandidate` function is responsible for the `write path` of the Ethereum fallback. This function creates and submits a transaction to the batch inbox address with the given data. It uses the underlying `txmgr` to handle transaction sending and gas price management.

If the transaction data can be published as a blob to Celestia, it replaces the calldata with a blob identifier and sends the transaction with this data. If it cannot be published to Celestia, it falls back to Ethereum without any change to the transaction.

The blob identifier starts with the special prefix `0xce`, which was chosen a mnemonic for Celestia, and indicates that the remaining data has to be interpreted as a little-endian encoded Block Height (8 bytes) and Blob Commitment (32 bytes). The combination of these can later be used to retrieve the original calldata from Celestia.

```
Prefix 8 bytes 32 bytes 0xce Block Height Blob Commitment go func (l * BatchSubmitter) sendTransaction ( txdata txData, queue * txmgr.Queue[txData], receiptsCh chan txmgr.TxReceipt[txData], ) { //
... } func (l * BatchSubmitter) sendTransaction ( txdata txData, queue * txmgr.Queue[txData], receiptsCh chan txmgr.TxReceipt[txData], ) { // ... } In calldata_source.go, theDataFromEVMTransactions
function defines the read path of the Ethereum fallback. This function filters all of the transactions and returns the calldata from transactions that are sent to the batch inbox address from the batch
sender address.
```

If the calldata matches the version prefix `0xce`, it is decoded as a blob identifier, the original calldata is retrieved from Celestia and returned for derivation. If the calldata does not match the prefix, the

entire calldata is returned for derivation.

go func

```
DataFromEVMTransactions ( config * rollup.Config, batcherAddr common.Address, txs types.Transactions, log log.Logger ) ([]eth.Data, error ) { // ... } func
```

DataFromEVMTransactions (config * rollup.Config, batcherAddr common.Address, txs types.Transactions, log log.Logger) ([]eth.Data, error) { // ... } These two functions work together to ensure that the Ethereum fallback mechanism operates correctly, allowing the rollup to continue functioning even during periods of downtime on Celestia.

Testing the fallback

Testing out the Ethereum fallback mechanism can be done with thego-da tool. Triggering a simultaneous blob transaction will cause theop-batcher blob transaction to fail, with anincorrect account sequence error, which triggers a fallback to Ethereum.

To trigger the transaction, send this command from the samego/proto/da directory:

```
bash grpcurl
```

```
-proto
```

```
da.proto
```

```
-plaintext
```

```
-d
```

```
'{"blobs": [{"value": "SGVsbG8gd28ybGQh"}]}'
```

```
127.0.0.1:26650
```

```
da.DAService.Submit grpcurl
```

```
-proto
```

```
da.proto
```

```
-plaintext
```

```
-d
```

```
'{"blobs": [{"value": "SGVsbG8gd28ybGQh"}]}'
```

```
127.0.0.1:26650
```

da.DAService.Submit Alternatively, you can shut off thelocal-celestia-devnet and see that the OP Stack devnet logs show that the rollup has fallen back to the L1, in this case Ethereum, for posting data.

Span batches

Span batches can be enabled by settingOP_BATCHER_BATCH_TYPE: 1 in yourdocker-compose.yml file.

Note that this requires the Delta activation time to be configured. For your devnet, you should set"l2GenesisDeltaTimeOffset": "0x0", indevnetL1-template.json . This will enable span batches and can be tested by greppingdocker compose logs -f | grep batch_type which should includebatch_type=SpanBatch andbatch_type=1 . [\[Edit this page on GitHub\]](#) Last updated: [Previous page Deploy a dapp on Bubs testnet](#) [Next page Deploy an OP Stack devnet on Celestia](#) [\[\]](#)