Deploy Arbitrum Orbit with Avail DA

This guide follows the deployment of an Arbitrum Orbit chain (L3) with Arbitrum Sepolia (L2) as the base chain. You can follow the same steps with anyother supported chains with the relevant contract addresses.

This guide is based onArbitrum Orbit Quickstart(opens in a new tab)

Prerequisites

Docker

- You need to havedocker(opens in a new tab)
- anddocker-compose(opens in a new tab)
- installed on your machine.

Avail

- An Avail account with some AVAIL tokens in it. You can refer toour docs here
- to get started.
- An application key (app-id
-) for your rollup. Learnhow to get yourapp-id here
- .

Arbitrum Sepolia

- At least 1 ETH over Arbsepolia Testnet. You can followStep 1(opens in a new tab)
- on Arbitrum Orbit Doc for acquiring testnet ETH
- This guide will useArbitrum's ArbSepolia(opens in a new tab)
- andAvail's Turing
- testnets.

Download avail-nitro-node docker image

Step-1: Download theavail-nitro-node(opens in a new tab)

image from docker hub

docker

pull

availj/avail-nitro-node:v2.2.1-upstream-v3.2.1

Deploy Rollup Contracts

Step-1: Clone thearbitrum-orbit-sdk(opens in a new tab)

with AvailDA

git

clone

https://github.com/availproject/arbitrum-orbit-sdk.git git

checkout

avail-develop-upstream-v0.20.1 cd

arbitrum-orbit-sdk/examples/create-avail-rollup-eth

Step-2: Create.env

file

Create a copy of.env.sample as.env file and edit the following:

- Set EVM account keys of settlement layer
- DEPLOYER_PRIVATE_KEY
- :-
- Orbit
- rollup
- owner,
- Rollup
- contract
- deployer
- BATCH_POSTER_PRIVATE_KEY
- -:
- Batch
- poster
- VALIDATOR_PRIVATE_KEY
- :-
- Validator
- Set avail account info for DA submissions
- AVAIL_ADDR_SEED
- =
- AVAIL_APP_ID
- =
- Set Fallback S3, if you want to enable S3 storage as a fallback for AvailDA (its an optional feature)
-
    - Fill these required details where makeFALLBACKS3_ENABLE = true
- FALLBACKS3_ENABLE
- =
- true
- FALLBACKS3_ACCESS_KEY
- =
- FALLBACKS3_SECRET_KEY
- =
- FALLBACKS3_REGION
- =
- FALLBACKS3_OBJECT_PREFIX
- =
- FALLBACKS3_BUCKET
- =
- Set parent chain rpc URL (other than infura will also works)
- PARENT_CHAIN_RPC
- =
- https://arbitrum-sepolia.infura.io/v3/YOUR_API_KEY

Step-3: Rollup contract deployment

- Run avail orbit rollup deployment command
- npm
- run
- deploy-avail-orbit-rollup
- You will find this kind of log
- You will get to see two files addednodeConfig.json
- andorbitSetupScriptConfig.json
- , which will be need in next step spin and set up avail orbit rollup

Before proceeding, let's briefly review what just happened:

1. You submitted a deployment transaction to an Orbit "factory" smart contract on the Arbitrum Sepolia testnet, the public L2 chain that your local Orbit chain will settle transactions to.
2. This Orbit smart contract then initialized your Orbit chain's base contracts with the values that you specified in the previous step, and deployed these base contracts to the Arbitrum Sepolia testnet.

Your Orbit chain's base contracts are responsible for facilitating the exchange of information between your chain's node(s) and its base chain's nodes. This includes the batch posting of transactions from your Orbit chain to its base chain, the staking of tokens by your Orbit chain's validators the challenge mechanism, bridging mechanisms, and more.

Spin up the avail orbit rollup chain using orbit-setup-script

Step-1: Downloadorbit-setup-script(opens in a new tab)

git

clone

https://github.com/availproject/orbit-setup-script.git cd

orbit-setup-script

Step-2 :- Run your chain

Add thenodeConfig.json andorbitSetupScriptConfig.json in./config , that you get after completing Deploy Rollup contracts

cd

orbit-setup-script docker-compose

up

-d A Nitro node and BlockScout explorer instance will be started. Visithttp://localhost:4000/ (opens in a new tab) to access your BlockScout explorer instance - this will allow you to view your chain's transactions and blocks, which can be useful for debugging.

Step-3: Final Setup - Token bridge deployment

A Hardhat script that handles the following tasks:

1. Fund thebatch-poster
2. andvalidator
3. (staker) accounts on your underlying L2 chain.
4. Deposit ETH into your account on the chain using your chain's newly deployed bridge.
5. Deploy your Token Bridge contracts on both L2 and local Orbit chains.
6. Configure parameters on the chain.

To run this script, issue the following command from the root of theorbit-setup-script repository, replacingOxYourPrivateKey with the private key of theOwner account you used to deploy your chain's contracts, and replacinghttp://localhost:8449 with the RPC URL of your chain's node.

PRIVATE_KEY

"0xYourPrivateKey" L2_RPC_URL = "https://sepolia-rollup.arbitrum.io/rpc" L3_RPC_URL = "http://localhost:8449"

yarn

run

setup

Hurray

Congratulations, Your local Orbit chain with Avail DA is now running.

Appendix A:- Use Arbitrum bridge for your local chain

After completing final setup, you can find two filesnetwork.json andoutputInfo.json . Add in your avail orbit rollup testnet to thebridge(opens in a new tab) . This will only be for local testing. PasteoutputInfo.json configuration in setting section of bridge

Appendix B: Logging

Run this command in the root directory of your orbit setup script repo to view your chain's logs:

docker-compose

logs

-f

nitro

Appendix C: Depositing ETH/native token

If you need to deposit more ETH (or native tokens) into your Orbit chain account, run this command on the base directory of the setup script, replacing0xYourPrivateKey with the private key of the originating account, and with the amount to send:

Using Arbitrum Sepolia:

PRIVATE_KEY

"0xYourPrivateKey" L2_RPC_URL = "https://sepolia-rollup.arbitrum.io/rpc" L3_RPC_URL = "http://localhost:8449" AMOUNT = ""

yarn

run

deposit

Appendix D: Refunding (For testing environments)

Once you're done with your local Orbit chain and want to refund any remaining balance from the batch poster and validator wallets, you can use the refund script by running:

PRIVATE_KEY

"0xYourPrivateKey" L2_RPC_URL = "https://sepolia-rollup.arbitrum.io/rpc" TARGET_ADDRESS = "0xYourTargetAddress"

yarn

run

refund Make sure to replace"0xYourPrivateKey" with the private key of the wallet you used to deploy the rollup contracts, and"0xYourTargetAddress" with the address where you want to receive the refunded funds. This will transfer any remaining balance from the batch poster and validator wallets to the specified target address.

Appendix E: Troubleshooting

You may see error getting latest batch count in your node's output logs (from Appendix B). This is usually safe to ignore. It's usually displayed when your Orbit chain's base contract deployment isn't yet finalized on the L1 chain. This finalization can take 15-20 minutes, but don't worry - the deployment doesn't need to be L1-finalized in order for your chain to function properly.

Learn more about customizing and using your Orbit chain from theArbitrum Orbit Docs(opens in a new tab).

Reference

Currently chains supporting Arbitrum orbit chain deployment with AvailDA:-

- Arbitrum Sepolia(opens in a new tab)
- Base Sepolia(opens in a new tab)

Soon will add more chains

Smart contract addresses

The following information may be useful to those building on Arbitrum. We list the addresses of the smart contracts related to the protocol

Arbitrum Sepolia(opens in a new tab) Base Sepolia(opens in a new tab) RollupCreator 0xE917553b67f630C3982236B6A1d7844B1021B909 0x66667c90de1E169e705F248614c72076aA1a2240 Bridge 0x4A9722A440506469CcA933969f790923Ac59A0A5 0x6a7a97E34E6D75724BB58C0d8EFF8cEB27b23A56 AvailDABridge 0x5EE5FAd194260bF039b32DEe0bC3D809989A3d00 0x36cE36dc5BE3E89F6B25315d8346503B307546A6 SequencerInbox 0xA04d516D245D1E11Eb7cF8D38Ec1540A25c2DCB5 0x7C44660caB5b6d330cd38a9468F3F84F2f175E9C Inbox 0x790d2BCD12Fa25db147fc033515CBCEc89d89E20 0x48dCee9af45842A905FDE5ff5a502EFd06862E49 RollupEventInbox 0x62F82278559e8aBdCe222D1A4c2D45dFD6D8C26E 0x1cd6c66A1C6b260147909bbd3211Fb9f4424A480 Outbox 0xdd414356a4D573865D78D0C5E05d62686c14E376

0xdB991f2943CD1A3c17FDb5b78434F4b7331bFb37 ERC20Bridge 0x6582b333408D8eF71a951a77E61a263857cd570C 0x3d238cf78f740E94753254f9B6339ACaa66c1574 ERC20SequencerInbox 0xA878d9432DaCCb14A94EAcB8f6158954e538874a 0xE6Cb93cA92f1C65D5709059456ad232721A87c02 ERC20Inbox 0x7484e03fF084Fd9C1Bc30de40D7A2ed4acb22AFC 0xa5E5c409f39e929087DFbb182C29FE8350Ad1bC8 ERC20RollupEventInbox 0xE26d16aAb33950C7bEEcf0F46C161bAc0495dB00 0x09Cb2789481759613886b5747c5fE49FAb7c4650 ERC20Outbox 0xC313E4F98C525b555A972a564A01bdB76D69BA10 0xa183436DAb3ea069010912e0beBB4c734A4aD90c BridgeCreator 0xe188683D96b82a3A1314424E5e9c438461667EdE 0xE6DeCD823d1480069348931833F97C3810ea7Ea5 OneStepProver0 0x6404d3539B6d37ebc9AC50d7f528Ee8168ece5cC 0x6032313a93F78b9Db91885E6d83e40626f371052 OneStepProverMemory 0x71FF43aeb6E788B855410447Bbc1F71916Bb98Ec 0x5AFee8C856386Ac18E56CdA14470637e13C1D47b OneStepProverMath 0xcCe345C15E7a9AA7784601e28811671cf41bF711 0x91EF8ffce4AbB1b256662A6a6C4C7051534aD575 OneStepProverHostIo 0x65DCCc4bA7836B743D2370A5fbea8Ef3263789d9 0x49Fb114B5c708eF2a5aa6Ba413da55E5F07389C4 OneStepProofEntry 0x7452e148a7faffac053f442080DE8f305d0C8642 0xd0A9C68313233C568AD19B55FECac9148245Fb76 ChallengeManager 0x1c8285a7BFc8E5F4960aB105ca0E595Ce80858b2 0x71BF09F4787187b3c62dA0a89c893Ca5687ffef0 RollupAdminLogic 0x0922ce6b911B420EBC5851943Adfba40587B6852 0x488088F12D354b2bEe8c5d5C27Dd266f0b70c5eb RollupUserLogic 0x256E8dD36893f00E6C0c1B1099561b8322692B91 0x3A12b3eB213CAb423324400D49Ec7d61c903ac20 UpgradeExecutor 0x0c78cDc973de62ae83D99cEe87D8bD45d5Cd2084 0x2F6DdA8EA7b41186c1C00980fF1b96df23E4D442 ValidatorUtils 0x0911837C1cdC5b5C6ACDBc887Be223A05c3c6FDD 0xCa56EB65F0C84e2BeCCd6A42Ff2ADAC5e362705e ValidatorWalletCreator 0x367Ad15A9aDFF8C2D1D51892F2d717bF2B86a3aD 0xD3E785B544bf85f124fa0f3aC557127882Ed2f09 DeployHelper 0xD58aa678074840c5A74c29d85A493723c4f48e8c 0xaD98D017aA12d0a7A265d4BFAfd289CE9968943f VectorX 0xA712dfec48AF3a78419A8FF90fE8f97Ae74680F0 0x13cBE46E168460a101c07efb6Ab7B9ec637F02aA Avail Bridge 0x420953898afC97147675F793a90d3793b974698b 0xBC3071f9dE256B3b9C6304731F28C98f102425De Avail Bridge Proxy 0xa3e9e20875b7A7461C4829663497F7c2baB9E13b 0x9378F9E35ff46a131406FEb7a226F7da4395DB48

Previous Releases

v2.1.0-upstream-v3.1.0

- nitro-node:avail-nitro-node:v2.1.0-upstream-v3.1.0(opens in a new tab)
- nitro-contracts:avail-nitro-contracts:v2.1.0-upstream-v2.0.0(opens in a new tab)
- Rollup creator address:* Arbitrum Sepolia:0x9673b69f55808c4eB56b4e5E29ec1aE85e7f20f4
-
    - Base Sepolia:0xfB3b37b5685340169de1BcEc5d607912eBbB6668
- wasm module root:0x29e13d2cf8e8e44b8306703e845e0b9ee732f5144da4666a0079e9f50aec21aa

v2.0.0-upstream-v2.3.1

- nitro-node:avail-nitro-node:v2.0.0-upstream-v2.3.1(opens in a new tab)
- nitro-contracts:avail-nitro-contracts:v2.0.0-upstream-v1.2.1(opens in a new tab)
- Rollup creator address:* Arbitrum Sepolia:0x364DE2F72f1496Fdf02c133A5fe60e6a079f2178
-
    - Base Sepolia:0x246A71eefDBd543D015dbE8CAe52263103d49315
- wasm module root:0x9c5556e303c5b0399bde1391cbc61a195c95318e1c1a15b1c402d6a89ba1ebb3

v1.0.0

⚠ The previous release of v1 is not recommended for install any more, but we will continue supporting it until a migration path from v1 to current release is provided. * nitro-node:avail-nitro-node:v1.0.0-upstream-v2.3.1(opens in a new tab)* nitro-contracts:avail-nitro-contracts:v1.0.0-upstream-v1.2.1(opens in a new tab)* Rollup creator address: *Arbitrum Sepolia:0xADCBE5c221566FA6e6Ebf5BA08759c83177DfCDA * wasm module root:* arm64:-0x30d81f66d40dbb88484524cbce0982ee9adee944d9416c283313560efd5af958 * * amd64:-0xc3d6717b86468b20ed2d74fa4484571b51595d56294d154b0fac19d55b599e09

Overview Validium