Quickstart: Launch an Orbit chain

This quickstart is for developers who want to launch their own Arbitrum Orbit chain using the Orbit chain deployment portal .

By the end of this quickstart, you'll have a local devnet chain that hosts EVM-compatible smart contracts. Your chain will process transactions locally while settling to the public Arbitrum Goerli or Sepolia testnets . Familiarity with Ethereum, Ethereum's testnets, and Arbitrum is expected.

If you're looking for a conceptual introduction to Orbit chains, see the Gentle introduction to Orbit chains .

PUBLIC PREVIEW, MAINNET READY Orbit chains are now Mainnet ready ! Note that Orbit is still a public preview capability - the Orbit product and its supporting documentation may change significantly as we capture feedback from readers like you.

To provide feedback, click the Request an update button at the top of this document, join the Arbitrum Discord , or reach out to our team directly by completing this form .

Prerequisites

- Docker
- A browser-based Ethereum wallet (like MetaMask
-)
- At least 1 testnet ETH (for custom gas token chains, 0.6 ETH and 0.4 native tokens)

Step 1: Acquire Arbitrum Testnet ETH (and the native token for Orbit chains with custom gas tokens)

You'll need at least 1 testnet ETH for regular Orbit chains or 0.6 ETH plus 0.4 of your desired native token for Orbit chains with a custom gas token. The funds are needed to cover the cost of deploying your Orbit chain's base contracts to its base chain (Arbitrum Goerli or Sepolia). Sepolia is our recommendation as Goerli will be deprecated in the near future.

At the time of this quickstart's writing, the easiest way to acquire ETH is to bridge testnet ETH from Ethereum's L1 Goerli or Sepolia network to Arbitrum's corresponding L2 testnet:

1. Use an L1 testnet ETH faucet like goerlifaucet.com
2. or sepoliafaucet.com
3. to acquire some testnet ETH on an L1 testnet.
4. Bridge your L1 testnet ETH into Arbitrum L2 using the Arbitrum bridge
5. .

Step 2: Choose your chain type: AnyTrust or Rollup

Arbitrum Rollup is an Optimistic Rollup protocol; it's trustless and permissionless. These properties are achieved by requiring all chain data to be posted on Ethereum's "Layer 1" (L1) chain. This means the availability of this data follows directly from the security properties of Ethereum itself, and, in turn, that any party can participate in validating the chain and ensuring its safety.

By contrast, Arbitrum AnyTrust introduces a trust assumption in exchange for lower fees. Instead of requiring all Arbitrum nodes to have access to the data of every L2 transaction in the Arbitrum chain's inbox, AnyTrust relies on an external Data Availability Committee to store data and provide it on demand, reducing the costs of batching and posting all L2 transaction data to L1. Visit our FAQ docs to learn more.

We generally recommend using Rollup chains for use-cases that demand the highest level of security (like decentralized exchanges and other DeFi dApps). AnyTrust chains are suitable for use-cases that require lower fees and generate lots of transactions (like gaming and social dApps).

Step 3: Configure your Orbit chain's deployment

Visit the Orbit chain deployment portal . You'll be prompted to connect your wallet. You may be prompted to add the Arbitrum Goerli or Arbitrum Sepolia network to your wallet and/or to switch your wallet to this network; approve this.

The deployment portal will then display a form that looks like this:

Chain ID Chain name Challenge period (blocks) Stake token Base stake Owner The below table provides a brief description of each of these configuration parameters. We recommend sticking to the defaults; to learn more about customizing your Orbit chain's deployment configuration, visit How (and when) to customize your Orbit chain's deployment config :

Parameter Description Chain ID A unique integer identifier that represents your chain's network. YourChain ID can be submitted to chain indexes likeChainlist.org . For devnets, this is randomly generated for each deployment - don't worry about it for now. Chain name A human-readable way to distinguish your Orbit chain from other Orbit chains. Users, developers and the wider community will refer to your chain by yourChain name and/or yourChain ID . Challenge period (blocks) The amount of time that your Orbit chain's nodes have to dispute the current state of the chain before it's confirmed (and ultimately finalized) on the underlying L2 chain (Arbitrum Goerli or Sepolia). Note that this refers to the number of blocks on the underlying L1 chain (Ethereum's Goerli or Sepolia chain). Stake token The token that your chain's validators must stake in order to participate in your chain. This is hardcoded to ETH for now, but future versions of Orbit chains will let you specify an arbitrary ERC-20 token contract here. Base stake The amount of your configuredStake token that your chain's validators must stake in order to participate in your chain. Should be greater than 0. Owner The administrative Ethereum address that will deploy, own, and update your chain's base contracts. This will default to your connected wallet's address. This needs to be a standard Ethereum wallet account - an EOA, not a contract address. Note that you'll have to specify this wallet's private key within a local JSON file later. Gas token The address of the ERC-20 token on the parent chain that is intended to be used as the native gas token on the Orbit chain. This token must already be deployed natively on the parent chain and is bridged to the Orbit chain during chain deployment. This feature is only supported on AnyTrust chains currently, and more information around token restrictions can be foundhere .

Step 4: Configure your chain's validator(s)

You should see aConfigure Validators section appear, with a form that looks like this:

Number of Validators Validator 1 (0x...) [...] Validator n (0x...) The first input field is an integer value that determinesthe number of validators that will support your initial deployment . Subsequent fields allow you to specify each of these validators' addresses.

The first validator address is randomly generated and can't be changed. Its private key will be automatically generated and stored within one of the JSON configuration files that will be generated in a moment.

Your chain's validators are responsible for validating the integrity of transactions and posting assertions of the current state of your Orbit chain to its base chain. In production scenarios, your Orbit chain would likely be hosted by a network of validator nodes working together. For your local Orbit chain, you can stick to the auto-generated single validator address.

Each of the validator addresses specified in this step will be added to an allow-list in one of your chain'sbase contracts , allowing them each tostake and validate transactions submitted to your Orbit chain.

NEW TERMINOLOGY When we say "base contracts" and "base chain", we're referring to your Orbit chain's L2 contracts and the L2 chain that they're deployed to, respectively. We'll use these terms throughout the rest of this guide. Once your validator addresses are configured, clickNext to proceed the next step:batch poster configuration .

Step 5: Configure your chain's batch poster

You should see aConfigure Batch Poster section appear, with a form that looks like this:

Batch Poster Address Your batch poster address is responsible for posting batches of transactions from your Orbit chain to its base contracts on its base chain. An address will automatically be generated for you; its private key will be automatically generated and stored within one of the JSON configuration files that will be generated in a moment.

Once your batch poster address is configured, clickNext to proceed to the next step:review & deploy your Orbit chain .

Step 6: Review & Deploy your Orbit chain

Deploy your chain's base contracts to Arbitrum Goerli or Sepolia

Click theDeploy button located below the config form. Your wallet should prompt you to submit a transaction to the Arbitrum testnet. You'll have to pay a little gas; your wallet may denominate this in ETH; as long as you see your chosen Arbitrum testnet in the transaction details, this gas fee will be paid in testnet ETH.

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

- You submitted a deployment transaction to an Orbit "factory" smart contract on the Arbitrum testnet, the public L2 chain that your local Orbit chain will settle transactions to.
- 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 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.

Once your transaction is complete, if you deployed an AnyTrust chain, you'll next be asked to configure your keyset.

Otherwise, continue toStep 8 to download your chain's configuration files and launch your chain.

Step 7: Configure keyset (AnyTrust chains only)

Not applicable to Rollup chains Skip this step if you're deploying a Rollup chain. For the Batch Poster to function correctly, it's essential that the keyset corresponding to its current configuration is active within theSequencerInbox contract. The production of the keyset and keyset hash binary blobs is mandatory, which should then be used as inputs for theSetValidKeyset method on theSequencerInbox contract.

The current version of Orbit AnyTrust chains uses a single Data Availability Server and assigns a null value to its private key in order to generate an initial keyset. As part of this transaction process, you'll assign this initial keyset to your recently generatedSequencerInbox contract.

Click theDeploy button located below the Deployment Summary. Again, your wallet should prompt you to submit a transaction to the Arbitrum testnet. Gas fees will be handled similarly to the previous transaction.

Once the transaction completes, you'll be directed to the download page to continue your chain deployment.

Step 8: Download your chain's configuration files and launch your chain

You should see two JSON code blocks appear labeledRollup Config andL3 Config . Use the download buttons at the top right of each code block to save it locally.

1. Download Rollup JSON
2. : This will generatenodeConfig.json
3. , which contains yourchain's node configuration
4. . Note that this includes the private keys for your validator (staker) and batch poster, which are used to sign transactions that post RBlocks and batches to your chain's base contracts on L2.
5. Download L3Config JSON
6. : This will generateorbitSetupScriptConfig.json
7. , which contains yourchain's configuration
8. , including that which supports yourToken Bridge contracts
9. .

Step 9: Clone the setup script repository and add your configuration files

1. Clone theorbit-setup-script
2. repository:git clone https://github.com/OffchainLabs/orbit-setup-script.git
3. Move thenodeConfig.json
4. file that you downloaded into theconfig
5. directory in the root of your clonedorbit-setup-script
6. repository.
7. Move theorbitSetupScriptConfig.json
8. file you downloaded into theconfig
9. directory in the root of your clonedorbit-setup-script
10. repository.
11. Install dependencies by runningyarn install
12. from the root of theorbit-setup-script
13. repository.

Step 10: Run your chain's node and block explorer

Run Docker, then rundocker-compose up -d from the root of theorbit-setup-script repository.

A Nitro node and BlockScout explorer instance will be started. Visithttp://localhost:4000/ 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 11: Finish setting up your chain

We've provided 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.

Using Arbitrum Goerli:

PRIVATE_KEY

"0xYourPrivateKey"

L2_RPC_URL

"https://goerli-rollup.arbitrum.io/rpc"

L3_RPC_URL

"http://localhost:8449"

yarn run setup Using Arbitrum Sepolia:

PRIVATE_KEY

"0xYourPrivateKey"

L2_RPC_URL

"https://sepolia-rollup.arbitrum.io/rpc"

L3_RPC_URL

"http://localhost:8449"

yarn run setup

Congratulations

Your local Orbit chain is now running. You'll see anoutputInfo.json file in the main directory of your script folder - this contains more information about your chain, including the addresses of your chain's base contracts.

Appendix A: Logging

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

docker-compose logs -f nitro

Appendix B: 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 Goerli:

PRIVATE_KEY

"0xYourPrivateKey"

L2_RPC_URL

"https://goerli-rollup.arbitrum.io/rpc"

L3_RPC_URL

"http://localhost:8449"

AMOUNT

""

yarn run deposit 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 C: Troubleshooting

- You may seeerror getting latest batch count
- in your node's output logs (from Appendix A). 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. Edit this page Last updatedonMar 7, 2024 Previous A gentle introduction: Orbit chains Next How to run a full node for an Orbit chain