OP Stack Smart Contract Deployment

The following guide shows you how to deploy the OP Stack L1 smart contracts. The primary development branch isdevelop . It contains the most up-to-date software that remains backwards compatible with the latest experimental network deployments.Changes to the smart contracts are generally not considered backwards compatible.

Standard OP Stack chains should use governance approved and audited versions of the smart contract code.

Deployment Configuration

Deploying your OP Stack contracts requires creating a deployment configuration JSON file. For the full set of options, you can see therollup configuration page .

Deployment Script

The smart contracts are deployed usingfoundry(opens in a new tab) and you can find the script's source code in the monorepo atpackages/contracts-bedrock/scripts/deploy/Deploy.s.sol(opens in a new tab) .

State Diff

Before deploying the contracts, you can verify the state diff by using therunWithStateDiff() function signature in the deployment script, which produces the outputs insidesnapshots/state-diff/ (opens in a new tab) . Run the deployment with state diffs by executing:

```
forge

script

-vvv

scripts/Deploy.s.sol:Deploy

--sig

'runWithStateDiff()'

--rpc-url ETH_RPC_URL --broadcast

--private-key PRIVATE_KEY
```

Execution

- Set theETHERSCAN_API_KEY
- and add the--verify
- flag to verify your
- contracts.
- DEPLOYMENT_OUTFILE
- will determine the filepath that the deployment
- artifact is written to on disk after the deployment. It comes in the form of a
- JSON file where keys are the names of the contracts and the values are the
- addresses the contract was deployed to.
- DEPLOY_CONFIG_PATH
- is the path on the filesystem that points to a deployment
- config. The same deployment config JSON file should be used for L1 contracts
- deployment as when generating the L2 genesis allocs. See thedeploy-config(opens in a new tab)
- directory for examples and therollup configuration page
- for descriptions of the values.
- IMPL_SALT
- env var can be used to set the create2 salt for deploying the
- implementation contracts.

This will deploy an entire new system of L1 smart contracts, including a new SuperchainConfig. In the future, there will be an easy way to deploy only proxies and use shared implementations for each of the contracts as well as a shared SuperchainConfig contract.

```
DEPLOYMENT_OUTFILE=deployments/artifact.json \ DEPLOY_CONFIG_PATH= \ forge script scripts/Deploy.s.sol:Deploy
\ --broadcast --private-key PRIVATE_KEY \ --rpc-url ETH_RPC_URL
```

Deploying a single contract

All functions for deploying a single contract are public, meaning that All functions for deploying a single contract are public, meaning that the--sig argument to forge script can be used to target the deployment of a single contract.

Best Practices

Production users should deploy their L1 contracts from a contracts release. All contracts releases are on git tags with the following format:op-contracts/vX.Y.Z . If you're deploying a new standard chain, you should deploy theMulti-Chain Prep (MCP) L1 release(opens in a new tab) . We're working on writing more documentation to prepare OP Stack chain operators to run a fault proof chain effectively.

Next Steps

- Learn how tocreate your genesis file
- See allconfiguration options
- and example configurations

Overview Genesis Creation