Local Testing

This guide introduces a ready-to-use mock test environment, that contains:

- A local Alto bundler
- A mock verifying paymaster
- An Anvil node
- ERC-4337 related contracts, including the EntryPoint and account factories for all major ERC-4337 account implementations.

The test environment is orchestrated usingdocker compose . Where the docker containers are pulled fromthis repo .

The mock environment is designed to mimic mainnet as closely as possible by building with the latest Alto version and by deploying all contracts (entrypoints, paymasters, smart account factories, etc.)

This has advantages over testing against a production testnet as you have more control over the testing environment and can make use of features like anvil cheatcodes.

Steps

Setup

To get started, create adocker-compose.yaml file at the root of your test directory with the following contents.

```

File docker-compose.yaml services: anvil: image:ghcr.io/foundry-rs/foundry:nightly-f6208d8db68f9acbe4ff8cd76958309efb61ea0b ports: ["8545:8545"] entrypoint: ["anvil","--host","0.0.0.0","--block-time","0.1","--silent"] platform:linux/amd64/v8

mock-paymaster: image:ghcr.io/pimlicolabs/mock-verifying-paymaster:main ports: ["3000:3000"] environment: -ALTO_RPC=http://alto:4337 -ANVIL_RPC=http://anvil:8545

alto: image:ghcr.io/pimlicolabs/mock-alto-bundler:main ports: ["4337:4337"] environment: -ANVIL_RPC=http://anvil:8545

```

To start the test environment, run

```

dockercompose up

```

Once docker has started, the following services can be accessed locally through the following endpoints:

- Anvil
- atlocalhost:8545
- Alto Bundler
- atlocalhost:4337
- Mock Paymaster
- atlocalhost:3000

You can now use permissionless like you normally would but instead of referencing the live endpoints, use the local endpoints mentioned above when creating the clients.

```

File clients.ts import{ createBundlerClient, ENTRYPOINT_ADDRESS_V07 }from'permissionless' import{ createPimlicoPaymasterClient }from"permissionless/clients/pimlico" import{ http, createPublicClient }from"viem" import{ foundry }from'viem/chains'

constpublicClient=createPublicClient({ transport:http("http://localhost:8545"), })

constbundlerClient=createBundlerClient({ chain: foundry, transport:http("http://localhost:4337"), entryPoint:ENTRYPOINT_ADDRESS_V07, })

constpaymasterClient=createPimlicoPaymasterClient({ chain: foundry, transport:http("http://localhost:3000"), entryPoint:ENTRYPOINT_ADDRESS_V07, })
```

```

Note: All smart account types supported by permissionless.js will work out of the box as all related factories and modules are deployed on the local anvil instance.

## Vitest Integration

You can add scripts in yourpackage.json to automatically set up and tear down your mock environment when running tests.

```

File package.json { "name":"aa-tests", "scripts": { "test":"bun run docker:up && vitest run && bun run docker:down", "docker:up":"docker-compose up -d", "docker:down":"docker-compose down" }, "dependencies": { "viem":"^2.9.17", "permissionless":"^0.1.24" }, "devDependencies": { "vitest":"^1.5.2" } }

```

When writing test cases, ensure that the bundler and paymaster are fully setup before sending any request to them. To do this, make a simple health check in thebeforeAll declaration.

basic.test.ts healthCheck.ts ```

File basic.test.ts import{ beforeAll, describe, expect, test }from"vitest" import{ ensureBundlerIsReady, ensurePaymasterIsReady }from"./healthCheck" import{ foundry }from'viem/chains' import{ http }from"viem" import{ createBundlerClient, ENTRYPOINT_ADDRESS_V06, ENTRYPOINT_ADDRESS_V07 }from'permissionless'

describe("Test basic bundler functions", ()=>{ beforeAll(async()=>{ awaitensureBundlerIsReady() awaitensurePaymasterIsReady() })

test("Can get chainId",async()=>{ constbundlerClient=createBundlerClient({ chain: foundry, transport:http("http://localhost:4337"), entryPoint:ENTRYPOINT_ADDRESS_V07, })

constchainId=awaitbundlerClient.chainId()

expect(chainId).toEqual(foundry.id) })

test("Can get supported entryPoints",async()=>{ constbundlerClient=createBundlerClient({ chain: foundry, transport:http("http://localhost:4337"), entryPoint:ENTRYPOINT_ADDRESS_V07, })

constsupportedEntryPoints=awaitbundlerClient.supportedEntryPoints()

expect(supportedEntryPoints).toEqual([ENTRYPOINT_ADDRESS_V06,ENTRYPOINT_ADDRESS_V07]) }) })

``` For a more detailed example, take a look at thepermissionless.js E2E integration tests which uses the same mock environment as this guide.

## Extension: Testing against forked state

If you want your tests to run against a live blockchain, you can slightly edit thedocker-compose.yaml file to fork from the latest block by adding the anvil flag--fork-url and the environment variableSKIP_DEPLOYMENTS to skip the local contract deployments.

```

File docker-compose.yaml services: anvil: image:ghcr.io/foundry-rs/foundry:nightly-f6208d8db68f9acbe4ff8cd76958309efb61ea0b ports: ["8545:8545"] entrypoint: ["anvil","--fork-url","https://rpc.sepolia.org","--host","0.0.0.0","--block-time","0.1","--silent"] platform:linux/amd64/v8

mock-paymaster: image:ghcr.io/pimlicolabs/mock-verifying-paymaster:main ports: ["3000:3000"] environment: -ALTO_RPC=http://alto:4337 -ANVIL_RPC=http://anvil:8545

alto: image:ghcr.io/pimlicolabs/mock-alto-bundler:main ports: ["4337:4337"] environment: -ANVIL_RPC=http://anvil:8545 -SKIP_DEPLOYMENTS=true

```