**Running tests on chains forks**

**Forking tests**

This course will cover 4 different types of tests:

* **Unit tests**: Focus on isolating and testing individual smart contract functions or functionalities.
* **Integration tests**: Verify how a smart contract interacts with other contracts or external systems.
* **Forking tests**: Forking refers to creating a copy of a blockchain state at a specific point in time. This copy, called a fork, is then used to run tests in a simulated environment.
* **Staging tests**: Execute tests against a deployed smart contract on a staging environment before mainnet deployment.

Coming back to our contracts, the central functionality of our protocol is the fund function.

For that to work, we need to be sure that Aggregator V3 runs the current version. We know from previous courses that the version returned needs to be 4. Let's put it to the test!

Add the following code to your test file:

function testPriceFeedVersionIsAccurate() public {

uint256 version = fundMe.getVersion();

assertEq(version, 4);

}

It ... fails. But why? Looking through the code we see this AggregatorV3 address 0x694AA1769357215DE4FAC081bf1f309aDC325306 over and over again. The address is correct, is the Sepolia deployment of the AggregatorV3 contract. But our tests use Anvil for testing purposes, so that doesn't exist.

**Note: Calling forge test over and over again when you are testing is not always efficient, imagine you have tens or hundreds of tests, some of them taking seconds to finish. From now on, when we test specific things let's use the following:**

forge test --mt testPriceFeedVersionIsAccurate

Back to our problem, how can we fix this?

Forking is the solution we need. If we run the test on an anvil instance that copies the current Sepolia state, where AggregatorV3 exists at that address, then our test function will not revert anymore. For that, we need a Sepolia RPC URL.

Remember how in a [previous lesson we delpoyed a smart contract on Sepolia](https://updraft.cyfrin.io/courses/foundry/foundry-simple-storage/deploying-smart-contract-testnet-sepolia)? It's similar, we can use the same RPC we used back then.

Thus:

1. Create a .env file. (Also make sure that your .gitignore file contains the .env entry)
2. In the .env file create a new entry as follows:

SEPOLIA\_RPC\_URL=https://eth-sepolia.g.alchemy.com/v2/YOURAPIKEYWILLGOHERE

1. Run source .env in your terminal;
2. Run forge test --mt testPriceFeedVersionIsAccurate --fork-url $SEPOLIA\_RPC\_URL

Ran 1 test for test/FundMe.t.sol:FundMeTest

[PASS] testPriceFeedVersionIsAccurate() (gas: 14118)

Suite result: ok. 1 passed; 0 failed; 0 skipped; finished in 2.29s (536.03ms CPU time)

Nice!

Please keep in mind that forking uses the Alchemy API, it's not a good idea to run all your tests on a fork every single time. But, sometimes as in this case, you can't test without. It's very important that our test have a high **coverage**, to ensure all our code is battle tested.

**Coverage**

Foundry provides a way to calculate the coverage. You can do that by calling forge coverage. This command displays which parts of your code are covered by tests. Read more about its options [here](https://book.getfoundry.sh/reference/forge/forge-coverage?highlight=coverage#forge-coverage).

forge coverage --fork-url $SEPOLIA\_RPC\_URL

Ran 3 tests for test/FundMe.t.sol:FundMeTest

[PASS] testMinimumDollarIsFive() (gas: 5759)

[PASS] testOwnerIsMsgSender() (gas: 8069)

[PASS] testPriceFeedVersionIsAccurate() (gas: 14539)

Suite result: ok. 3 passed; 0 failed; 0 skipped; finished in 1.91s (551.69ms CPU time)

Ran 1 test suite in 2.89s (1.91s CPU time): 3 tests passed, 0 failed, 0 skipped (3 total tests)

| File | % Lines | % Statements | % Branches | % Funcs |

| ------------------------- | ------------- | ------------- | ------------- | ------------ |

| script/DeployFundMe.s.sol | 0.00% (0/3) | 0.00% (0/3) | 100.00% (0/0) | 0.00% (0/1) |

| src/FundMe.sol | 21.43% (3/14) | 25.00% (5/20) | 0.00% (0/6) | 33.33% (2/6) |

| src/PriceConverter.sol | 0.00% (0/6) | 0.00% (0/11) | 100.00% (0/0) | 0.00% (0/2) |

| Total | 13.04% (3/23) | 14.71% (5/34) | 0.00% (0/6) | 22.22% (2/9) |

These are rookie numbers! Maybe 100% is not feasible, but 13% is as good as nothing. In the next lessons, we will up our game and increase these numbers!

**Updates:**

Changes on test command

*Last updated on July 25, 2024*

On the lastes version of Foundry, the test flag -m is not supported, so instead of using the command.

forge test -m testPriceFeedVersionIsAccurate -vvv --fork-url $SEPOLIA\_RPC\_URL

You need to use:

forge test --mt testPriceFeedVersionIsAccurate -vvvv --fork-url $SEPOLIA\_RPC\_URL