**Refactoring the mock smart contract**

**Solving the Anvil problem**

When we needed the Sepolia priceFeed we made sure that our deployments script pointed to it. How can we solve this in Anvil, because the contract that we need doesn't exist there. Simple, we deploy mocks.

Our getAnvilEthConfig function in HelperConfig must deploy a mock contract. After it deploys it we need it to return the mock address, so our contracts would know where to send their calls.

First of all, we need to make sure we import Script.sol in the HelperConfig.s.sol contract. Also, HelperConfig needs to inherit from Script. This will give us access to the vm.startBroadcast/vm.stopBroadcast functionality. We will use this to deploy our mocks. But ... what is a mock contract?

**A mock contract is a special type of contract designed to simulate the behavior of another contract during testing.**

Update your start of the HelperConfig.s.sol file as follows:

import {Script} from "forge-std/Script.sol";

contract HelperConfig is Script {

}

In order to be able to deploy a mock contract we need ... a mock contract. So, in test folder please create a new folder called mocks. Inside mocks create a new file MockV3Aggregator.sol. Rewriting the AggregatorV3 as a mock is not the easiest task out there. Please copy the contents of [this contract](https://github.com/Cyfrin/foundry-fund-me-f23/blob/main/test/mock/MockV3Aggregator.sol) into your newly created file.

What next?

We need to import this in our HelperConfig.s.sol file and deploy it in the getAnvilEthConfig then return the address.

Perform the following changes in HelperConfig:

import {MockV3Aggregator} from "../test/mocks/MockV3Aggregator.sol";

[...]

// In state variables section

MockV3Aggregator mockPriceFeed;

[...]

function getAnvilEthConfig() public returns (NetworkConfig memory) {

vm.startBroadcast();

mockPriceFeed = new MockV3Aggregator(8, 2000e8);

vm.stopBroadcast();

NetworkConfig memory anvilConfig = NetworkConfig({

priceFeed: address(mockPriceFeed)

});

return anvilConfig;

}