**Refactoring your tests**

**Refactoring code and test**

The way our code is currently structured is not that flexible. Given the hardcoded Sepolia address we cannot deploy to other chains, and if we wish to do so we would have to come and copy-paste another address everywhere throughout the code. In bigger codebases, with multiple addresses / other items to copy-paste for each deployment (in case we deploy on multiple chains) this update activity is extremely prone to error. We can do better.

To fix this we can make our project more modular, which would help improve the maintainability, testing and deployment. This is done by moving the hardcoded changing variables to the constructor, thus regardless of the chain we deploy our contracts to, we provide the chain-specific elements once, in the constructor, and then we are good to go.

Changing code without changing its functionality bears the name of **refactoring**.

Do the following modifications in FundMe.sol

In the storage variables section create a new variable:

AggregatorV3Interface private s\_priceFeed;

We need to add this as an input in our constructor and assign it to the state variable. This is done as follows:

constructor(address priceFeed){

i\_owner = msg.sender;

s\_priceFeed = AggregatorV3Interface(priceFeed);

}

Inside the getVersion function, where AggregatorV3Interface is invoked, replace the hardcoded address with the state variable s\_priceFeed:

function getVersion() public view returns (uint256){

AggregatorV3Interface priceFeed = AggregatorV3Interface(s\_priceFeed);

return priceFeed.version();

}

1. In PriceConverter.sol modify the getPrice function to take an input function getPrice(AggregatorV3Interface priceFeed) internal view returns (uint256) { and delete the AggregatorV3Interface priceFeed = AggregatorV3Interface(0x694AA1769357215DE4FAC081bf1f309aDC325306); line;
2. In the same library update getConversionRate to take a priceFeed input and update the first line to pass the required AggregatorV3Interface to the getPrice function:

function getConversionRate(

uint256 ethAmount,

AggregatorV3Interface priceFeed

) internal view returns (uint256) {

uint256 ethPrice = getPrice(priceFeed);

uint256 ethAmountInUsd = (ethPrice \* ethAmount) / 1000000000000000000;

// the actual ETH/USD conversion rate, after adjusting the extra 0s.

return ethAmountInUsd;

}

Back in FundMe.sol pass the s\_priceFeed as input for getConversionRate in the fund function.

Take a moment and think if we missed updating anything in our project.

Ready? The deploy script is not providing the priceFeed as an input when calling new FundMe();, also, the setUp function in FundMe.t.sol is not providing the priceFeed as an input when calling fundMe = new FundMe();.

For now, let's hardcode the address 0x694AA1769357215DE4FAC081bf1f309aDC325306 in both places.

As you've figured out this isn't ideal either. Every time we want to do something from now on do we have to update in both places? Not good.

Update the run function from the DeployFundMe script:

function run() external returns (FundMe) {

vm.startBroadcast();

FundMe fundMe = new FundMe(0x694AA1769357215DE4FAC081bf1f309aDC325306);

vm.stopBroadcast();

return fundMe }

Now when we call run it returns the FundMe contract we deployed.

In FundMe.t.sol:

1. Let's import the deployment script into the FundMe.t.sol.

import {DeployFundMe} from "../script/DeployFundMe.s.sol";

1. Create a new state variable DeployFundMe deployFundMe;;
2. Update the setUp function as follows:

function setUp() external {

deployFundMe = new DeployFundMe();

fundMe = deployFundMe.run();

}

Let's call a forge test --fork-url $SEPOLIA\_RPC\_URL to make sure everything compiles.

Looks like the testOwnerIsMsgSender fails again. Take a moment and think about why.

When we changed the method of deployment and made it go through the run command of the deployFundMe contract we also changed the owner.

**Note**: vm.startBroadcast is special, it uses the address that calls the test contract or the address / private key provided as the sender. You can read more about it here.

To account for the way vm.startBroadcast works please perform the following modification in FundMe.t.sol:

function testOwnerIsMsgSender() public {

assertEq(fundMe.i\_owner(), msg.sender);

}

Run forge test --fork-url $SEPOLIA\_RPC\_URL again.

Congrats!