**How to refactor magic number**

**Magic numbers or What was this exactly?**

Magic numbers refer to literal values directly included in the code without any explanation or context. These numbers can appear anywhere in the code, but they're particularly problematic when used in calculations or comparisons. By using magic numbers you ensure your smart contract suffers from **Reduced Readability**, **Increased Maintenance Difficulty** and **Debugging Challenges**. You also make your work extremely prone to error, imagine you used the same magic number in 10 places and you want to change it. Will you remember all the 9 places or will you change it only in 8?

**Don't be like that.**

Write clean, maintainable, and less error-prone code. You make your own life easier, you make your auditor(s) life easier. Use constants and configuration variables.

Let's apply this.

Open HelperConfig.s.sol, go to the getAnvilEthConfig function and delete the 8 corresponding to the decimals and 2000e8 corresponding to the \_initialAnswer that are used inside the MockV3Aggregator's constructor.

At the top of the HelperConfig contract create two new variables:

uint8 public constant DECIMALS = 8;

int256 public constant INITIAL\_PRICE = 2000e8;

**Note: Constants are always declared in ALL CAPS!**

Now replace the deleted magic numbers with the newly created variables.

function getAnvilEthConfig() public returns (NetworkConfig memory) {

vm.startBroadcast();

mockPriceFeed = new MockV3Aggregator(DECIMALS, INITIAL\_PRICE);

vm.stopBroadcast();

NetworkConfig memory anvilConfig = NetworkConfig({

priceFeed: address(mockPriceFeed)

});

return anvilConfig; }

Awesome! Let's keep refactoring!