**Test and deploy the lottery smart contract pt.1**

**Deploying and testing our lottery**

Now that we've got all the prerequisites for deployment let's proceed in deploying the raffle.

Let's open the DeployRaffle.s.sol and use our new tools.

First, import the newly created HelperConfig.

import {HelperConfig} from "./HelperConfig.s.sol";

Then, modify the run function:

function run() external returns (Raffle, HelperConfig) {

HelperConfig helperConfig = new HelperConfig(); // This comes with our mocks!

(

uint256 entranceFee;

uint256 interval;

address vrfCoordinator;

bytes32 gasLane;

uint64 subscriptionId;

uint32 callbackGasLimit;

) = helperConfig.activeNetworkConfig();

}

Great! Now that we have deconstructed the NetworkConfig we have all the variables we need to deploy::

vm.startBroadcast();

Raffle raffle = new Raffle(

entranceFee,

interval,

vrfCoordinator,

gasLane,

subscriptionId,

callbackGasLimit

)

vm.stopBroadcast();

return raffle;

We use the vm.startBroadcast and vm.stopBroadcast commands to indicate that we are going to send a transaction. The transaction is the deployment of a new Raffle contract using the parameters we've obtained from the HelperConfig. In the end, we are returning the newly deployed contract.

This code is good on its own, but, we can make it better. For example, we need a subscriptionId. We can either obtain this through the front end as we've learned in a previous lesson, or we can get on programmatically. For now, we'll leave everything as is, but we will refactor this in the future.

Before that, let's write some tests.

Inside the test folder create two new folders called integration and unit. Here we'll put our integration and unit tests. Inside the newly created unit folder create a file called RaffleTest.t.sol.

Let's start writing the first test. You've already done this at least two times in this section. Try to do it on your own and come back when you get stuck.

Your unit test should start like this:

// SPDX-License-Identifier: MIT

pragma solidity ^0.8.19;

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

import {Raffle} from "../../src/Raffle.sol";

import {Test, console} from "forge-std/Test.sol";

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

contract RaffleTest is Test {

}

We've declared the SPDX-License-Identifier, the solidity version, imported the DeployRaffle which we will use to deploy our contract, then Raffle the contract to be deployed and then Test and console which are required for Foundry to function.

In DeployRaffle.s.sol we need to make sure that run also returns the HelperConfig contract:

function run() external returns (Raffle, HelperConfig) {

HelperConfig helperConfig = new HelperConfig();

(

uint256 entranceFee,

uint256 interval,

address vrfCoordinator,

bytes32 gasLane,

uint64 subscriptionId,

uint32 callbackGasLimit

) = helperConfig.activeNetworkConfig();

vm.startBroadcast();

Raffle raffle = new Raffle(

entranceFee,

interval,

vrfCoordinator,

gasLane,

subscriptionId,

callbackGasLimit

);

vm.stopBroadcast();

return (raffle, helperConfig);

}

Next comes the state variables and setUp function in RaffleTest.t.sol:

contract RaffleTest is Test {

Raffle public raffle;

HelperConfig public helperConfig;

uint256 entranceFee;

uint256 interval;

address vrfCoordinator;

bytes32 gasLane;

uint64 subscriptionId;

uint32 callbackGasLimit;

address public PLAYER = makeAddr("player");

uint256 public constant STARTING\_USER\_BALANCE = 10 ether;

function setUp() external {

DeployRaffle deployer = new DeployRaffle();

(raffle, helperConfig) = deployer.run();

vm.deal(PLAYER, STARTING\_USER\_BALANCE);

(

entranceFee,

interval,

vrfCoordinator,

gasLane,

subscriptionId,

callbackGasLimit

) = helperConfig.activeNetworkConfig();

}

}

This seems like a lot, but it isn't, let's go through it.

* We made RaffleTest contract inherit Test to enable the testing functionality;
* We've defined a raffle and helperConfig variables to store the contracts;
* Next, we defined the variables required for the deployment;
* Then, we created a new user called PLAYER and defined how many tokens they should receive;
* Inside the setUp function, we deploy the DeployRaffle contract then we use it to deploy the Raffle and HelperConfig contracts;
* We deal the PLAYER the defined STARTING\_USER\_BALANCE;
* We call helperConfig.activeNetworkConfig to get the Raffle configuration parameters.

Amazing! With all these done let's write a small test to ensure our setUp is functioning properly.

First, we need a getter function to retrieve the raffle state. Put the following towards the end of the Raffle.sol:

function getRaffleState() public view returns (RaffleState) {

return s\_raffleState;

}

Inside RaffleTest.t.sol paste the following test:

function testRaffleInitializesInOpenState() public view {

assert(raffle.getRaffleState() == Raffle.RaffleState.OPEN);

}

**Note**: we used Raffle.RaffleState.OPEN to get the value attributed to OPEN inside the RaffleState enum. This is possible because RaffleState is considered a [type](https://docs.soliditylang.org/en/latest/types.html#enums). So we can access that by calling the type RaffleState inside a Raffle contract to retrieve the OPEN value.

Great! Let's run the test and see how it goes:

forge test --mt testRaffleInitializesInOpenState -vv

The output being:

Ran 1 test for test/unit/RaffleTest.t.sol:RaffleTest

[PASS] testRaffleInitializesInOpenState() (gas: 7707)

Suite result: ok. 1 passed; 0 failed; 0 skipped; finished in 12.42ms (51.80µs CPU time)

Ran 1 test suite in 2.25s (12.42ms CPU time): 1 tests passed, 0 failed, 0 skipped (1 total tests)

Ok, so our Raffle starts in an OPEN state. Exactly like we coded it!

Great job! We started testing, let's see what we can do next!