**Implement the Chainlink VRF**

**Getting Started with Chainlink VRF**

🗒️ **NOTE**: This written lesson uses VRF V2. Video lesson uses VRF V2.5. There are some changes. Import path for VRF contract is slightly different, and the requestRandomWords() parameter is slightly different\*\*

Continuing the previous lesson, let's integrate Chainlink VRF into our Raffle project.

Coming back to our pickWinner function.

// 1. Get a random number

// 2. Use the random number to pick a player

// 3. Automatically called

function pickWinner() external {

// check to see if enough time has passed

if (block.timestamp - s\_lastTimeStamp < i\_interval) revert();

}

Let's focus on points 1 and 2. In the previous lesson, we learned that we need to request a randomWord and Chainlink needs to callback one of our functions to answer the request. Let's copy the requestId line from the [Chainlink VRF docs](https://docs.chain.link/vrf/v2/subscription/examples/get-a-random-number#analyzing-the-contract) example inside our pickWinner function and start fixing the missing dependencies.

function pickWinner() external {

// check to see if enough time has passed

if (block.timestamp - s\_lastTimeStamp < i\_interval) revert();

uint256 requestId = COORDINATOR.requestRandomWords(

keyHash,

s\_subscriptionId,

requestConfirmations,

callbackGasLimit,

numWords

);

}

You know the keyHash, subId, requestConfirmations, callbackGasLimit and numWords from our previous lesson.

Ok, starting from the beginning what do we need?

1. We need to establish the fact that our Raffle contract is a consumer of Chainlink VRF;
2. We need to take care of the VRF Coordinator, define it as an immutable variable and give it a value in the constructor;

Let's add the following imports:

import {VRFCoordinatorV2Interface} from "chainlink/src/v0.8/vrf/interfaces/VRFCoordinatorV2Interface.sol";

import {VRFConsumerBaseV2} from "chainlink/src/v0.8/vrf/VRFConsumerBaseV2.sol";

Let's make our contract inherit the VRFConsumerBaseV2:

contract Raffle is VRFConsumerBaseV2

Add a new immutable variable:

// Chainlink VRF related variables

address immutable i\_vrfCoordinator;

I've divided the Raffle variables from the Chainlink VRF variables to keep the contract tidy.

Adjust the constructor to accommodate all the new variables and imports:

constructor(uint256 entranceFee, uint256 interval, address vrfCoordinator) {

i\_entranceFee = entranceFee;

i\_interval = interval;

s\_lastTimeStamp = block.timestamp;

i\_vrfCoordinator = vrfCoordinator;

}

For our imports to work we need to install the Chainlink library, and run the following command in your terminal:

forge install smartcontractkit/chainlink@42c74fcd30969bca26a9aadc07463d1c2f473b8c --no-commit

*P.S. I know it doesn't look amazing, bear with me.*

Now run forge build. **It will fail**, it should fail because we didn't define a ton of variables. But what matters at this point is how it fails! We need it to fail with the following error:

Error:

Compiler run failed:

Error (7576): Undeclared identifier.

--> src/Raffle.sol:53:8:

If it doesn't fail with that error but fails with this error then we need to do additional things:

Error:

Compiler run failed:

Error (6275): Source "lib/chainlink/contracts/contracts/src/v0.8/vrf/dev/VRFConsumerBaseV2Plus.sol" not found: File not found. Searched the following locations:

If you got the error above, then forge was not able to find out the contracts we imported. Run the following command in your terminal:

forge remappings>remappings.txt

This will create a new file that contains your project remappings:

chainlink/=lib/chainlink/contracts/

forge-std/=lib/forge-std/src/

This is to be read as follows: chainlink/ in your imports becomes lib/chainlink/contracts/ behind the stage. We need to make sure that if we apply that change to our import the resulting **PATH is correct**.

chainlink/src/v0.8/vrf/VRFConsumerBaseV2.sol becomes lib/chainlink/contracts/src/v0.8/vrf/VRFConsumerBaseV2.sol, which is correct. Sometimes some elements of the PATH are either missing or doubled, as follows:

lib/chainlink/contracts/contracts/src/v0.8/vrf/dev/VRFConsumerBaseV2Plus.sol

or

lib/chainlink/src/v0.8/vrf/dev/VRFConsumerBaseV2Plus.sol

Both these variants are **incorrect**. You need to always be able to access the PATH in your explorer window on the left, if you can't then you need to adjust the remappings to match the lib folder structure.

Great, now that we were successfully able to run the imports let's continue fixing the missing variables.

Don't ever be afraid of calling forge build even if you know your project won't compile. Our contract lacks some variables that are required inside the pickWinner function. Call forge build.

Output:

Compiler run failed:

Error (7576): Undeclared identifier.

--> src/Raffle.sol:55:26:

|

55 | keyHash: s\_keyHash, // gas lane

| ^^^^^^^^^

Error (7576): Undeclared identifier.

--> src/Raffle.sol:56:24:

|

56 | subId: s\_subscriptionId, // subscription ID

| ^^^^^^^^^^^^^^^^

Error (7576): Undeclared identifier.

--> src/Raffle.sol:57:39:

|

57 | requestConfirmations: requestConfirmations,

| ^^^^^^^^^^^^^^^^^^^^

Error (7576): Undeclared identifier.

--> src/Raffle.sol:58:35:

|

58 | callbackGasLimit: callbackGasLimit,// make sure we don't overspend

| ^^^^^^^^^^^^^^^^

Error (7576): Undeclared identifier.

--> src/Raffle.sol:59:27:

|

59 | numWords: numWords, // number random numbers

At least now we know what's left :smile:

Let's add the above-mentioned variables inside the VRF state variables block:

// Chainlink VRF related variables

VRFCoordinatorV2Interface private immutable i\_vrfCoordinator;

bytes32 private immutable i\_gasLane;

uint64 private immutable i\_subscriptionId;

uint16 private constant REQUEST\_CONFIRMATIONS = 3;

uint32 private immutable i\_callbackGasLimit;

uint32 private constant NUM\_WORDS = 1;

We have changed the keyHash name to i\_gasLane which is more descriptive for its purpose. Also, we've changed the type of i\_vrfCoordinator. For our pickWinner function to properly call uint256 requestId = i\_vrfCoordinator.requestRandomWords( we need that i\_vrfCoordinator to be a contract, specifically the VRFCoordinatorV2Interface contract that we've imported.

For simplicity we request only 1 word, thus we make that variable constant. The same goes for request confirmations, this number can vary depending on the blockchain we chose to deploy to but for mainnet 3 is perfect. Cool!

The next step is to attribute values inside the constructor:

constructor(uint256 entranceFee, uint256 interval, address vrfCoordinator, bytes32 gasLane, uint64 subscriptionId, uint32 callbackGasLimit) VRFConsumerBaseV2(vrfCoordinator) {

i\_entranceFee = entranceFee;

i\_interval = interval;

s\_lastTimeStamp = block.timestamp;

i\_vrfCoordinator = VRFCoordinatorV2Interface(vrfCoordinator);

i\_gasLane = gasLane;

i\_subscriptionId = subscriptionId;

i\_callbackGasLimit = callbackGasLimit;

}

Ok, breathe, it's a lot but it's not complicated, let's go through it together:

* First, we need to initiate the VRFConsumerBaseV2 using our constructor VRFConsumerBaseV2(vrfCoordinator);
* We are providing the gasLane, subscriptionId and callbackGasLimit in our input section of the constructor;
* We are assigning the inputted values to the state variables we defined at an earlier point;
* We are casting the vrfCoordinator address as VRFCoordinatorV2Interface to be able to call it inside the pickWinner function.

The last step is to create a new function:

function fulfillRandomWords(uint256 requestId, uint256[] memory randomWords) internal override {}

This will be called by the vrfCoordinator when it sends back the requested randomWords. This is also where we'll select our winner!

Call the forge build again.

[⠒] Compiling...

[⠔] Compiling 9 files with Solc 0.8.25

[⠒] Solc 0.8.25 finished in 209.77ms

Compiler run successful with warnings:

Warning (2072): Unused local variable.

--> src/Raffle.sol:61:9:

|

61 | uint256 requestId = i\_vrfCoordinator.requestRandomWords(

| ^^^^^^^^^^^^^^^^^

Perfect! Don't worry. We will use that requestId in a future lesson.