

HARDHAT SMART CONTRACT LOTTERY-RAFFLE 13.14.03

HARDHAT SETUP 13.43.44

```
mkdir hardhat-smartcontract-lottery-cc
```

```
cd hardhat-smartcontract-lottery-cc
```

```
code .
```

```
// terminal aç
```

```
yarn add --dev hardhat
```

NOT : EĞER NVM UYUMSUZLUĞU VAR İSE

```
nvm use --delete-prefix v16.15.0
```

PREFIX KISMINDA SİZDE KURULU OLAN SÜRÜMÜ YAZIN

yarn hardhat

? What do you want to do? ...

Create a basic sample project

Create an advanced sample project

Create an advanced sample project that uses TypeScript

► **Create an empty hardhat.config.js**

Quit

KOYU OLANI SEÇ

Gerekli kurulumları

<https://github.com/smartcontractkit/full-blockchain-solidity-course-js#lesson-9-hardhat-smart-contract-lottery>

adresinden kopyalayla



```
yarn add --dev @nomiclabs/hardhat-ethers@npm:hardhat-deploy-ethers ethers @nomiclabs/hardhat-etherscan  
@nomiclabs/hardhat-waffle chai ethereum-waffle hardhat hardhat-contract-sizer hardhat-deploy hardhat-gas-reporter  
prettier prettier-plugin-solidity solhint solidity-coverage dotenv
```

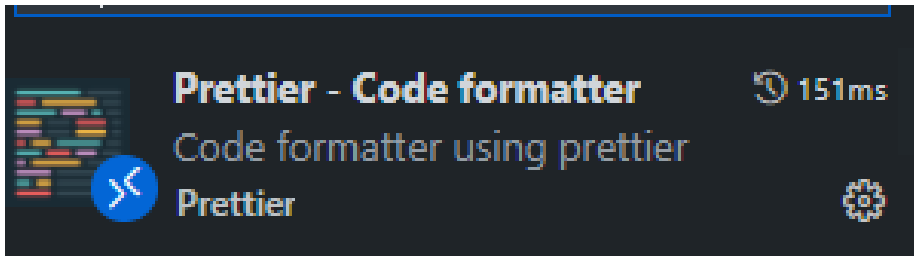
Kurulan paketleri package.json dosyasından kontrol edebilirsiniz.

```
{ } package.json X
1  {
2    "devDependencies": {
3      "@nomiclabs/hardhat-ethers": "npm:hardhat-deploy-ethers",
4      "@nomiclabs/hardhat-etherscan": "^3.1.0",
5      "@nomiclabs/hardhat-waffle": "^2.0.3",
6      "chai": "^4.3.6",
7      "dotenv": "^16.0.1",
8      "ethereum-waffle": "^3.4.4",
9      "ethers": "^5.6.8",
10     "hardhat": "^2.9.9",
11     "hardhat-contract-sizer": "^2.5.1",
12     "hardhat-deploy": "^0.11.10",
13     "hardhat-gas-reporter": "^1.0.8",
14     "prettier": "^2.6.2",
```

hardhat.config.js için gerekli paketleri içe aktar.

```
require("@nomiclabs/hardhat-waffle");
require("@nomiclabs/hardhat-etherscan");
require("hardhat-deploy");
require("solidity-coverage");
require("hardhat-gas-reporter");
require("hardhat-contract-sizer");
require("dotenv").config();
```

;" kullanmamak için prettier ayarını yap. .prettierrc dosyası oluştur ve aşağıdakileri ekle. ( Bunun için aşağıdaki extension ı kurmak gerekir )



```
{
  "tabWidth": 4,
  "useTabs": false,
  "semi": false,
  "singleQuote": false,
  "printWidth": 100
}
```

```
module.exports = {  
  solidity: "0.8.7",  
}
```



RAFFLE.SOL SETUP 13.46.57

Contracts/Raffle.sol

KLAÖSR VE DOSYAYI OLUŞTUR.

Yazacağımız contractta;

Herhangi biri bir miktar ödeme yaparak çekilişe katılacak

Katılanlardan rastgele bir kazanan seçilecek. ( Rastgele işlemi doğrulanabilir - verifiably - olacak )

Kazanan her X dakika da bir seçilecek.

Bu işlem tamamen otomatik olacak.

Rastgele ve otomatik işlemler için CAINLINK ORACLE kullanılacak.

Raffle.sol

```
pragma solidity ^0.8.7;

contract Raffle {
    constructor() {}

    function enterRaffle() {}

    function pickRandomWinner() {}
}
```

Tanımladığımız `i_entranceFee`, çekilişe katılmak için gerekli olan en düşük miktardır. Katılan bundan daha fazla ödeme yapmalıdır.

"i" harfi değişkenin değiştirilemez olduğunu vurgulamak içindir.  
private olması daha az maliyet sağlar.

```
contract Raffle {  
    // Storage Variables  
    uint256 private immutable i_entranceFee;  
  
    // Constructor  
    constructor(uint256 entranceFee) {  
        i_entranceFee = entranceFee;  
    }  
}
```

```
// View - Pure Functions  
function getEntranceFee() public view returns (uint256) {  
    return i_entranceFee;  
}
```

Kullanıcı katılım sağladığında gönderdiği miktar kontrol edilecek.

Eğer az ise hata oluşacak.

Bunun için özel hata oluşturulacak. Hata contract ın dışında, üstünde oluşturulur.

```
error Raffle_NotEnoughETHEntered();
```

```
function enterRaffle() {  
    if (msg.value < i_entranceFee) {  
        revert Raffle_NotEnoughETHEntered();  
    }  
}
```

Katılımcı adreslerini tutmak için dizi kullanacağız.

Ödeme işlemleri yapılacağı için bunlar payable olmalıdır ve private olmalıdır.

```
address payable[] private s_players;
```

```
function getPlayer(uint256 index) public view returns (address) {  
    return s_players[index];  
}
```

İşlem yapanların hesapları s\_players a eklenecektir.

```
function enterRaffle() public payable {  
    if (msg.value < i_entranceFee) {  
        revert Raffle_NotEnoughETHEntered();  
    }  
    s_players.push(payable(msg.sender));  
}
```

INTRODUCTION TO EVENTS 13.54.03



Dinamic bir nesne güncellendiğinde - diziler veya mapping gibi - event lar ile bunu takip edebiliriz.

EVM ler log ları emit edebilirler.

Blockchainde birşeyler yapıldığında bunlar loglanmaktadır. Log özel bir veri yapısıdır. Bunları blockchain node larından okuyabiliriz.

Log lar smart contract lara erişemezler ama smart contractlar loğlara erişebilirler. Bu yüzden de ucuzdurlar.

Event lar smart contract a yada hesaba bağlıdırlar. Ve bu da transaction içinde yayılmalarını sağlamaktadır. Bu eventları dinlemek çok yardımcı olur. Bir işlem yapılacağı zaman bu işlemin dinlenmesini sağlayabiliriz.

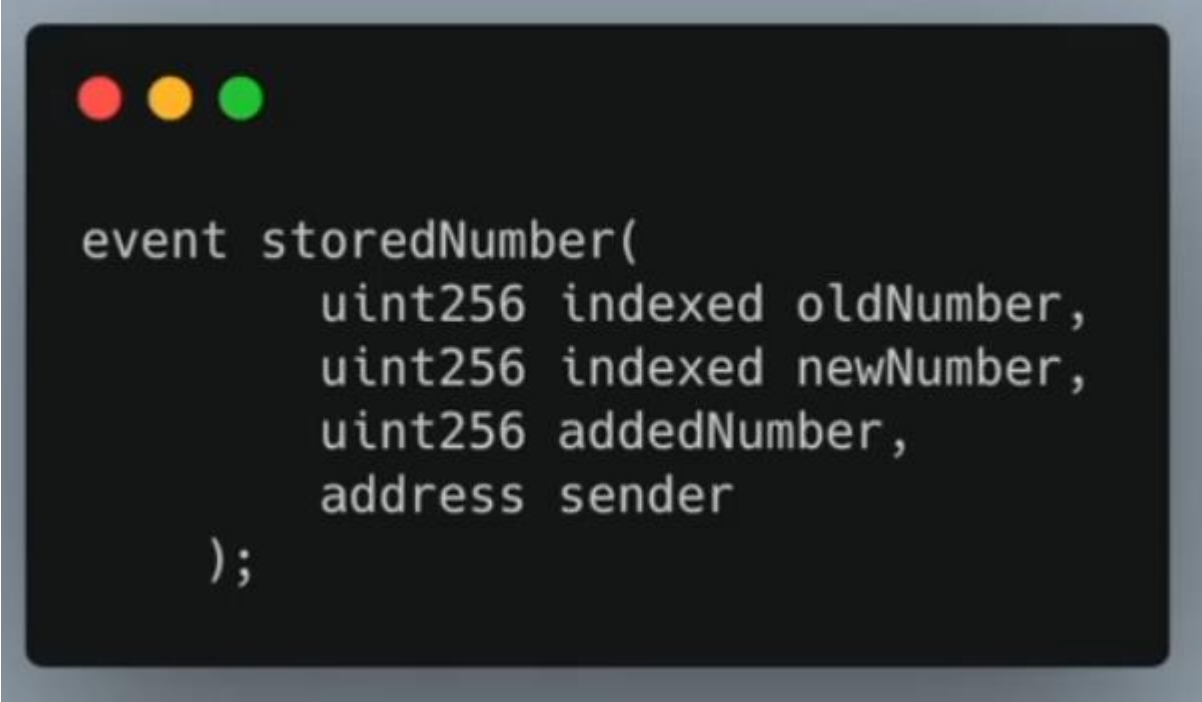
Bir websitesindeyken ve transaction tamamlandığında tekrar yüklenir, bunu gerçekleştiren event ı dinleyerek yapmaktadır. Bu işlem frontend için önemlidir. Ağdaki chain link, graph içinde önemlidir.

Chainlink node aslında bir random number almak için, API çağırmak için, vb. veri isteği olayını dinlemektedir

Birden fazla event varsa bunlar indexlenir.

Graphlar eventları dinler ve onları graph içerisinde saklar. Ve sonrasında sorgulamak kolay olur.

### Event yapısı



```
event storedNumber(  
    uint256 indexed oldNumber,  
    uint256 indexed newNumber,  
    uint256 addedNumber,  
    address sender  
);
```

Eventlarda iki önemli parametre vardır.  
index parametreleri  
non indexed parametreleri

3-üç tane index parametresi vardır. Index parametrelerini sorgulama non-index leri sorgulamaktan daha kolaydır.

# Indexed Parameters = Topics

Indexed Parameters are  
searchable

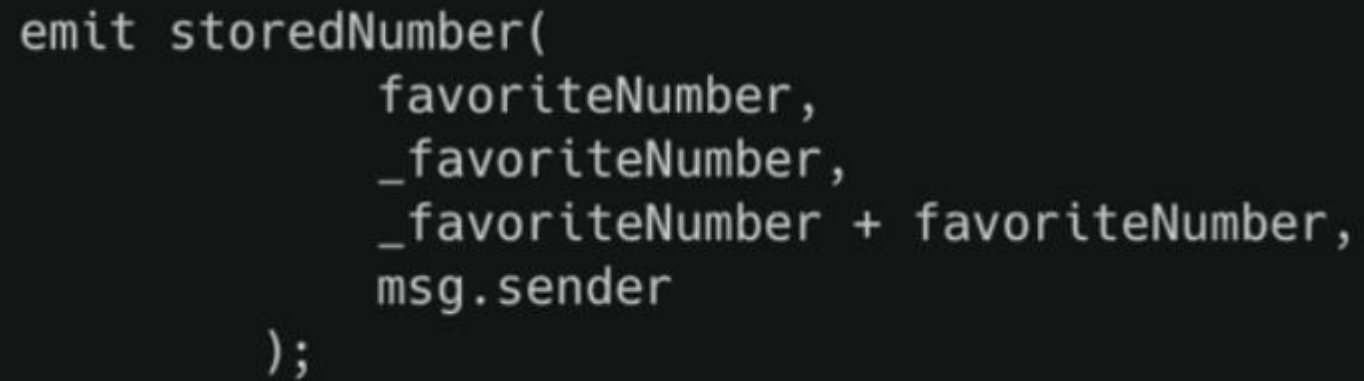
## REQUEST PARAMS

### FILTER OBJECT

- `address` *[optional]* - a string representing the address (20 bytes) to check for balance
- `fromBlock` *[optional, default is "latest"]* - an integer block number, or the string "latest", "earliest" or "pending"
- `toBlock` *[optional, default is "latest"]* - an integer block number, or the string "latest", "earliest" or "pending"
- `topics` *[optional]* - Array of 32 Bytes DATA topics. Topics are order-dependent.
- `blockhash` *:[optional]* With the addition of EIP-234, `blockHash` restricts the logs returned to the single block with the 32-byte hash `blockHash`. Using `blockHash` is equivalent to `fromBlock = toBlock =` the block number with hash `blockHash`. If `blockHash` is present in the filter criteria, then neither `fromBlock` nor `toBlock` are allowed.

Non indexed parametreler ABI ile encode edilmişlerdir ve decode etmek için ABI ın bilinmesi gerekir.

Event in emit edilmesi



```
emit storedNumber(  
    favoriteNumber,  
    _favoriteNumber,  
    _favoriteNumber + favoriteNumber,  
    msg.sender  
);
```

ÖRNEK

```
// SPDX-License-Identifier: MIT      You, a day ago • init commi
pragma solidity ^0.8.7;
contract SimpleStorage {
    uint256 favoriteNumber;
    event storedNumber(
        uint256 indexed oldNumber,
        uint256 indexed newNumber,
        uint256 addedNumber,
        address sender
    );

    function store(uint256 _favoriteNumber) public {
        emit storedNumber(
            favoriteNumber,
            _favoriteNumber,
            _favoriteNumber + favoriteNumber,
            msg.sender
        );
        favoriteNumber = _favoriteNumber;
    }

    function retrieve() public view returns (uint256) {
        return favoriteNumber;
    }
}
```

EVENT İÇEREN BİR TRANSACTION

### Transaction Details

Overview Logs (1) State

Transaction Receipt Event Logs

11

Address

0xd53410f44590ba924f7d2bc042a1e00853bd24cf

storedNumber

(index\_topic\_1 uint256 oldNumber, index\_topic\_2 uint256 newnumber,

Name uint256 addedNumber, address sender)

View Source

Topics

0

0xe304654f140653267f4d09df4968c896d7d6fd0b8e23c45f1db26a86cf930001

1

Dec

→ 0

2

Dec

→ 77

Data

addedNumber : 77

sender : 0x643315c9be056cdea171f4e7b2222a4ddab9f88d

Dec

Hex

index parametreleri

ABI encoded index parameteleri

EVENTS IN RAFFLE.SOL 14.00.47

## Events and Logging in Solidity

<https://youtu.be/KDYJC85eS5M>

<https://github.com/PatrickAlphaC/hardhat-events-logs>

Event oluşturulması ( constructor dan önce ). Eventler isimlendirilirken kullanılan fonksiyona göre isim vermek iyi olur.

```
/* Events */  
event RaffleEnter(address indexed player);
```

Kullanılması ( enterRaffle() içinde)

```
    s_players.push(payable(msg.sender));  
    emit RaffleEnter(msg.sender);  
}
```



INTRODUCTION TO CHAINLINK VRF ( Randomness in Web3 )  
14.02.30

<https://docs.chain.link/docs/get-a-random-number/>

Metamask ta rinkeby testnet i seç ve yukarıdaki bağlantıda yer alan aşağıdaki içeriğe git.

### 3. Open the Subscription Manager page.

Bu hesapta fon tutulmakta ve farklı chain ler arasında kullanılmaktadır.

Create Subscription

Cüsdan bağla

Create subscription tıkla. Bir miktar gas ücreti olarak işlem tamamlanmaktadır.

Ad funds diyerek işlem yapabilirsin.

## Add Funds

Add funds to your subscription. Your subscription is only billed after your contract receives a random value and you will never be billed more than the maximum price you specify. You can withdraw your funds at anytime. [Learn more](#)



Need LINK for testing? Visit the [Chainlink faucet](#) to receive testnet LINK.

Add funds (LINK)

Your wallet balance: 1094.0 LINK

Add funds

I'll do it later

## How to Create a Subscription?

Easy as 1, 2, 3

- ✓ Create a subscription with your wallet address.
- 2 Add funds to your subscription. You can always do it later.
- 3 Add consumers.



Need help or have questions? [Talk to an expert](#) or visit the [VRF webpage](#) to learn more

İşlemden önce ;

<https://faucets.chain.link/rinkeby> adresinden cüzdana LINK tokenları aktar.

Metamakstan token ları içe aktar.

LINK ethereum rikeby token sözleşme adresi

[0x01BE23585060835E02B77ef475b0Cc51aA1e0709](https://faucets.chain.link/rinkeby) şeklindedir.

adresini <https://docs.chain.link/docs/link-token-contracts/#ethereum> bağlantısından kontrol edebilirsiniz.

10 LINK fund olarak gönder

Bu tokenlar, her random number isteğinde gas olarak ödnecektir. Yani random number kullandıkça masraf buradan gidecek.

## Add Funds

Add funds to your subscription. Your subscription is only billed after your contract receives a random value and you will never be billed more than the maximum price you specify. You can withdraw your funds at anytime. [Learn more](#)



Need LINK for testing? Visit the [Chainlink faucet](#) to receive testnet LINK.

Add funds (LINK)

Your wallet balance: 20.0 LINK

Add funds

I'll do it later

Fund işleminden sonra add consumers ile nerede kullanacağını belirtmen gerekir. Add consumers tıklayınca consumer adresi ister. Bu aslında random number in kullanacağı contract adresidir.



## Funds added

Congratulations, 10 LINK added to your subscription.

View your transaction here:

0x2a0ad75b290ba7afff41d77cfc658d03f30bd8a8dee5517f8ccc866d6896347

Add consumers

Home / My Subscriptions / ID 6373 /

## Add Consumer

Add a consumer by specifying the contract address that will request a random v  
can add/remove consumer contract addresses later. [Learn more](#)

**i** Important: Your consumer contract must use the Subscription ID **6373** in the VRF request to make use of these funds.

Consumer address ⓘ

Add consumer

I'll do it later

Kullanımı test etmek için örnek bir contract oluşturabiliriz.

Yandaki contract üzerinde test edeceğiz.

```
// see https://docs.chain.link/docs/vrf-co  
bytes32 keyHash = 0xd89b2bf150e3b9e1344698
```

Contract taki keyhash değeri, random number kullanacağımız chain içindir. Kullanılan chain e göre, random number istediğimizde farklı gas limit ler belirleyecektir.

Bununla ilgili bilgilere <https://docs.chain.link/docs/vrf-contracts/#configurations> adresinden erişebiliriz.

```
// function.  
uint32 callbackGasLimit = 100000;
```

Ne kadar gas kullanacağınız belirtmek içindir. Uygun bir şekilde ayarlamamız gerekir.

Build and deploy the contract on Rinkeby.

1. Open the `VRfV2Consumer.sol` [\(link\)](#) contract in Remix.

Open in Remix

What is Remix?

```
// The default is 3, but you can set this higher.  
uint16 requestConfirmations = 3;
```

Doğrulama içindir, chain yada request e göre değiştirilebilir.

```
uint32 numWords = 2;
```

Bir istek ile almak istediğiniz random number sayısıdır.

Constructor da;

- coordinator adresi ( chainlink node ),
- link token adresi,
- subscriptionId

Bilgileri gereklidir.



Test etmek için, sözleşmeyi compile işlemini yap. DOĞRU CONTRACTI SEÇTİĞİNDEN EMİN OL ( VRFv2Consumer.sol )

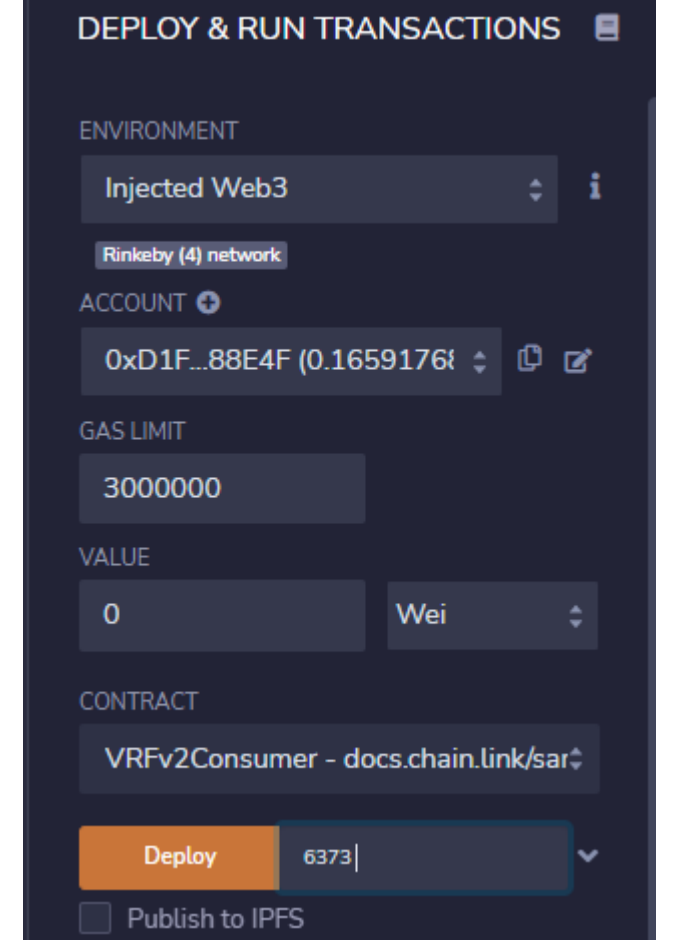
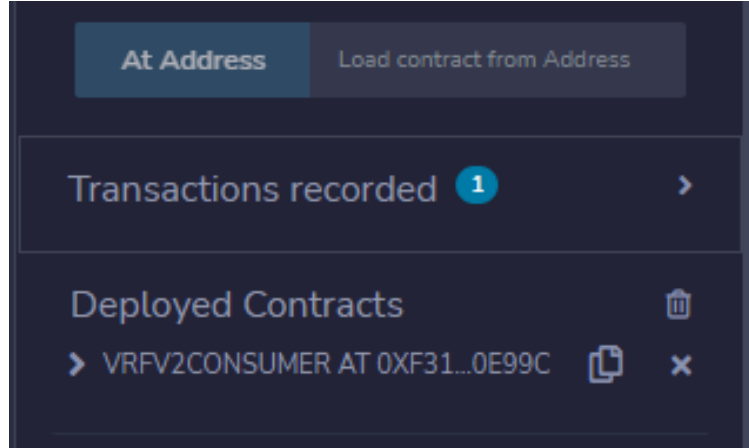
Deploy etmeden önce;

Remixte Injected Web3 seç . Bu metamask rinkeby tesnet hesabına bağlanır.  
Deploy kısmına subscription ID yi gir.

Deploy butonuna tıkla ve metamasktan onayla.

Deploy işleminden sonra;

Remix te Deployed contractstan adresi kopyala.



Add consumer a ekleyerek işlemi tamamla. Metamasktan işlemi onaylamayı unutma

[View subscription](#)

Yukarıdaki bağlantıdan da consumer ları görebilirsin.

## Add Consumer

Add a consumer by specifying the contract address that will request a random value. You can add/remove consumer contract addresses later. [Learn more](#)



**Important:** Your consumer contract must use the Subscription ID **6373** in the VRF request to make use of these funds.

Consumer address ⓘ

0xf3159a9382DD8A88f52592E6BAD961f22FC0e99d

[Add consumer](#)

[I'll do it later](#)

[Home](#) / [Ethereum Rinkeby](#) /

# Subscription

[Add Funds](#)


Status	ID ⓘ	Admin ⓘ	Consumers	Fulfillments ⓘ	Balance ⓘ
Active	6373	 0xd1f6...8e4f	1	0	10 LINK

[Cancel Subscription](#)

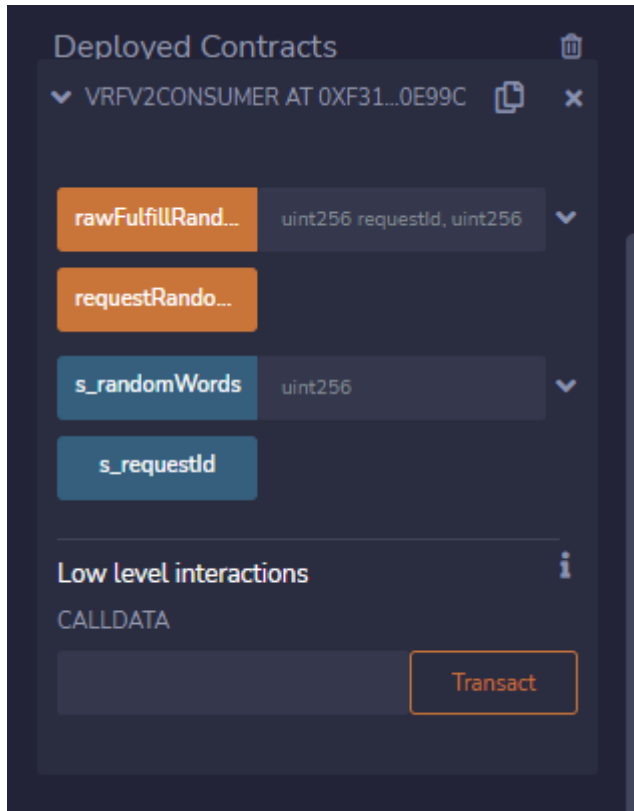
After canceling, the funds will be returned to the specified address

## Consumers

[Add consumer](#)

Address	Added	Last fulfillment	Total spent	Actions
 0xf315...e99c	June 12, 2022, 17:22 UTC	-	-	⋮

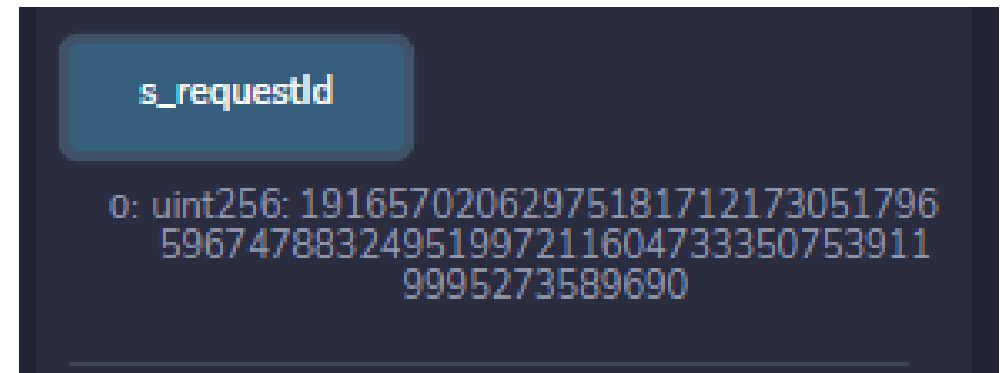
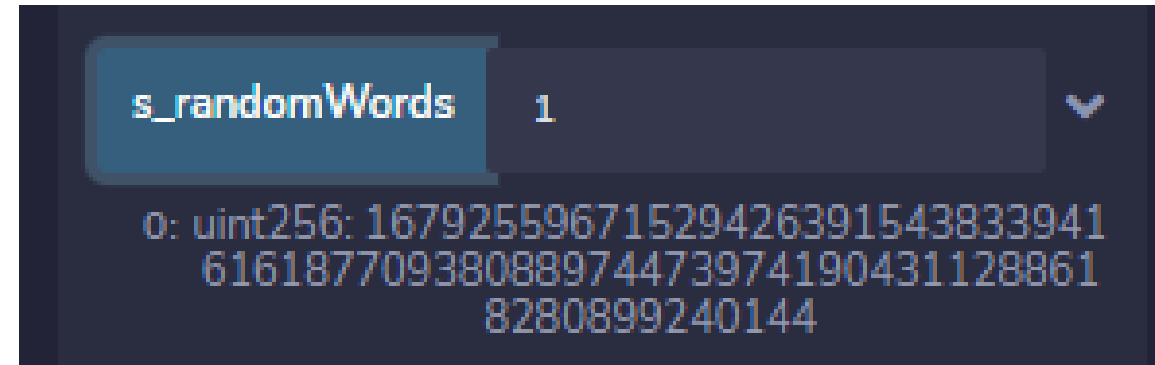
BÖYLECE RANDOM NUMBER ALMAK İÇİN GEREKLİ İŞLEMLERİ TAMAMLAMIŞ OLDUK.



RequestRandom tıklayınca iki adet random number alınacaktır. Tıkladıktan sonra metamasktan confirm etmemiz gerekir.

Transaction tammalandıktan sonra ORACLE fulfillRandomWords() fonksiyonunu çağıracaktır. Ve random numberlar contract ta saklanacaktır. Storage variable olarak saklanacaktır.




Sayıları görmek için s\_randomWords değişkenine index gönder.



<https://vrf.chain.link/rinkeby/> aboneliğe baktığımızda harcadığımız link leri görebiliriz.

## Consumers

Add consumer

Address	Added	Last fulfillment	Total spent	Actions
 0xf315...e99c 	June 12, 2022, 17:22 UTC	June 12, 2022, 17:29 UTC	0.3042163616386635 LINK	

IMPLEMENTING CHAINLINK VRF Introduction 14.09.54

Random number alamak için kullanacağımız `function pickRandomWinner()` {} fonksiyonu chainlink keepers network tarafından otomatik olarak çağrılacaktır.

External olarak tanımlanmalıdır. Public ten biraz daha ucuzdur.

Bu fonksiyonda

- random number isteğinde bulunacağız

- random number aldıktan sonra kullanacağız

- Burada kasıtlı olarak iki-2 transaction gerçekleşmektedir. Ve bu bir transaction ile almaktan daha iyidir.

- Çünkü tek transaction olunca insanlar brute force ile simule ederek bu transaction ı çağırabilirlerdi. Bu da kazananın kendileri olması için zorlayabilirlerdi. Ve iki transaction bunun önüne geçmektedir.

- Bu fonksiyon request ediyor. İkinci fonksiyonda random number gönderiliyor.

Random number gelince kazanan belirlenmiş olacak ve para gönderilecek.

Fonksiyon adını daha belirgin olsun diye requestRandomWinner olarak değiştir.

İkinci fonksiyon da fulfillRandomWords olacak. Bu fonksiyon override edilecektir.

Bunu kullanmak için chainlink kodunu da import etmemiz gerekir.

## İMPORT İŞLEMİNİ YAP

```
/* Imports */  
import "@chainlink/contracts/src/v0.8/VRFConsumerBaseV2.sol";
```

yarn add --dev @chainlink/contracts    ÇALIŞTIR.

Raffle contractımızı VRFconsumerBase den inherit etmemiz gerekir.

[node\\_modules/@chainlink/contracts/src/v0.8/VRFConsumerBaseV2.sol](#) adresinden baktığınızda fulfillRandomWords() fonksiyonunun VIRTUAL olarak tanımlandığını göreceksiniz. Bu override edilebilir olduğunu gösterir.

Kullanacağımız VR coordinator adresi. bu fonksiyonunu çağıracağını bilmektedir. Bu yüzden de override ederek kullanacağız ve coordinator tarafından çağrılacak.



INHERIT işlemini yap

```
contract Raffle is VRFConsumerBaseV2 {
```

FONKSİYONU OVERRIDE ET

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

CONSTRUCTORA GEREKLİ PARAMETRELERİ GÖNDER

```
// Constructor  
    constructor(address vrfCoordinatorV2, uint256 entranceFee)  
VRFConsumerBaseV2(vrfCoordinatorV2) {  
    i_entranceFee = entranceFee;  
}
```

yarn hardhat compile → kontrol et. (Compiled 2 Solidity files successfully ) BAŞARILI

HARDHAT SHORTHAND 14.14.32

<https://hardhat.org/guides/shorthand>

yarn global add hardhat-shorthand

Hardhat i "hh" olarak kullanabiliriz.

hh compile gibi

IMPLEMENTING CHAINLIN VRF The Request 14.15.32

Coordinator adresini almak için constructor da fulfillRandomWords() un çağırılması gerekir. Bunun içinde interface kullanılmaktadır.

Coordinator adresi state variable olarak saklanacak.

INTERFACE İMPORT ET

```
import "@chainlink/contracts/src/v0.8/interfaces/VRFCoordinatorV2Interface.sol";
```

Constructor a gidecek değişkeni tanımla

```
VRFCoordinatorV2Interface private immutable i_vrfCoordinator;
```

Constructor da atamasını yap

```
i_vrfCoordinator = VRFCoordinatorV2Interface(vrfCoordinatorV2);
```

gasLane ( gashash ) değıştkenini ayarla

```
bytes32 private immutable i_gasLane;
```

Açıklamaları burada

<https://docs.chain.link/docs/get-a-random-number/>

Constructor a ekle.

```
uint256 entranceFee,bytes32 gasLane  
...  
i_gasLane = gasLane;
```

Subscription id için işlemleri yap

```
uint64 private immutable i_subscriptionId;
```

Constructor a ekle.

```
uint64 subscriptionId
```

```
i_subscriptionId = subscriptionId;
```

```
uint16 private constant REQUEST_CONFIRMATIONS = 3;
```

CALLBACKGASLIMIT AYARLA: fulfillRandomWords çağrılırken ödenen gas ı belirlemek içindir. Bu bizi fazla gas ödemekten kurtarır.

```
uint32 private immutable i_callbackGasLimit;
```

Constructora ekle

```
uint32 callbackGasLimit
```

```
    i_callbackGasLimit = callbackGasLimit;
```

ALINACAK RANDOM NUMBER SAYISINI AYARLA

```
uint16 private constant NUM_WORDS = 1;
```

Request fonksiyonunu yaz:

```
function requestRandomWinner() external {  
    i_vrfCoordinator.requestRandomWords(  
        i_gasLane,  
        i_subscriptionId,  
        REQUEST_CONFIRMATIONS,  
        i_callbackGasLimit,  
        NUM_WORDS  
    );  
}
```

RequestRandomWords uint256 request id döndürmektedir ve bu unique yani tektir. Bu ide de bu isteği kimin gönderdiği ve diğer bilgileri içerir. Bunu saklamak için değişkene atamalıyız.



```
function requestRandomWinner() external {  
    uint256 requestId = i_vrfCoordinator.requestRandomWords(  
        i_gasLane,  
        i_subscriptionId,  
        REQUEST_CONFIRMATIONS,  
        i_callbackGasLimit,  
        NUM_WORDS  
    );  
}
```

Yeni bir event oluşturun.

```
event RequestedRaffleWinner(uint256 indexed requestId);
```

Event i kullan

```
function requestRandomWinner() external {  
    uint256 requestId = i_vrfCoordinator.requestRandomWords(  
        i_gasLane,  
        i_subscriptionId,  
        REQUEST_CONFIRMATIONS,  
        i_callbackGasLimit,  
        NUM_WORDS  
    );  
    emit RequestedRaffleWinner(requestId);  
}
```

IMPLEMENTING CHAINLINK VRF The Fulfil

14.22.57

Random numara gelince, oyuncular arasından rastgele bir kazanan seçeceğiz.

Bu işlem için MODULO denen bir fonksiyon ile yapacağız.

Gelen random number aşağıdaki gibidir.

```
// 786970238456976893421970832539624178-27569870418798562470824-1079652808-413072|
```

<https://docs.soliditylang.org/en/v0.8.14/types.html?highlight=modulo#modulo>

Modulo fonksiyonu mod alma gibi çalışmaktadır.

```
function fulfillRandomWords(uint256 requestId, uint256[] memory randomWords) internal
override {
    uint256 indexOfWinner = randomWords[0] % s_players.length;
    address payable recentWinner = s_players[indexOfWinner];
}
```

Burada seçtimiz kazanan doğrulanabilir bir veridir.

```
// Lottery Variables
address private s_recentWinner;
```

fulfillRandomWords()

```
    address payable recentWinner = s_players[indexOfWinner];
    s_recentWinner = recentWinner;
}
```

```
function getRecentWinner() public view returns (address) {  
    return s_recentWinner;  
}
```

fulfillRandomWords -- parayı gönder ve başarısız olursa hata ver

```
(bool success, ) = recentWinner.call{value: address(this).balance}("");  
if (!success) {  
    revert Raffle_TransferFailed();  
}
```

```
error Raffle_TransferFailed();
```

Önceden kazanaları tutmak için event kullan.

```
event WinnerPicked(address indexed winner);
```

```
fulfillRandomWords()
```

```
    if (!success) {  
        revert Raffle_TransferFailed();  
    }  
    emit WinnerPicked(recentWinner);  
}
```

Request id yi kulanmadığımız için hata olmaktadır. Bunu engellemek için aşağıdaki yapıyı kullanırız.

```
function fulfillRandomWords(  
    uint256,  
    /* requestId */  
    uint256[] memory randomWords
```

hh compile

BAŞARILI



INTRODUCTION TO CHAINLINK KEEPERS

14.28.27

Kazananı seçme işlemini de belli bir zamana göre kod ile yapmamız gerekir. Bu fonksiyonları bizim çağırmamamız gerekir. Bunun için chainlink keepers kullanacağız.

Kazananı seçme işlemini ;

    Zamana göre

    Bir ürünün miktarı ya da fiyatına göre,

    Hesaptaki paraya göre,

Vb. durumlarda akıllı contract ın çalışmasını sağlayabiliriz.

<https://docs.chain.link/docs/chainlink-keepers/introduction/>

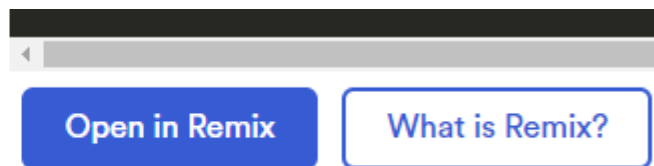
<https://docs.chain.link/docs/chainlink-keepers/compatible-contracts/>

Öncelikle chainlink keepers a uygun bir contract geliştirmemiz gerekir. Bunun için iki fonksiyon kullanılmaktadır.

## Functions

Function Name	Description
<code>checkUpkeep</code>	Runs off-chain at every block to determine if the <code>performUpkeep</code> function should be called on-chain.
<code>performUpkeep</code>	Contains the logic that should be executed on-chain when <code>checkUpkeep</code> returns true.

Yukarıdaki bağlantıda bulunan contract ı remix ile aç



`function checkUpkeep()` → Özel bir metottur. Kapalı chain hesaplaması yapılmaktadır. Yani off olan yani kapalı bir ağda bir node üzerinden keeper ağına kanal oluşturmaktadır. Kullanılan gas gerçek gas değildir. Sadece chainlink node üzerinde çalışmaktadır.

Çalıştıktan upkeep döndürdükten sonra `performUpkeep()` metodu chain üzerinde çalışmaya başlar.

Yani kapalı chain üzerinde veri üretilir ve açık chain e gönderilir.

`performUpkeep()` metodu da bu verilerin doğruluğunu kontrol eder. Ve sonrasında chain in state durumunda değişiklik yapar.

Buradaki örnek KOVAN network üzerinde kullanılmaktadır.

Remiz te

metamasktan kovan network seç  
injected web3 seç  
Contract ı seç  
Compile et

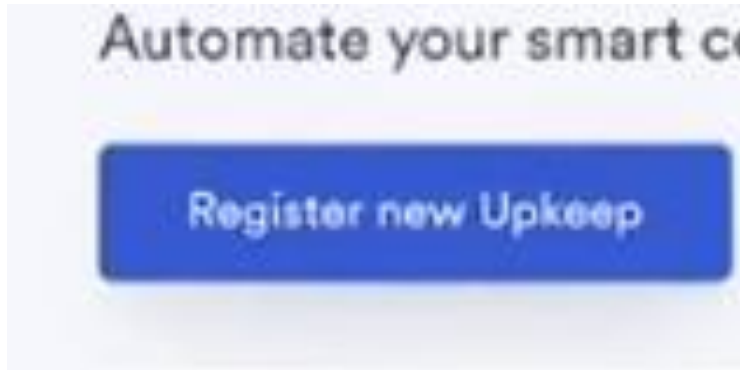
Deploy işleminde 30 değerini constructor a gönder. 30 sn sonra contract tekiklenecek demektir.

Kovan ağına para göndermen gerekir.

Yayınlanan kontraktın adresi alınır ve Chainlink Keepers uygulamasına eklenir.

<https://keepers.chain.link/> adresine git

Metamask bağla



 Need LINK for testing? Visit the Chainlink Kovan Faucet to receive 100 test LINK.


Email address

devrel@smartcontract.com

Enter your email address so we can send important notifications about your Upkeep.

Upkeep address

0xF0710EE3Dd97bfF2cD03149c1F6e6B2489520A13

 Unable to verify if this is a Keeper compatible contract.

Gas limit

200000

Amount of gas to provide the target contract when performing Upkeep. This will impact minimum balance requirements, and should be approximately the maximum amount of gas the transaction

Minimum gas

Starting balance (LINK)

5

Upkeep name

Counter Contract

Select a unique name for your Upkeep.

Admin address

0x3E77Fd1B4d4176CA9d54dB60f132FbB88BFA43CA

Address to cancel Upkeep and withdraw remaining funds.

Check data (Hexadecimal) *Optional*

Pass static data into your checkUpkeep function. This will be converted to bytes. See [docs](#) for details.

Belli bir miktar fund ödemesi yapman gerekir.

## Details

Upkeep ID

492

Admin address

 0x3E77Fd1B4d4176CA9d54dB60f132FbB88BFA43CA 

Copy address to clipboard

Gas limit

200,000

Date added

16 August 2021, 05:08

Last keeper

-

Check data (Base16)

0x

30 saniye sonra counter değeri kendiliğinden bir-1 olacaktır.



IMPLEMENTING CAHINLINK KEEPERS checkUpkeep 14.34.47

checkUpkeep() ve performUpkeep() fonksiyonlarını, contractı otomatikleştirmek için contracta ekleceğiz.

checkUpkeep() → Random number almak için zamanı kontrol etmek ve kazananı güncellemek ve fund ları kazanana göndermek için kullanılır.

FONKSİYONLARI YAZMADAN ÖNCE INTERFACELERİ İMPORT ETMELİ VE INHERIT ETMELİYİZ.

```
import "@chainlink/contracts/src/v0.8/interfaces/KeeperCompatibleInterface.sol";
```

```
contract Raffle is VRFConsumerBaseV2, KeeperCompatibleInterface{
```

```
function checkUpkeep(bytes calldata checkData) external override {}
```

checkData değişkeni checkupkeep çağrıldığında bize gerekli olan bütün herşeyi sağlamaktadır. Byte türünde olması, başka fonksiyonları çağırmak için özelleştirilebileceğini gösterir. Bu pek çok avantaj sağlamaktadır.

Şu anda kullanayacağımız için /\* \*/ içerisinde belirtiyoruz. Bu hata vermemesini sağlar. Fakat fonk parametresi olarak kalmaya devam eder, gerekli olduğu için.

CHECKUPKEEP():

Bu fonksiyonu CHAINLINK KEEPERS çağırarak  
upkeepNeeded parametresinin TRUE olduğu kontrol edilecek yani yeni bir random number gönderilecek mi diye kontrol edilecek  
TRUE ise yeni bir random numbe gönderilecek.

UPKEEPNEEDED in TRUE olması için:

Gerekli sürenin dolması gerekir.  
Contract in en az bir oyuncuya ve belli bir ETH ye sahip olması gerekir.  
Keeper hesabı LINK ile fonlanmış olması gerekir ( random number isterken gas ücreti olarak kullanılır yani chainlinke ödeme yaparız )  
Lottery "open" olması gerekir. Kazananı seçerken contracta yeni bir kişinin katılmaması gerekir. Bunu bir state değişkeni ile kontrol edeceğiz.

ENUMS 14.38.51

## CONTRACT DURUMU İÇİN ( AÇIK - KAPALI ) DEĞİŞKEN TANIMLAMA;

Önce özel enum tipi oluştur. Contract adından hemen sonra

```
/* Type Declarations */  
enum RaffleState{  
    OPEN,  
    CALCULATING  
} // uint256 0 = OPEN, 1 = CALCULATING
```

Lottery değişkenlerinde değişkeni oluştur.

```
RaffleState private s_raffleState;
```

Contract yayınlanınca katılımın açık olması için başlangıç değerini 0 olarak ayarla. CONSTRUCTORa parametre olarak gönderilmeyecek

```
s_raffleState = RaffleState.OPEN;
```

Oyuncu giriş yaparkende contract durumu ( açık - kapalı ) kontrol edilmedir. enterRaffle() fonk. ekle

```
if (s_raffleState != RaffleState.OPEN) {  
    revert Raffle__NotOpen();  
}
```

```
error Raffle__NotOpen();
```

Hatasını tanımla

Random number istenirken state i kapalı yapacağımız için requestRandomWinner da bu durumu kapatmamız gerekir. RequestRandomNumber() a ekle

```
s_raffleState = RaffleState.CALCULATING;
```

Kazanan belli olduktan sonra da durumu open yapmamız gerekir. fulfillrandomWords() e ekle.

```
s_recentWinner = recentWinner;  
s_raffleState = RaffleState.OPEN;
```

Kazanandan sonra katılımcıları resetlememiz gerekir.

```
s_raffleState = RaffleState.OPEN;  
s_players = new address payable[](0);
```

Zamanı tutmak için contract yayınlanınca başlangıç zamanını belirlemeliyiz. Kontract değişkeni tanımla ve constructor da ayarla ( parametre olarak gönderilmeyecek )

```
uint256 private s_lastTimeStamp;
```

```
s_lastTimeStamp = block.timeStamp;
```

Çekiliş süresi için diğer bir değişken tanımlamamız gerekir. ( interval gibi )

```
uint256 private immutable i_interval;
```

Contstructora paametre olarak ekle.

```
uint256 interval
```

```
i_interval = interval
```



Ve son olarak checkUpkeep() fonk aşağıdaki gibi olur.

```
function checkUpkeep(bytes calldata checkData)
    external
    override
    returns (
        bool upkeepNeeded,
        bytes memory /* performData */
    )
{
    bool isOpen = (RaffleState.OPEN == s_raffleState);
    bool timePassed = ((block.timestamp - s_lastTimeStamp) > i_interval);
    bool hasPlayers = (s_players.length > 0);
    bool hasBalance = address(this).balance > 0;
    upkeepNeeded = (isOpen && timePassed && hasPlayers & hasBalance);
}
```

IMPLEMENTING CHAINLINK KEEPERS performUpkeep  
14.47.16

Contract ta yazdığımız requestRandomWinner() fonksiyonu checkUpkeep() tarafından otomatik çağrılıyordu. Bu fonksiyon chainlink keper da performUpkeep() fonksiyonudur.

Bu yüzden adını performUpkeep() olarak değiştirelim ve dönüştürelim.

Şu anda performUpkeep() fonksiyonun herkes çağırabiliyor. Bu değiştirmemiz gerekir. Bunun sadece checkUpkeep true iken çalışması gerekir.

Önce checkUpkeep() i public yap

```
function checkUpkeep(bytes calldata checkData)
    public
    override
```

performUpkeep() i checkUpkeep e göre ayarla  
True olduğunu kontrol et  
Değilse hata üret

Bu hata üretildiğinde sebebini bilmek için veriler gönderilecek. Gönderilecek veriler parametre olarak belirlenir.

```
error Raffle__UpkeepNotNeeded(uint256 currentBalance, uint256 numPlayers, uint256 raffleState);
```

performUpkeep() fonksiyonu

```
function performUpkeep(  
    bytes calldata /* performData */  
) external override {  
    (bool upkeepNeeded, ) = checkUpkeep("");  
    if (!upkeepNeeded) {  
        revert Raffle__UpkeepNotNeeded(  
            address(this).balance,  
            s_players.length,  
            uint256(s_raffleState)  
        );  
    }  
    s_raffleState = RaffleState.CALCULATING;  
    uint256 requestId = i_vrfCoordinator.requestRandomWords(  
        i_gasLane,  
        i_subscriptionId,  
        REQUEST_CONFIRMATIONS,  
        i_callbackGasLimit,  
        NUM_WORDS  
    );  
    emit RequestedRaffleWinner(requestId);  
}
```

KAZANAN BELLİ OLUNCA TIMESTAMP SIFIRLANMALIDIR.

`fulfillRandomWords()` fonksiyonunda

```
s_players = new address payable[](0);  
s_lastTimeStamp = block.timestamp;
```

CODE CLEAN UP    14.50.36

## Contract adından önce açıklama ekleme - geliştiriciler için

```
/** @title A sample Raffle Contract
 * @author Celal AKSU from Patric Collins lessons
 * @notice This contract is for creating an untamperable decentralized smart contract
 * @dev This implements Chainlink VRF v2 and Chainlink Keepers
 */
```



Birkaç view function ekle

```
function getRaffleState() public view returns (RaffleState) {  
    return s_raffleState;  
}  
  
function getNumWords() public view returns (uint256) {  
    return NUM_WORDS;  
}
```

hh compile

```
patrick@iMac: [~/hh-fcc/hardhat-smartcontract-lottery-fcc] $ hh compile  
Error HH404: File @chainlink/contracts/src/v0.8.interfaces/KeeperCompatibleInterface.sol, imported from contracts/Raffle.sol, not found.
```

```
For more info go to https://hardhat.org/HH404 or run Hardhat with --show-stack-traces  
patrick@iMac: [~/hh-fcc/hardhat-smartcontract-lottery-fcc] $ █
```

```
tracts/src/v0.8.interfaces/KeeperC
```

Burdaki yazım hatasından kaynaklanmaktadır.

```
eemcs@DESKTOP-LJJ06I:~/freecodecamp/hardhat-smartcontract-lottery-cc$ hh compile
TypeError: Invalid type for argument in function call. Invalid implicit conversion from literal_string "" to bytes calldata requested.
--> contracts/Raffle.sol:119:45:
    |
119 |         (bool upkeepNeeded, ) = checkUpkeep("");
    |                                         ^^

Error HH600: Compilation failed
```

```
function checkUpkeep(bytes calldata checkData)
    public
```

Calldata -checkData string olduğu için - → memory olmalıdır ve checkData kullanılmadığı için aşağıdaki yapıda olmalıdır.

```
function checkUpkeep(
    bytes memory /* checkData */
)
```

Compile ile gelen uyarılar. checkUpkeep() fonk view yapılmamasının özel bir sebebi vardır. Sonradan değinilecek.

```
eemcs@DESKTOP-LJJC06I:~/freecodecamp/hardhat-smartcontract-lottery-cc$ hh compile
Warning: Unnamed return variable can remain unassigned. Add an explicit return with value to all non-reverting code paths or name the variable.
--> contracts/Raffle.sol:108:13:
   |
108 |         bytes memory /* performData */
   |         ^^^^^^^^^^^^^
Warning: Function state mutability can be restricted to view
--> contracts/Raffle.sol:101:5:
   |
101 |     function checkUpkeep(
   |     ^ (Relevant source part starts here and spans across multiple lines).
Warning: Function state mutability can be restricted to pure
--> contracts/Raffle.sol:175:5:
   |
175 |     function getNumWords() public view returns (uint256) {
   |     ^ (Relevant source part starts here and spans across multiple lines).

Compiled 2 Solidity files successfully
eemcs@DESKTOP-LJJC06I:~/freecodecamp/hardhat-smartcontract-lottery-cc$
```

```
Warning: Function state mutability can be restricted to pure
--> contracts/Raffle.sol:175:5:
   |
175 |     function getNumWords() public view returns (uint256) {
   |     ^ (Relevant source part starts here and spans across multiple lines).
```

NUM\_WORDS aslında bytecode içerisinde yer alır. Sabit değişkenler teknik olarak storage den okunmaz. O yüzden PURE function olmalıdır.

Birkaç fonksiyon daha ekle

```
function getNumberOfPlayers() public view returns (uint256) {  
    return s_players.length;  
}  
  
function getLatestTimeStamp() public view returns (uint256) {  
    return s_lastTimeStamp;  
}  
  
function getRequestConfirmations() public pure returns (uint256) {  
    return REQUEST_CONFIRMATIONS;  
}
```

DEPLOYING RAFFLE.SOL      14.56.00

Deploy / O1-deploy-raffle.js oluştur.

```
JS hardhat.config.js X  Raffle.sol 2  JS O1-deploy-raffle.js X  VRFConsumerBaseV2.sol
1  module.exports = async function ({ getNamedAccounts, deployments }) {
2      ...
3      const { deploy, log } = deployments
4      const { deployer } = await getNamedAccounts()
5  }
```



```
JS hardhat.config.js X  Raffle.sol 2  JS O1-deploy-raffle.js  VRFCConsumer

8
9 /**
10  * @type import('hardhat/config').HardhatUserConfig
11  */
12 module.exports = {
13   solidity: "0.8.7",
14   namedAccounts: {
15     deployer: {
16       default: 0,
17     },
18     player: {
19       default: 1,
20     },
21   },
22 }
23
```

Ağdaki sıfırncı hesap deployer olacak, birinci hesap ta oyuncu olacak. Bu test ve deploy için hesapların ayarlanmasıdır.

Açıklamaları önceki slayttadır.

```
hardhat.config.js x .env x Raffle.sol 2 JS O1-deploy-raffle.js VRFConsumerBaseV2.sol  
RINKEBY_RPC_URL=https://eth-rinkeby.alchemyapi.io/v2/i0fx4ZF4IN_vobajVDYvlSCjDrgTRjvf  
PRIVATE_KEY=d64eecf0eb8a06d1b43a2aa0e9dff0052807d707dd310d429a4b7dc93c9c0bd6  
ETHERSCAN_API_KEY=QJZRWKXAW2HN7644S3AG43GE6TIK4GFRQ5  
COINMARKETCAP_API_KEY=f85e9459-346e-4257-97fe-1a2039eecd70
```

```
JS hardhat.config.js x .env Raffle.sol 2 JS O1-deploy-raffle.js VRFConsumerBaseV2.sol  
12  
13 const RINKEBY_RPC_URL = process.env.RINKEBY_RPC_URL  
14 const PRIVATE_KEY = process.env.PRIVATE_KEY  
15 const ETHERSCAN_API_KEY = process.env.ETHERSCAN_API_KEY  
16 const COINMARKETCAP_API_KEY = process.env.COINMARKETCAP_API_KEY  
17
```

JS hardhat.config.js X

⚙ .env

💎 Raffle.sol 2

JS 01-dep

```
17
18 module.exports = {
19   solidity: "0.8.7",
20   networks: {
21     hardhat: {
22       chainId: 31337,
23       blockConfirmations: 1,
24     },
25     rinkeby: {
26       chainId: 4,
27       blockConfirmations: 6,
28       url: RINKEBY_RPC_URL,
29       accounts: [PRIVATE_KEY],
30     },
31   },
32   namedAccounts: {
```

t.config.js X

⚙️ .env

💎 Raffle.sol 2

JS O1-deploy-raffle.js X

💎 VRFConsumerBase

```
module.exports = async function ({ getNamedAccounts, deployments }) {  
  const { deploy, log } = deployments  
  const { deployer } = await getNamedAccounts()  
  
  const raffle = await deploy("Raffle", {  
    from: deployer,  
    args: [],  
    log: true,  
    waitConfirmations: network.config.blockConfirmations || 1,  
  })  
}
```

vrfCoordinatorV2 adresini chain dışından alacağımız için mock kullanmamız gerekir. Kullanacağımız adrese aşağıdaki bağlantıdan erişebilirsiniz

docs.chain.link/docs/vrf-contracts/

NEW

The Chainlink Hackathon kicks off April 22! [Sign up today to compete for \\$480k+ in prizes.](#)

Harmony Data Feeds

Optimism Data Feeds

Moonriver Data Feeds

USING RANDOMNESS

Introduction to Chainlink VRF

Get a Random Number

Example Contracts

Security Considerations

Best Practices

Contract Addresses

Migrating to VRF v2

CONNECT TO ANY API

Rinkeby Faucets

Testnet LINK is available from <https://faucets.chain.link/rinkeby>  
Testnet ETH is available from: <https://faucets.chain.link/rinkeby>  
Backup Testnet ETH Faucets: <https://rinkeby-faucet.com/>, <https://app.mycrypto.com/faucet>

Item	Value
LINK Token	0x01BE23585060835E02B77ef475b0Cc51aA1e0709
VRF Coordinator	0x6168499c0cFfCaCD319c818142124B7A15E857ab
30 gwei Key Hash	0xd89b2bf150e3b9e13446986e571fb9cab24b13cea0a43ea20a6049a85cc807cc
Premium	0.25 LINK

JS hardhat.config.js X

⚙️ .env

💎 Raffle.sol 2

JS O1-deploy-raffle.js

JS helper-hardhat-config.js X

💎

```
1  const networkConfig = {
2    4: {
3      name: "rinkeby",
4      vrfCoordinatorV2: "0x6168499c0cFfCaCD319c818142124B7A15E857ab",
5    },
6  }
7  |
```

JS hardhat.config.js

.env

Raffle.sol 2

JS 01-deploy-raffle.js

JS helper-hardhat-config.js

JS 00-deploy-mocks.js X

```
1 module.exports = async function ({ getNamedAccounts, deployments }) {  
2     const { deploy, log } = deployments  
3     const { deployer } = await getNamedAccounts()  
4     const chainId = network.config.chainId  
5 }  
6
```

JS 01-deploy-raffle.js X

JS helper-hardhat-config.js X

JS hardhat.config.js

JS

```
5     },  
6 }  
7 const developmentChains = ["hardhat", "localhost"]  
8 module.exports = {  
9     networkConfig,  
10    developmentChains,  
11 }  
12
```

JS 01-deploy-raffle.js X

JS helper-hardhat-config.js

JS hardhat.config.js

JS 00-deploy-mocks.js X

```
1  const { network } = require("hardhat")
2  const { developmentChains } = require("../helper-hardhat-config")
3
4  module.exports = async function ({ getNamedAccounts, deployments }) {
```

JS 01-deploy-raffle.js

JS helper-hardhat-config.js

JS hardhat.config.js

JS 00-deploy-mocks.js X

```
7      const chainId = network.config.chainId
8
9      if (developmentChains.includes(network.name)) {
10         log("Local network detected! Deploying mocks...")
11         // deploy a mock vrfcoordinator....
12     }
13 }
```



DEPLOYING RAFFLE.SOL Mock Chainlink & VRF Coordinator  
15.04.28

<https://github.com/smartcontractkit/chainlink/blob/develop/contracts/src/v0.8/mocks/VRFCoordinatorV2Mock.sol>

Bu contractı mock contract olarak kullanacağız.

Bunun için deploy/test/VRFCoordinatorV2Mock.sol contractı oluşturup bu contractı import edeceğiz.

A screenshot of a VS Code editor tab. The tab has a blue flame icon on the left, followed by the text "VRFCoordinatorV2Mock.sol" in orange, a small "1" in white, and a close button "X" in white on the right.

```
// SPDX-License-Identifier: MIT

pragma solidity ^0.8.7;


import "@chainlink/contracts/src/v0.8/mocks/VRFCoordinatorV2Mock.sol";
```

hh compile ile test et.

Mock contractın deploy edilmesi:

Contract deploy edilirken iki-2 tane parametre almaktadır.

import edilen mock contractına bakınca constructor da bunları görebiliriz.

 *VRFCoordinatorV2Mock.sol* .../mocks

```
constructor(uint96 _baseFee, uint96 _gasPriceLink) {  
    BASE_FEE = _baseFee;  
    GAS_PRICE_LINK = _gasPriceLink;  
}
```

Bunları değişken olarak tanımlayıp arg kısmına ekleyeceğiz.

BASE FEE: Random number istenirken verilen fee ücretidir. Oracle gas ücreti olarakta düşünebilirsiniz.

Contract Addresses	
Migrating to VRF v2	
CONNECT TO ANY API	
Introduction to Using Any API	
Coordinator	0x6168499C0CF1CaCD319C818142124B7A15E857ad
30 gwei Key Hash	0xd89b2bf150e3b9e13446986e571fb9cab24b13cea0a43ea20a6049a85cc807cc
Premium	0.25 LINK
Minimum	7

<https://data.chain.link/ethereum/mainnet/crypto-usd/eth-usd> sayfasından kontrol edilirse, mainnet üzerinde bu fee ücretleri sponsor olan protocoller tarafından karşılanmaktadır.

```
JS 00-deploy-mocks.js X
2  const { developmentChains } = require("../helper-hardhat-config")
3
4  const BASE_FEE = ethers.utils.parseEther("0.25") // 0.25 is the premium
5  // It costs 0.25 LINK
6
```

GAS\_PRICE\_LINK : Gas ücretine göre hesaplanan değerdir.

Ethereum dan random number isterseniz, gas çok çok fazla ( 1 milyon dolar gibi ) olabilir. Chain link nodeları, bize random veren ve harici hesaplama yapan chain link nodelarının gas fee ödemesine karşılık verirler. Chain link nodeları aslında bir tane gas ödemesini, upkeep ile random döndürüldüğünde vb. durumda yapar.

Bu da gerçek ağda fiyatın dalgalanmasından ve yüksek miktarlardan kurtarır.

JS 00-deploy-mocks.js X

```
4  const BASE_FEE = ethers.utils.parseEther( 0.25 ) // 0.25
5  // It costs 0.25 LINK
6  const GAS_PRICE_LINK = 1e9 //1000000000 LINK per gas.
7
```

JS 00-deploy-mocks.js X

```
10     const { deployer } = await getNamedAccounts()
11     const args = [BASE_FEE, GAS_PRICE_LINK]
12
13     if (developmentChains.includes(network.name)) {
14         log("Local network detected! Deploying mocks...")
15         // deploy a moc vrfcoordinator....
16         await deploy("VRFCoordinatorV2Mock", {
17             from: deployer,
18             log: true,
19             args: args,
20         })
21         log("Mocks deployed")
22         log("-----")
23     }
24 }
25 module.exports.tags = ["all", "mocks"]
26
```

MOCK CONTRACTI DEPLOY EDİLDİKTEN SONRA BU CONTRACT BİLGİLERİ RAFFLE DEPLOY İŞLEMİNDE KULLANILACAKTIR. BU BİLGİLERİ RAFFLE DEPLOY ALIYORUZ.

JS 00-deploy-mocks.js X

JS 01-deploy-raffle.js X

```
1  const { network, ethers } = require("hardhat")
2  const { developmentChains } = require("../helper-hardhat-config")
3
```

JS 00-deploy-mocks.js

JS 01-deploy-raffle.js X

```
6  const { deployer } = await getNamedAccounts()
7  let vrfcoordinatorV2Address
8
9  if (developmentChains.includes(network.name)) {
10     const vrfcoordinatorV2Mock = await ethers.getContract("VRFCoordinatorV2Mock")
11     vrfcoordinatorV2Address = vrfcoordinatorV2Mock.address
12 }
13 const args = []
14 const raffle = await deploy("Raffle", {
15     from: deployer,
16     args: args,
```

```
JS 00-deploy-mocks.js X JS 01-deploy-raffle.js X
2  const { developmentChains, networkConfig } = require("../helper-hardhat-config")
3
```

```
JS 00-deploy-mocks.js X JS 01-deploy-raffle.js X
7  const chainId = network.config.chainId
8  let vrfcoordinatorV2Address
```

TEST AĞINDA YAYINLANDIĞINDA MOCK CONTRACTA İHTİYAÇ OLMADIĞI İÇİN, TEST AĞINDAKİ COORDİNATOR ADRESİ KULLANILACAKTIR.

```
JS 00-deploy-mocks.js JS 01-deploy-raffle.js X
12  vrfcoordinatorV2Address = vrfcoordinatorV2Mock.address
13  } else {
14  vrfcoordinatorV2Address = networkConfig[chainId]["vrfCoordinatorV2"]
15  }
```



RAFFLE.SOL PARAMETRELERİ: Bunları helper-hardhat-config.js de tanımlayabiliriz. Çünkü ağa göre farklı veriler olacaktır.

entranceFee → contractın bulunduğu chain e göre değişir. Lottery katılanın ödeyeceği min miktardır.

```
JS 00-deploy-mocks.js X JS 01-deploy-raffle.js JS helper-hardhat-config.js X
1  const { ethers } = require("hardhat")
2
3  const networkConfig = {
4    4: {
5      name: "rinkeby",
6      vrfCoordinatorV2: "0x6168499c0cFfCaCD319c818142124B7A15E857ab",
7      entranceFee: ethers.utils.parseEther("0.01"),
8    },
9    31337: {
10     name: "hardhat",
11     entranceFee: ethers.utils.parseEther("0.01"),
12   },
}
```

JS 00-deploy-mocks.js X

JS 01-deploy-raffle.js X

JS helper-hardhat-config.js

16

17     `const entranceFee = networkConfig[chainId]["entranceFee"]`

18     `const args = [vrfcoordinatorV2Address, entranceFee]`

19     `const raffle = await deploy("Raffle", {`

20         `from: deployer,`

gasLane : <https://docs.chain.link/docs/vrf-contracts/> adresinden- AĞA GÖRE DEĞİŞİR

Harmony Data Feeds

Optimism Data Feeds

Moonriver Data Feeds

## USING RANDOMNESS

Introduction to Chainlink VRF

Get a Random Number

Example Contracts

Security Considerations

Best Practices

**Contract Addresses**

Migrating to VRF v2

## CONNECT TO ANY API

Introduction to Using Any API

### Rinkeby Faucets

Testnet LINK is available from <https://faucets.chain.link/rinkeby>

Testnet ETH is available from: <https://faucets.chain.link/rinkeby>

Backup Testnet ETH Faucets: <https://rinkeby-faucet.com/>, <https://app.mycrypto.com/faucet>

Item	Value
LINK Token	0x01BE23585060835E02B77ef475b0Cc51aA1e0709
VRF Coordinator	0x6168499c0cFfCaCD319c818142124B7A15E857ab
30 gwei Key Hash	0x8d89b2bf150e3b9e13446986e571fb9cab24b13cea0a43ea20a6049a85cc807cc
Premium	0.25 LINK

```
00-deploy-mocks.js X JS 01-deploy-affle.js JS helper-hardhat-config.js X
4      4: {
5          name: "rinkeby",
6          vrfCoordinatorV2: "0x6168499c0cFfCaCD319c818142124B7A15E857ab",
7          entranceFee: ethers.utils.parseEther("0.01"),
8          gasLane: "0xd89b2bf150e3b9e13446986e571fb9cab24b13cea0a43ea20a6049a85cc807cc",
```

```
JS 00-deploy-mocks.js X JS 01-deploy-affle.js JS helper-hardhat-config.js X
10     31337: {
11         name: "hardhat",
12         entranceFee: ethers.utils.parseEther("0.01"),
13         // Anything here, it dosent matter
14         gasLane: "0xd89b2bf150e3b9e13446986e571fb9cab24b13cea0a43ea20a6049a85cc807cc",
```

```
JS 00-deploy-mocks.js X JS 01-deploy-affle.js X JS helper-hardhat-config.js
18     const gasLane = networkConfig[chainId]["gasLane"]
19     const args = [vrfCoordinatorV2Address, entranceFee, gasLane]
20     const raffle = await deploy("Raffle", {
```

SUBSCRIPTION ID : Testnet ağı için vrf.chain.link adresinden almıştık. Fakat yerel ağ için işler biraz karışık. Bunu almayı ve fonlamayı programatik olarak yapacağız. Herhangi bir UI kullanmaya gerek kalmayacak.

VRFCoordinatorV2Mock.sol contractında createSubscription() fonksiyonu bulunmaktadır. Buradan alabiliriz. ID bu fonksiyonda bulunan emit edilen event ile alınmaktadır.

Fund lama işlemini ise link token ile gerçek bir network üzerinden yapmamız gerekir.

```
JS 00-deploy-mocks.js X JS 01-deploy-raffle.js X JS helper-hardhat-config.js
8 let vrfcoordinatorV2Address, subscriptionId
```

```
JS 00-deploy-mocks.js X JS 01-deploy-raffle.js X JS helper-hardhat-config.js
14 const transactionResponse = await vrfcoordinatorV2Mock.createSubscription()
15 const transactionReceipt = await transactionResponse.wait(1)
16 subscriptionId = transactionReceipt.events[0].subId
17 } else {
18     vrfcoordinatorV2Address = networkConfig[chainId]["vrfCoordinatorV2"]
19 }
20
21 const entranceFee = networkConfig[chainId]["entranceFee"]
```

## MOCK İÇİN SUBSCRIPTION ID ALMA VE FONLAMA

JS 00-deploy-mocks.js

JS 01-deploy-raffle.js X

JS helper-hardhat-config.js

```
3
4  const VRF_SUB_FUND_AMOUNT = ethers.utils.parseEther("30")
5  // 2 would work
6
7  module.exports = async function ({ getNamedAccounts, deployments }) {
```

JS 00-deploy-mocks.js

JS 01-deploy-raffle.js X

JS helper-hardhat-config.js

```
15  vrfcoordinatorV2Address = vrfcoordinatorV2Mock.address
16  // Get subscriptionid
17  const transactionResponse = await vrfcoordinatorV2Mock.createSubscription()
18  const transactionReceipt = await transactionResponse.wait(1)
19  subscriptionId = transactionReceipt.events[0].subId
20  // Fund to subscriptionid
21  await vrfcoordinatorV2Mock.fundSubscription(subscriptionId, VRF_SUB_FUND_AMOUNT)
22  } else {
23  vrfcoordinatorV2Address = networkConfig[chainId]["vrfCoordinatorV2"]
```

RINKEBY İÇİN

```
8         gasLane: "0xd89b2bf150e3b9  
9         subscriptionId: "0",  
10     },
```

```
    vrfcoordinatorV2Address = networkConfig[chainId]["vrfCoordinatorV2"]  
    subscriptionId = networkConfig[chainId]["subscriptionId"]  
}
```

```
const args = [vrfcoordinatorV2Address, entranceFee, gasLane, subscriptionId]  
const ref1 = await deploy("Ref1", [
```

CALLBACKGASLIMIT:

```
subscriptionId: "0", // will change  
callbackGasLimit: "500000",  
},
```

Rinkeby içinde hardhat için de aynı

```
const callbackGasLimit = networkConfig[chainId]["callbackGasLimit"]  
const args = [vrfCoordinatorV2Address, entranceFee, gasLane, subscriptionId, callbackGasLimit]  
const tx = await vrfCoordinatorV2.connect().transaction(...args, {value: 0})
```



INTERVAL :

```
callbackGasLimit: "500000",  
interval: "30",  
}
```

Rinkeby içinde hardhat için de aynı

```
const interval = networkConfig[chainId]["interval"]  
const args = [  
  vrfcoordinatorV2Address,  
  entranceFee,  
  gasLane,  
  subscriptionId,  
  callbackGasLimit,  
  interval,  
]
```

DOĞRULAMA İŞLEMLERİ    Kök dizinde utils/verify.js oluştur. Önceki projeden de kopyalayabilirsin.

```
const { run } = require("hardhat")

const verify = async (contractAddress, args) => {
  console.log("Verifying contract...")
  try {
    await run("verify:verify", {
      address: contractAddress,
      constructorArguments: args,
    })
  } catch (e) {
    if (e.message.toLowerCase().includes("already verified")) {
      console.log("Already Verified!")
    } else {
      console.log(e)
    }
  }
}

module.exports = { verify }
```

00-deploy-mocks.js

JS 01-deploy-raffle.js X

JS helper-hardhat-config.js

```
1  const { network, ethers } = require("hardhat")
2  const { developmentChains, networkConfig } = require("
3  const { verify } = require("../utils/verify")
4
```

JS 00-deploy-mocks.js

JS 01-deploy-raffle.js X

JS helper-hardhat-config.js

```
40
47      if (!developmentChains.includes(network.name) && process.env.ETHERSCAN_API_KEY) {
48          log("Verifying...")
49          await verify(raffle.address, args)
50      }
51      log("_____")
52  }
53  module.exports.tags = ["all", "raffle"]
54
```

hh deploy / yarn hardhat deploy

Mocks deployed

-----  
An unexpected error occurred:

```
Error: ERROR processing /home/eemcs/freecodecamp/hardhat-smartcontract-lottery-cc/deploy/01-deploy-raffle.js:
Error: invalid BigNumber value (argument="value", value=undefined, code=INVALID_ARGUMENT, version=bignumber/5.6.2)
    at Logger.makeError (/home/eemcs/freecodecamp/hardhat-smartcontract-lottery-cc/node_modules/@ethersproject/logger/src.ts/index.ts:261:28)
    at Logger.throwError (/home/eemcs/freecodecamp/hardhat-smartcontract-lottery-cc/node_modules/@ethersproject/logger/src.ts/index.ts:273:20)
    at Logger.throwArgumentError (/home/eemcs/freecodecamp/hardhat-smartcontract-lottery-cc/node_modules/@ethersproject/logger/src.ts/index.ts:277:21)
    at Function.BigNumber.from (/home/eemcs/freecodecamp/hardhat-smartcontract-lottery-cc/node_modules/@ethersproject/bignumber/src.ts/bignumber.ts:289:23)
```

```
subscriptionId = transactionReceipt.events[0].subId
```

```
subscriptionId = transactionReceipt.events[0].args.subId
```

```
deploying "VRFCoordinatorV2Mock" (tx: 0x1319418ccc7f88788f4cc6c636585b9e49e87dafc1fca4cccb96f87be089ac55)...: deployed
at 0x5FbDB2315678afecb367f032d93F642f64180aa3 with 1803306 gas
```

Mocks deployed

```
-----
deploying "Raffle" (tx: 0x69b3e90958bb495ce147ff4204bd8aa24cc947e3d39b325bb4abea9be44a279b)...: deployed at 0xCf7Ed3Acc
A5a467e9e704C703E8D87F634fB0Fc9 with 1197876 gas
```

UNIT TESTS

15.20.07

test/unit/Raffle.test.js      dosyasını oluştur.

COVERAGE İŞLEMİ ( ÜCRET HESABI ) BURADA DAHA DETAYLI OLARAK YAPILACAK

```
function getInterval() public view returns (uint256) {  
    return i_interval;  
}
```

Eksik olan fonsiyonu contract a ekle

Raffle.test.js

```
const { assert } = require("chai")  
const { network, getNamedAccounts, deployments, ethers } = require("hardhat")  
const { developmentChains, networkConfig } = require("../helper-hardhat-config")
```

```
!developmentChains.includes(network.name)
  ? describe.skip
  : describe("Raffle Unit Tests", async function () {

    let raffle, vrfcoordinatorV2Mock
    const chainId = network.config.chainId

    beforeEach(async function () {
      const { deployer } = await getNamedAccounts()
      await deployments.fixture(["all"])
      raffle = await ethers.getContract("Raffle", deployer)
      vrfcoordinatorV2Mock = await ethers.getContract("VRFCoordinatorV2Mock", deployer)
    })

    describe("constructor", async function () {
      it("Initializes the raffle correctly", async function () {
        const raffleState = await raffle.getRaffleState()
        const interval = await raffle.getInterval()
        assert.equal(affleState.toString(), "0")
        assert.equal(interval.toString(), networkConfig[chainId]["interval"])
      })
    })
  })
```

Constructor test

hh test  
yada  
yarn hardhat test

```
eemcs@DESKTOP-LJJ06I:~/freecodecamp/hardhat-smartcontract-lottery-cc$ hh test

Raffle Unit Tests
  constructor
    ✓ Initializes the raffle correctly

-----|-----|-----|-----|
| Solc version: 0.8.8 | Optimizer enabled: false | Runs: 200 | Block limit: 30000000 gas |
|-----|-----|-----|-----|
| Methods |
|-----|-----|-----|-----|
| Contract | Method | Min | Max | Avg | # calls | eur (avg) |
|-----|-----|-----|-----|-----|
| VRFCoordinatorV2Mock | createSubscription | - | - | 68632 | 2 | - |
```



Test işleminde her zaman gas raporu verilmeyebilir. Bu yüzden harthat config e gas reporter ayarlarını ekleyebiliriz.

```
gasReporter: {  
  enabled: true,  
  outputFile: "gas-report.txt",  
  noColors: true,  
  currency: "USD",  
  coinmarketcap: COINMARKETCAP_API_KEY,  
  token: "ETH",  
},
```

**enterRaffle() fonk testi. -- Birinci if bloğunun testi.**

Expect i import etmeyi unutma ( otomatik olmaz ise )

Hh test --grep "you don't pay enough" ile testi çalıştır.

```
describe("enterRaffle", async function () {
  it("reverts when you don't pay enough", async function () {
    await expect(raffle.enterRaffle()).to.be.revertedWith(
      "Raffle_NotEnoughETHEntered"
    )
  })
})
})
```

Raffle Unit Tests

enterRaffle

✓ reverts when you don't pay enough

1 passing (2s)

**enterRaffle()** fonk testi. -- ikinci if testi.

Önce raffleEntranceFee değerinin beforeEach() de alınması gerekir. Değişken global tanımlanmalıdır. Ayrıca deployer ında global tanımlanması gerekir.

```
let raffle, vrfcoordinatorV2Mock, raffleEntranceFee, deployer
```

beforeEach()

```
deployer = (await getNamedAccounts()).deployer
```

```
...
```

```
raffleEntranceFee = await raffle.getEntranceFee()
```

```
it("records players when they enter", async function () {  
  await raffle.enterRaffle({ value: raffleEntranceFee })  
  // Deployer ın doğru kayıt edildiğinden emin olmamız gerekir.  
  const playerFromContract = await raffle.getPlayer(0)  
  assert.equal(playerFromContract, deployer)  
})
```

Hh test --grep "records players when they enter"

```
eemcs@DESKTOP-LJJ06I:~/freecodecamp/hardhat-smartcontract-lottery-cc$ hh test --grep "records players when they enter"
```

```
Raffle Unit Tests
```

```
  enterRaffle
```

```
    ✓ records players when they enter
```

```
1 passing (2s)
```

TESTING EVENTS & CHAI MATCHERS 15:30:20

<https://ethereum-waffle.readthedocs.io/en/latest/matchers.html#emitting-events>

```
it("emits event on enter", async function () {  
    await expect(raffle.enterRaffle({ value: raffleEntranceFee })).to.emit(  
        raffle,  
        "RaffleEnter"  
    )  
})
```

```
eemcs@DESKTOP-LJJC86I:~/freecodecamp/hardhat-smartcontract-lottery-cc$ hh test --grep "emits event on enter"
```

Raffle Unit Tests

enterRaffle

✓ emits event on enter

1 passing (2s)

OPEN - CLOSE TEST : Open state durumuna ulaşmamız gerekir. performUpkeep te rafflestate calculating durumundadır. Burada checkupkeep in true değeri döndürdüğünü kontrol etmemiz gerekir. Çünkü performUpkeep sadece bu durumda çalışmaktadır. Aksi halde raffle upkeep gerekli değil durumunu üretir.

True değerini kontrol etmek için, keeper ağı ile kanal oluşturup checkupkeep in true olmasını bekleyeceğiz. True yaptığımızda da upkeep çalışacak ve contract durumu calculating olacak.

BU DURUMU OLUŞTURMAK İÇİN : ----- KONTROL ET BU KISMI ----- SONRAKİ SLAYTTA

Kontract için belli bir süre ayarladık. Bu süre gerçek bir kontractta günler gibi uzun bir zaman alabilir. Fakat test için o kadar bekleyemeyiz. Bunun için gerekli araçları kullanacağız.

Hardhat Network ünün Çalışması : <https://hardhat.org/hardhat-network/refence/>

JSON-RPC methods support kullanarak devam edeceğiz.

Burada farklı senaryoları denemek için kullanılan metotlar bulunmaktadır.

Special testing/ debugging methods ta bulunan evm\_increaseTime ( otomatik zamanı arttır ) ; evm\_mine ( otomatik block oluşturur) vb.

HARDHAT METHODS & "Time Travel" 15:32:46



BU DURUMU OLUŞTURMAK İÇİN : ----- KONTROL ET BU KISMI ----- SONRAKİ SLAYTTA

Kontract için belli bir süre ayarladık. Bu süre gerçek bir kontractta günler gibi uzun bir zaman alabilir. Fakat test için o kadar bekleyemeyiz. Bunun için gerekli araçları kullanacağız.

Hardhat Network ünün Çalışması : <https://hardhat.org/hardhat-network/reference/>

JSON-RPC methods support kullanarak devam edeceğiz.

Burada farklı senaryoları denemek için kullanılan metotlar bulunmaktadır.

Special testing/ debugging methods ta bulunan `evm_increaseTime` ( otomatik zamanı arttır ) ; `evm_mine` ( otomatik block oluşturur) vb.

```
let raffle, vrfcoordinatorV2Mock, raffleEntranceFee, deployer, interval
```

```
interval = await raffle.getInterval()
```

```
    it("doesn't allow entrance when raffle id calculating", async function () {  
        await raffle.enterRaffle({ value: raffleEntranceFee })  
        // Block oluşması için zamanı hızlandırıyoruz. Böylece gerekli süreyi beklememiz gerekmez  
        // ve chain durumu calculating olur. Bu sayede checkUpkeep teki gerekli likler tamamlanır ve  
        // performUpkeep çalışır duruma gelir. Kazanan hesaplandığı için kimse işlem yapamaz.  
        await network.provider.send("evm_increaseTime", [interval.toNumber() +  
1])  
        await network.provider.send("evm_mine", [])  
        // We pretend to be a Chainlink Keeper  
        await raffle.performUpkeep([])  
        await expect(raffle.enterRaffle({ value: raffleEntranceFee  
})).to.be.revertedWith(  
            "Raffle__NotOpen"  
        )  
    })
```

```
eemcs@DESKTOP-LJJ06I:~/freecodecamp/hardhat-smartcontract-lottery-cc$ hh test --grep "doesn't allow entrance when raffle id calculating"
```

Raffle Unit Tests

enterRaffle

✓ doesn't allow entrance when raffle id calculating

1 passing (2s)

```
eemcs@DESKTOP-LJJ06I:~/freecodecamp/hardhat-smartcontract-lottery-cc$
```

HH TEST

İLE TÜM TESLEERİ KONTROL ET.

```
eemcs@DESKTOP-LJJC06I:~/freecodecamp/hardhat-smartcontract-lottery-cc$ hh test
```

Raffle Unit Tests

constructor

- ✓ Initializes the raffle correctly

enterRaffle

- ✓ reverts when you don't pay enough

- ✓ records players when they enter

- ✓ emits event on enter

- ✓ doesn't allow entrance when raffle id calculating

5 passing (2s)

```
eemcs@DESKTOP-LJJC06I:~/freecodecamp/hardhat-smartcontract-lottery-cc$
```

Hh coverage ile test miktarını kontrol et

- ✓ Initializes the raffle correctly
- enterRaffle
  - ✓ reverts when you don't pay enough (47ms)
  - ✓ records players when they enter (49ms)
  - ✓ emits event on enter (71ms)
  - ✓ doesn't allow entrance when raffle id calculating (126ms)

5 passing (1s)

File	% Stmts	% Branch	% Funcs	% Lines	Uncovered Lines
contracts/	65	62.5	57.14	63.64	
Raffle.sol	65	62.5	57.14	63.64	... 179,183,187
contracts/test/	100	100	100	100	
VRFCoordinatorV2Mock.sol	100	100	100	100	
All files	65	62.5	57.14	63.64	

## CHECKUPKEEP TESTİ:

```
describe("checkUpkeep", async function () {
  it("returns false if people haven't sent any ETH", async function () {
    await network.provider.send("evm_increaseTime", [interval.toNumber() + 1])
    await network.provider.send("evm_mine", [])

    // Burada checkupkeep fonk dan upkeepNeeded değerini almamız gerekir. Fakat bu fonk view fonk
    // değil. Bunu yapabilmek için callStatic fonk kullanmamız gerekir. Callstatic transaction
    // simülasyonu yaparak geri dönen değeri almaktadır.
    const { upkeepNeeded } = await raffle.callStatic.checkUpkeep([])
    assert(!upkeepNeeded)
  })
})
```

```
eemcs@DESKTOP-LJJ06I:~/freecodecamp/hardhat-smartcontract-lottery-cc$ hh test --grep "returns false if people haven't sent any ETH"
```

Raffle Unit Tests

checkUpkeep

✓ returns false if people haven't sent any ETH

1 passing (2s)

```
it("returns false if raffle isn't open", async function () {
  await raffle.enterRaffle({ value: raffleEntranceFee })
  await network.provider.send("evm_increaseTime", [interval.toNumber() + 1])
  await network.provider.send("evm_mine", [])
  await raffle.performUpkeep([]) // boş byte nesnesi göndermek için ("0x")
  // yapısında kullanılabilir.
  const raffleState = await raffle.getRaffleState()
  const { upkeepNeeded } = await raffle.callStatic.checkUpkeep([])
  assert.equal(affleState.toString(), "1")
  assert.equal(upkeepNeeded, false)
})
```

```
eemcs@DESKTOP-LJJC06I:~/freecodecamp/hardhat-smartcontract-lottery-cc$ hh test --grep "returns false if raffle isn't open"
```

Raffle Unit Tests

checkUpkeep

✓ returns false if raffle isn't open

1 passing (2s)

SONRAKİ TESTLER GİTHUB REPOSUNDAN alındı test edild. Çalışıyorlar

```
it("returns false if enough time hasn't passed", async () => {
  await raffle.enterRaffle({ value: raffleEntranceFee })
  await network.provider.send("evm_increaseTime", [interval.toNumber() - 1])
  await network.provider.request({ method: "evm_mine", params: [] })
  const { upkeepNeeded } = await raffle.callStatic.checkUpkeep("0x")
  assert(!upkeepNeeded)
})
it("returns true if enough time has passed, has players, eth, and is open", async
() => {
  await raffle.enterRaffle({ value: raffleEntranceFee })
  await network.provider.send("evm_increaseTime", [interval.toNumber() + 1])
  await network.provider.request({ method: "evm_mine", params: [] })
  const { upkeepNeeded } = await raffle.callStatic.checkUpkeep("0x")
  assert(upkeepNeeded)
})
```



## Performupkeep() testi

```
describe("performUpkeep", function () {
  it("can only run if checkupkeep is true", async () => {
    await raffle.enterRaffle({ value: raffleEntranceFee })
    await network.provider.send("evm_increaseTime", [interval.toNumber() + 1])
    await network.provider.request({ method: "evm_mine", params: [] })
    const tx = await raffle.performUpkeep([])
    assert(tx)
  })
  it("reverts if checkup is false", async () => {
    await expect(affle.performUpkeep("0x")).to.be.revertedWith(
      "Raffle__UpkeepNotNeeded"
    )
  })
})
```

```
it("updates the raffle state, emits and event, and calls the vrf coordinator", async () => {  
    await raffle.enterRaffle({ value: raffleEntranceFee })  
    await network.provider.send("evm_increaseTime", [interval.toNumber() + 1])  
    await network.provider.request({ method: "evm_mine", params: [] })  
    const txResponse = await raffle.performUpkeep([])  
    const txReceipt = await txResponse.wait(1)  
    const raffleState = await raffle.getRaffleState()  
    const requestId = txReceipt.events[1].args.requestId  
    assert(requestId.toNumber() > 0)  
    assert(raffleState.toString() == "1")  
})  
})
```

```
describe("fulfillRandomWords", function () {
  beforeEach(async () => {
    await raffle.enterRaffle({ value: raffleEntranceFee })
    await network.provider.send("evm_increaseTime", [interval.toNumber() + 1])
    await network.provider.request({ method: "evm_mine", params: [] })
  })
  it("can only be called after performupkeep", async () => {
    await expect(
      vrfcoordinatorV2Mock.fulfillRandomWords(0, raffle.address)
    ).to.be.revertedWith("nonexistent request")
    await expect(
      vrfcoordinatorV2Mock.fulfillRandomWords(1, raffle.address)
    ).to.be.revertedWith("nonexistent request")
  })
})
```

MASSIVE PROMISE TEST 15.52.10

Yerel ağda test yapmak için yani çelişe katılan ve kazanan olması için yerel ağdan gelen sahte hesapları kullanacağız. Bunun için üç hesap alacağız. Hesaplarda 0. index deployer olacağı için katılımcılar 1. indexten başlayacak.

```
it("picks a winner, resets the lottery, and sends money", async () => {
  const additionalEntrances = 3
  const startingAccountIndex = 1 // deployer = 0
  const accounts = await ethers.getSigners()
  for (
    let i = startingAccountIndex;
    i < startingAccountIndex + additionalEntrances;
    i++
  ) {
    const accountConnectedRaffle = raffle.connect(accounts[i])
    await accountConnectedRaffle.enterRaffle({ value: raffleEntranceFee
  })

  const startingTimeStamp = await raffle.getLastTimeStamp()
  // performUpkeep ( mock being Chainlink Keepers)
  // fulfillRandomWords ( mock being the Chainlink VRf )
})
```

Burada fulfillRandowWords içindeki bütün değişkenleri kontrol etmemiz gerekir. fulfillRandomWords te kazanan hediyesi verilince bütün değerler resetlenmektedir. Bu işlemi test kısmında daha özel bir işlem ile yapacağız. Normalde fulfillRandowmWords un çağrılması için beklememiz gerekir. Burada simule edeceğiz.

Simule etmek için listener kullanacağız. Bunu da promise ile yapacağız.

```
// We will have to wait for the fulfillRandowmWords to be called
// But for local chain test we will simulate with promise
// Aşağıdaki yapı fulfillRandomWords fonk da WinnerPicked olayını yakalamak için dinler (
raffle.once("WinnerPicked"). Ve yakalandığında işlemler {} kısmına yazılacaktır.
    await new Promise(async (resolve, reject) => {
        raffle.once("WinnerPicked", () => {})
    })
})
```

```
    await new Promise(async (resolve, reject) => {
      raffle.once("WinnerPicked", () => {
        resolve()
      })
      // Setting up the listener

      // Below, we will fire the event, and the listner will pick it up, and
resolve
    })
```

Süreyi hızlandırmak için hardhat config de aşağıdaki ayarı yapacağız. 2 saniyede event ın tetiklenmesi gerekir. Tetiklenmez ise test başarısız olacaktır. Bu da bizim beklediğimiz sonuçtur. Hata oluşacağı için try-catch kullanacağız.

```
mocha: {
  timeout: 200000, // 200 seconds max
},
```

```
await new Promise(async (resolve, reject) => {
    raffle.once("WinnerPicked", () => {
        // 2 saniyede gerçekleşmezse reject yani geri çevrilecek.
        try {
        } catch (e) {
            reject(e)
        }
        // 2 saniyede tetiklenirse test edilecek
        resolve()
    })
    // Setting up the listener

    // Below, we will fire the event, and the listener will pick it up,
and resolve
})
```



```

await new Promise(async (resolve, reject) => {
  raffle.once("WinnerPicked", async () => {
    // 2 saniyede gerçekleşmezse reject yani geri çevrilecek.
    try {
    } catch (e) {
      reject(e)
    }
    // 2 saniyede tetiklenirse test edilecek
    resolve()
  })

```

```

// Setting up the listener

```

```

// Below, we will fire the event, and the listener will pick it up, and

```

resolve

```

// Ve bu fonksiyonun
(import edilen mock
contractındaki )
simulasyonu
yapılarak
contracttaki fonk.
içindeki değişkenler
kontrol edilecektir.
( sıfırlanacaklar )

```

```

// Mocking Chainlink Keepers an Chainlink VRF

```

```

const tx = await raffle.performUpkeep([])

```

```

const txReceipt = await tx.wait(1) // 1 block oluşana kadar bekle

```

```

// fullfillRandomWords V2Mock contractından çağrıldığını unutmayalım.

```

```

await vrfcoordinatorV2Mock.fullfillRandomWords(

```

```

// Buradaki fullfillRandomWords çağrıldığında "WinnerPicked" olayı

```

```

// yayınlanacaktır. Ve raffle.once("WinnerPicked") listener 1 olayı

```

```

// yakalayıp içindeki kodları çalıştıracaktır.

```

```

txReceipt.events[1].args.requestId,

```

```

raffle.address

```

```

)

```

```

})

```

WinnerPicked yakalanınca  
fulfillRandomWords() -  
raffle.sol contract içindeki  
fonksiyonda bulunan -  
içindeki değişkenleri kontrol  
edebiliriz.

```
console.log("Found the event!")
// 2 saniyede gerçekleşmezse reject yani geri çevrilecek.
try {
    const recentWinner = await raffle.getRecentWinner()
    console.log(recentWinner)
    console.log(accounts[2].address)
    console.log(accounts[0].address)
    console.log(accounts[1].address)
    console.log(accounts[3].address)
    const raffleState = await raffle.getRaffleState()
    const endingTimeStamp = await raffle.getLastTimeStamp()
    const numPlayers = await raffle.getNumberOfPlayers()
    // Herşeyin sıfırlandığını kontrol edeceğiz. Çünkü kazanan

    // ve yeni çekiliş süreci başlıyor
    assert.equal(numPlayers.toString(),"0")
    assert.equal(raffleState.toString(),"0")
    assert(endingTimeStamp > startingTimeStamp)
} catch (e) {
    reject(e)
}
// 2 saniyede tetiklenirse test edilecek
resolve()
})
```

belirlendi

Let's fix my spelling errors & Run Tests

16.02.31

```
eemcs@DESKTOP-LJJC06I:~/freecodecamp/hardhat-smartcontract-lottery-cc$ hh test --grep "picks a winner, resets the lottery, and sends money"
```

```
Raffle Unit Tests
```

```
  fulfillRandomWords
```

```
Found the event!
```

```
0x70997970C51812dc3A010C7d01b50e0d17dc79C8
```

```
0x3C44CdDdB6a900fa2b585dd299e03d12FA4293BC
```

```
0xf39Fd6e51aad88F6F4ce6aB8827279cFfFb92266
```

```
0x70997970C51812dc3A010C7d01b50e0d17dc79C8
```

```
0x90F79bf6EB2c4f870365E785982E1f101E93b906
```

```
✓ picks a winner, resets the lottery, and sends money
```

```
1 passing (6s)
```

Kazanan hesap 1. index nolu hesaptır. Bu hesabın başlangıç ve kazandıktan sonraki balance değerlerine bakalım.

```
const txReceipt = await tx.wait(1) // 1 block oluşana kadar bekle
const winnerStartingBalance = await accounts[1].getBalance()
```

```
const numPlayers = await raffle.getNumberOfPlayers()
const winnerEndingBalance = await accounts[1].getBalance()
```

```
// Kazanana ödeme yapıp yapılmadığından emin olmak için
assert.equal(
  winnerEndingBalance.toString(),
  winnerStartingBalance.add(
    raffleEntranceFee
      .mul(additionalEntrances)
      .add(affleEntranceFee)
      .toString()
  )
)
```

```
eemcs@DESKTOP-LJJC06I:~/freecodecamp/hardhat-smartcontract-lottery-cc$ hh test --grep "picks a winner, resets the lottery, and sends money"
```

Raffle Unit Tests

fulfillRandomWords

Found the event!

0x70997970C51812dc3A010C7d01b50e0d17dc79C8

0x3C44CdDdB6a900fa2b585dd299e03d12FA4293BC

0xf39Fd6e51aad88F6F4ce6aB8827279cFfFb92266

0x70997970C51812dc3A010C7d01b50e0d17dc79C8

0x90F79bf6EB2c4f870365E785982E1f101E93b906

✓ picks a winner, resets the lottery, and sends money

1 passing (6s)

```
eemcs@DESKTOP-LJJC06I:~/freecodecamp/hardhat-smartcontract-lottery-cc$
```

Testnet ortamında transaction işleminin ne zaman biteceğini bilemeyiz. Yerel ağda mock işlemi ile bunu gerçekleştirmiş olduk. Testnet te test yaparken tx, txReceipt ve diğer işlemleri kullanamayız.

Staging testte geçecek süreyi bilmediğimiz için hardhat config teki timeout u da ona göre denememiz gerekecek. Yani arttırmamız gerekecek.

## TÜM TESTLERİN ÇALIŞTIRILMASI

```
eemcs@DESKTOP-LJJC06I:~/freecodecamp/hardhat-smartcontract-lottery-cc$ hh test

Raffle Unit Tests
  constructor
    ✓ Initializes the raffle correctly
  enterRaffle
    ✓ reverts when you don't pay enough
    ✓ records players when they enter
    ✓ emits event on enter
    ✓ doesn't allow entrance when raffle id calculating
  checkUpkeep
    ✓ returns false if people haven't sent any ETH
    ✓ returns false if raffle isn't open
    ✓ returns false if enough time hasn't passed
    ✓ returns true if enough time has passed, has players, eth, and is open
  performUpkeep
    ✓ can only run if checkupkeep is true
    ✓ reverts if checkup is false
    ✓ updates the raffle state, emits and event, and calls the vrf coordinator
  fulfillRandomWords
    ✓ can only be called after performupkeep
Found the event!
0x70997970C51812dc3A010C7d01b50e0d17dc79C8
0x3C44CdDdB6a900fa2b585dd299e03d12FA4293BC
0xf39Fd6e51aad88F6F4ce6aB8827279cFfFb92266
0x70997970C51812dc3A010C7d01b50e0d17dc79C8
0x90F79bF6EB2c4f870365E785982E1f101E93b906
    ✓ picks a winner, resets the lottery, and sends money

14 passing (7s)

eemcs@DESKTOP-LJJC06I:~/freecodecamp/hardhat-smartcontract-lottery-cc$
```

RAFFLE.SOL STAGING TESTS

16.07.44



test/staging/Raffle.staging.test.js                      unit test ile benzer olduğu için unit test teki başlangıç kodlarını, constructor a kadar kopyala-yapıştır. Ve aşağıdaki düzenlemeleri yap.

- Mock a ihtiyaç olmadığı ve gerçek test nette test edileceği için vrfcoordinatorV2Mock a gere yoktur.
- interval a gerek yoktur.
- zaten deploy edildiği için fixture a gerek yoktur.

```
const { assert, expect } = require("chai")
const { network, getNamedAccounts, deployments, ethers } = require("hardhat")
const { developmentChains, networkConfig } = require("../..../helper-hardhat-config")

developmentChains.includes(network.name)
  ? describe.skip
  : describe("Raffle Unit Tests", async function () {
      let raffle, raffleEntranceFee, deployer
      const chainId = network.config.chainId

      beforeEach(async function () {
        deployer = (await getNamedAccounts()).deployer
        raffle = await ethers.getContract("Raffle", deployer)
        raffleEntranceFee = await raffle.getEntranceFee()
      })
    })
```

```
describe("fulfillRandomWords", function () {  
  it("works with live Chainlink Keepers and Chainlink VRF, we get a random  
winner", async function () {  
    // enter the raffle  
    const startingTimeStamp = await raffle.getLatestTimeStamp()  
    // setup listener before we enter the raffle  
    // just in case the blockchain moves REALLY fast  
    // await raffle.enterRaffle({value: raffleEntranceFee})  
  })  
})
```

```
const startingTimeStamp = await raffle.getLatestTimeStamp()
// setup listener before we enter the raffle
await new Promise(async(resolve, reject)=>{
  raffle.once("WinnerPicked", async () =>{
    try{
      //add our asserts here
    } catch(error){
      console.log(error)
      reject(error)
    }
  })
})
})
```

```
    await new Promise(async (resolve, reject) => {
      raffle.once("WinnerPicked", async () => {
        try {
          //add our asserts here
        } catch (error) {
          console.log(error)
          reject(error)
        }
      })
      // Then entering the raffle
      await raffle.enterRaffle({ value: raffleEntranceFee })
      // and this code WONT complete until our listener has finished
      listening!
    })
```

```
raffle.once("WinnerPicked", async () => {
  try {
    const recentWinner = await raffle.getRecentWinner()
    const raffleState = await raffle.getRaffleState()
    const winnerEndingBalance = await accounts[0].getBalance()
    const endingTimeStamp = await raffle.getLatestTimeStamp()
    await expect(affle.getPlayer(0)).to.be.reverted
    assert.equal(recentWinner.toString(), accounts[0].address)
    assert.equal(raffleState, 0)
    assert.equal(
      winnerEndingBalance.toString(),
      winnerStartingBalance.add(raffleEntranceFee).toString()
    )
    assert(endingTimeStamp > startingTimeStamp)
    resolve()
  } catch (error) {
    console.log(error)
    reject(error)
  }
})
// Then entering the raffle
await raffle.enterRaffle({ value: raffleEntranceFee })
const winnerStartingBalance = await accounts[0].getBalance()
```

TESTING ON A TESTNET 16.18.19

## GEREKLİ İŞLEMLER

1- Chainlink VRF için SubscriptionID al

<https://vrf.chain.link/>

Öncesinde eğer yok ise, metamaskta rinkeby testnet hesabını ayarla ve <https://faucets.chain.link> adresinden test ETH ve LINK al

Metamaskta LINK token gözükmüyorsa, <https://docs.chain.link/docs/link-token-contacts/> adresinden, Rinkeby kısmından token adresini alarak tokene içe aktar.

2- SubscriptionID kullanarak contract ı deploy et

3- Chainlink VRF ve SubscriptionId ile contractı kayıt et

<https://vrf.chain.link/rinkeby/<subscription id >> adresinden Raffle.sol contract adresi consumer ekle

4- Chainlink Keepers a contract ı kayıt et

<https://keepers.chain.link/rinkeby> adresinden Raffle.sol contract adresi ile kayıt et

5- Staging testi çalıştır.

[Home](#) /

# Create Subscription

Create a Subscription ID using your wallet address. The Subscription ID will be used in your contract when requesting a random value. [Learn more](#)

## Please connect your wallet

To create a Subscription please connect to your wallet first.

Connect wallet



Need help or have questions? [Talk to an expert](#) or visit the [VRF webpage](#) to learn more





Home /

# Create Subscription

Create a Subscription ID using your private key and your contract when requesting a subscription.

Subscription address

0xd1f68a3b433f02c3640bb327

As an admin, you may cancel or edit a subscription.

[Create subscription](#)

1

[Approve subscription creation](#)

## Approve subscription creation

Please confirm the transaction.

[Close](#)

3

[Add consumers.](#)

After you create the subscription,  
Add as many subscriptions as you want.

MetaMask Notification

Rinkeby Test Ağı

Encode Solidi... → 0x616...57ab

Yeni adres algılandı! Adres defterinize eklemek için buraya tıklayın.

New gas experience

We've updated how gas fee estimation and customization works.

[Turn on Enhanced Gas Fee UI in Settings](#)

https://vrf.chain.link

0x616...57ab : CREATE SUBSCRIPTION ⓘ

\$0.00

AYRINTILAR

VERİ

ON ALTILI

Tahmini gaz ücreti

DOZENLE

\$0.10 0.00009 ETH ⓘ

Home /

# Create Subscription

Create a Subscription ID using your contract when requesting a subscription.

Subscription address

0xd1f68a3b433f02c3640bb327

As an admin, you may cancel or edit a subscription.

Create subscription



Approve subscription creation

Receive confirmation

## Subscription created

Congratulations, subscription was created.

View your transaction here:

[0xb6dd92a79643dd72205ff4380725533719952f2753ceb912204f8969fa40fc77](#)

Add funds

Add as many subscriptions as you like.

[Home](#) / [My Subscriptions](#) / [ID 6978](#) /

## Add Funds

Add funds to your subscription. Your subscription is only billed after your contract receives a random value and you will never be billed more than the maximum price you specify. You can withdraw your funds at anytime. [Learn more](#)

 Need LINK for testing? Visit the [Chainlink faucet](#) to receive testnet LINK.

Add funds (LINK)

Your wallet balance: 30.0 LINK

Add funds

I'll do it later

## How to Create a Subscription?

Easy as 1, 2, 3

- ✓ Create a subscription with your wallet address.
- 2 Add funds to your subscription. You can always do it later.
- 3 Add consumers.

Subscription Id yi helper-hardhat-config.js dosyasına ekle

```
Raffle.sol 2 x JS Raffle.test.js JS Raffle.staging.test.js JS helper-hardhat-config.js JS hardhat.config.js
1  const { ethers } = require("hardhat")
2
3  const networkConfig = {
4    4: {
5      name: "rinkeby",
6      vrfCoordinatorV2: "0x6168499c0cFfCaCD319c818142124B7A15E857ab",
7      entranceFee: "1000000000000000000",
8      gasLane: "0xd89b2bf150e3b9e13446986e571fb9cab24b13cea0a43ea20a6049a85cc807cc",
9      subscriptionId: "6978", // will change
10     callbackGasLimit: "500000",
11     interval: "30",
12   },
}
```

[Home](#) / [My Subscriptions](#) / [ID 6978](#) /

## Add Funds

Add funds to your subscription. Your subscription is only billed after your contract receives a random value and you will never be billed more than the maximum price you specify. You can withdraw your funds at anytime. [Learn more](#)

**i** Need LINK for testing? Visit the [Chainlink faucet](#) to receive testnet LINK.

Add funds (LINK)

Your wallet balance: 30.0 LINK

Add funds

I'll do it later

Mainnet ile çalışırken ne kadar link gerektiğine <https://docs.chain.link/docs/vrf-contracts> bağlantısından kontrol edebilirsiniz.

BU İŞLEM DE METAMASK HESABINDAN YAPILACAK YANİ METAMASKTANDA ONAYLANACAKTIR.


## How to Create a Subscription?

Easy as 1, 2, 3

- ✓ Create a subscription with your wallet address.
- 2 Add funds to your subscription. You can always do it later.
- 3 Add consumers.

## My Subscriptions

Active




ID	Created	Consumers	Balance
 6978	June 22, 2022 at 11:11 UTC	0	5 LINK
 6373	June 12, 2022 at 16:52 UTC	1	9.695783638361336 LINK
 6372	June 12, 2022 at 16:42 UTC	0	0 LINK

Prev

Showing 1 to 3 of 3 entries

Next

## Recent subscriptions

ID	Admin address	Created	Consumers	Balance
 6980	 0xcb37...4308 	June 22, 2022 at 11:14 UTC	0	0.5 LINK

## 2 - DEPLOY CONTRACT

.env dosyasını kontrol et

```
Raffle.sol 2 JS Raffle.test.js JS Raffle.staging.test.js JS helper-hardhat-config.js .env X
1 RINKEBY_RPC_URL=https://eth-rinkeby.alchemyapi.io/v2/i0fx4ZF4IN_vobajVDYv1SC-
2 PRIVATE_KEY=d64eecf0eb8a06d1b43a2aa0e9dff0052807d707dd310d429a4b7dc93c9c0bd6
3 ETHERSCAN_API_KEY=QJZRWKXAW2HN7644S3AG43GE6TIK4GFRQ5
4 COINMARKETCAP_API_KEY=f85e9459-346e-4257-97fe-1a2039eecd70
```

**Before deploying, be sure to check the github repo for the optimal hardhat-config. So that all module.exports are present.**

DOĞRULAMA İÇİN ETHERSCAN AYARINI HARDHAT.CONFIG DOSYASINA EKLE.

```
mocha: {  
  timeout: 200000, // 200 seconds max  
},  
etherscan: {  
  apiKey: ETHERSCAN_API_KEY,  
},
```





```
eemcs@DESKTOP-LJJC06I:~/freecodecamp/hardhat-smartcontract-lottery-cc$ yarn hardhat deploy --network rinkeby
yarn run v1.22.15
warning package.json: No license field
$ /home/eemcs/freecodecamp/hardhat-smartcontract-lottery-cc/node_modules/.bin/hardhat deploy --network rinkeby
Nothing to compile
reusing "Raffle" at 0xc0a2C3BEFdc64adF652EA54482A612db205E3373
Verifying....
Verifying contract...
Nothing to compile
Warning: Unnamed return variable can remain unassigned. Add an explicit return with value to all non-reverting code paths or name the variable.
--> contracts/Raffle.sol:107:13:
|
107 |         bytes memory /* performData */
|         ^^^^^^^^^^^^^
Warning: Function state mutability can be restricted to view
--> contracts/Raffle.sol:100:5:
|
100 |     function checkUpkeep(
|     ^ (Relevant source part starts here and spans across multiple lines).

Successfully submitted source code for contract
contracts/Raffle.sol:Raffle at 0xc0a2C3BEFdc64adF652EA54482A612db205E3373
for verification on the block explorer. Waiting for verification result...

Successfully verified contract Raffle on Etherscan.
https://rinkeby.etherscan.io/address/0xc0a2C3BEFdc64adF652EA54482A612db205E3373#code

Done in 30.73s.
eemcs@DESKTOP-LJJC06I:~/freecodecamp/hardhat-smartcontract-lottery-cc$
```

BAĞLANTIDAN CONTRACT I  
KONTROL EDEBİLİRSİNİZ.

Status	ID ⓘ	Admin ⓘ	Consumers	Fulfillments ⓘ	Balance ⓘ
Active	6978	 0xd1f6...8e4f 	0	0	5 LINK
<a href="#">Cancel Subscription</a> After canceling, the funds will be returned to the specified address					


### 3 - CHAINLIN vrf KAYDI:

Raffle.sol contract adresini kopyala ve CONSUMER olarak ekle.

İşlemi METAMASK TAN ONAYLA. Bu işlem contractın bu site ile etkileşim kurmasını sağlayacaktır.

### No consumers

Your subscription is ready. You can now add consumers.

 Important: Your consumer contract must use the Subscription ID **6978** in the VRF request to make use of these funds.

Consumer address ⓘ

0xc0a2C3BEFdC64adF652EA54482A612db205E3373

Add consumer

Cancel



NEW

Chainlink Keepers is live on Polygon and BSC mainnet. [Get started today](#)

# Chainlink Keepers

Automate your smart contract with Chainlink's hyper-reliable network of keepers.

Register new Upkeep

Go to the docs



Need help creating  
your first upkeep?

Automate your smart contract  
in 5 minutes

## Quick links

[What is Chainlink Keepers?](#)

[Custom logic automation](#)

**i** Need LINK for testing? Visit the [Chainlink faucet](#) to receive testnet LINK.

Email address

celalaksu@gmail.com

Enter your email address so we can send important notifications about your Upkeep.

Upkeep address

0xc0a2C3BEFdC64adF652EA54482A612db205E3373

✔ This is a keeper compatible contract.

Gas limit

500000

Amount of gas to provide the target contract when performing Upkeep. This will impact minimum balance requirements, and should be approximately the maximum amount of gas the transaction might use.

Starting balance (LINK)

8

Deposit LINK to your Upkeep. Select an amount that will satisfy multiple performances to start, then fund the Upkeep directly once it's operational.

Register Upkeep

Cancel

Upkeep name

Raffle Upkeep

Select a unique name for your Upkeep.

Admin address

0xD1F68A3b453102c3b40Bb3271fa9132b73A68E4f

Address to cancel Upkeep and withdraw remaining funds.

Check data (Hexadecimal) *Optional*

```
function performUpkeep(  
    bytes calldata /* performData */  
){  
    // performUpkeep function  
}
```

foksiyonunda veri olmadığı için boş bırakıldı

Metamask tan onayla

 Need LINK for testing? V

Email address

celalaksu@gmail.com

Enter your email address so we ca

Upkeep address

0xc0a2C3BEFdc64adF652EAS

 This is a keeper compatible co

Gas limit

500000

Amount of gas to provide the targ  
minimum balance requirements, a  
the transaction might use.

Starting balance (LINK)

8



Submit registration request

Receive confirmation

## Upkeep registration request submitted successfully

You can view your Upkeep registration via button below. If it's pending approval, you will receive an email once it's approved.

View your transaction here:

[0x6a13bb888318a706ca1c0d1749d6748b29b7d43fd2f4e08cd21175dd9eb09f38](#)

Close

View Upkeep

be converted to bytes. See [docs](#)

**NEW** Chainlink Keepers is live on Polygon and BSC mainnet. [Get started today](#)

# Chainlink Keepers

Automate your smart contract with Chainlink's hyper-reliable network of keepers.

[Register new Upkeep](#)

[Go to the docs](#)

[How it works](#)

Registry address

 0x409CF388DaB66275dA3e44005D182c12EeAa12A0 

[All upkeeps](#)

[My upkeeps](#)

Name

Address

Balance

[Raffle Upkeep](#)



0xc0a2c3befdc64adf652ea54482a612db205e3373



8 LINK

Home /









# Raffle Upkeep

DETAYLAR

Actions ▾

Status	Registry address	Balance	Minimum Balance ⓘ
● Active	 0x409C...12A0 	8 LINK	N/A

## Details ^

Registration 	Upkeep 	Trigger 
<div>Owner address  0xD1F6...8E4F </div> <div>Date June 22, 2022 at 11:51 UTC</div> <div>Transaction Hash 0x6a13...9f38 </div>	<div>ID 1437</div> <div>Address  0xc0a2...3373 </div> <div>Gas limit 500,000</div>	<div>Type Custom</div> <div>Check data (Base16) <div>0x</div></div>

## 5 - RUN STAGING TEST

scripts/enter.js oluştur.

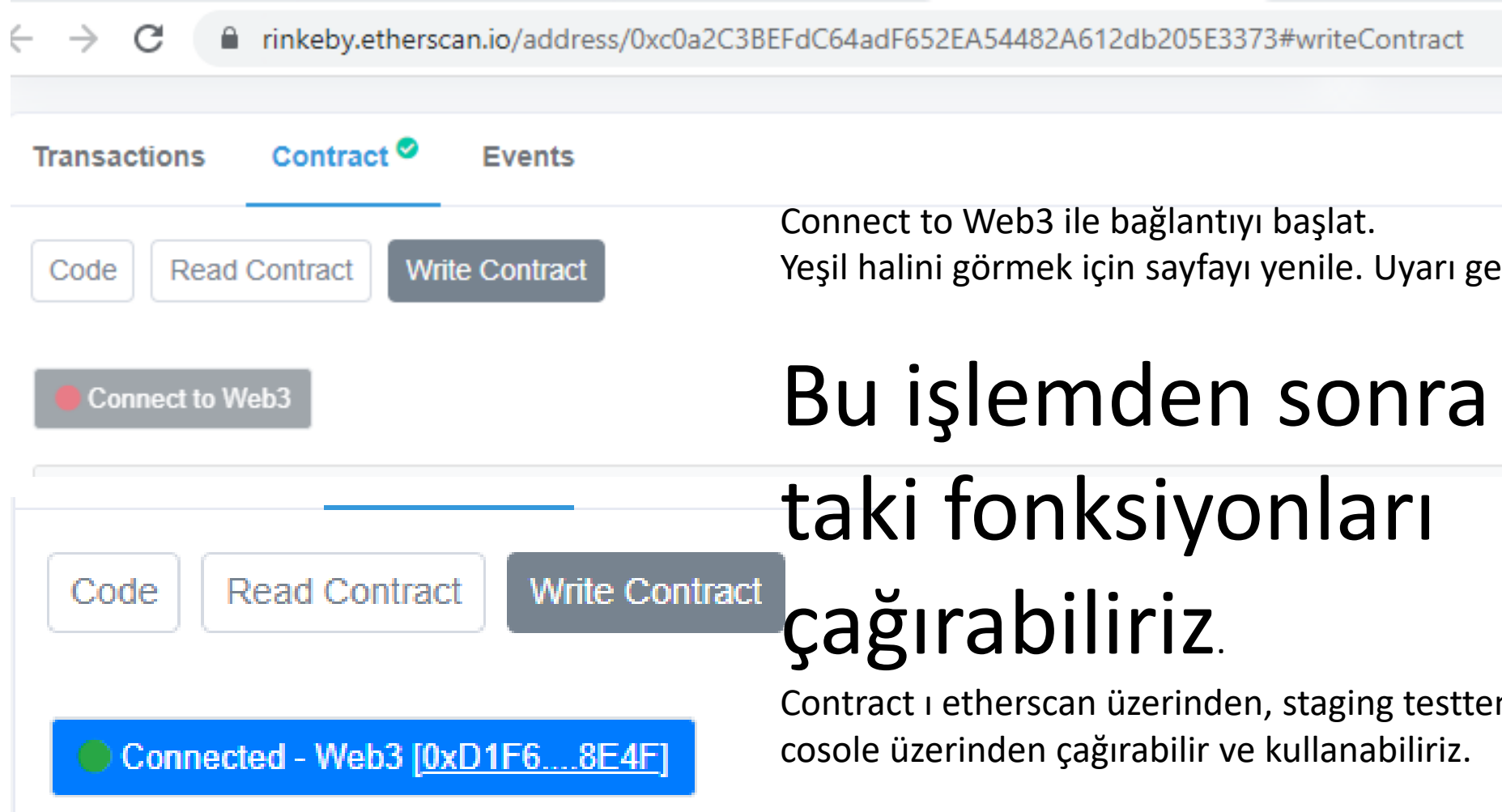
```
const { ethers } = require("hardhat")

async function enterRaffle() {
  const raffle = await ethers.getContract("Raffle")
  const entranceFee = await raffle.getEntranceFee()
  await raffle.enterRaffle({ value: entranceFee + 1 })
  tx = await raffle.enterRaffle({ value: entranceFee + 1 })
  await tx.wait(1)
  console.log(tx.hash)
  console.log("Entered!")
}

enterRaffle()
  .then(() => process.exit(0))
  .catch((error) => {
    console.error(error)
    process.exit(1)
  })
```



Test işleminden önce, contract doğrulandıktan sonra, contractı metamask cüzdanına bağlamamız gerekir. Contract adresinden [rinkeby.etherscan.io/address/<contract adresi>](https://rinkeby.etherscan.io/address/0xc0a2C3BEFdC64adF652EA54482A612db205E3373#writeContract) CONTRACT kısmında WRITE CONTRACT butonu tıklanarak metamask cüzdanı bağlantısı yapılır.



Connect to Web3 ile bağlantıyı başlat.  
Yeşil halini görmek için sayfayı yenile. Uyarı gelir, onayla.

# Bu işlemten sonra contract taki fonksiyonları çağırabiliriz.

Contract ı etherscan üzerinden, staging testten, scripts lerden,  
cosole üzerinden çağırabilir ve kullanabiliriz.

Staging testi yaptıktan sonra;  
keepers chain de  
vrf chainde  
Transaction görmemiz gerekir.

hh test --network rinkeby

TEST GERÇEKLEŞİRKEN;  
rinkeby.etherscan.io dan gerçekleşen transaction ları sayfayı yenileyerek görebiliriz.

```
eemcs@DESKTOP-LJJ06I:~/freecodecamp/hardhat-smartcontract-lottery-cc$ hh test --network rinkeby

Raffle Unit Tests
  fulfillRandomWords
    ✓ works with live Chainlink Keepers and Chainlink VRF, we get a random winner (69332042 gas)

1 passing (16m)

eemcs@DESKTOP-LJJ06I:~/freecodecamp/hardhat-smartcontract-lottery-cc$
```

Contract 0xc0a2C3BEFdC64adF652EA54482A612db205E3373

Contract Overview

Balance: 0 Ether

More Info

My Name Tag: Not Available

Contract Creator: 0xd1f68a3b433f02c3640... at txn 0xe37f73a91804ff7facfc4...

TransactionsInternal TxnsContractEvents

Latest 2 from a total of 2 transactions

Txn Hash	Method	Block	Age	From	To	Value	Txn Fee
0x89837ac033e16f9bc6...	Enter Raffle	10896510	3 mins ago	0xd1f68a3b433f02c3640...	IN 0xc0a2c3befdc64adf652...	0.1 Ether	0.000155369741
0xe37f73a91804ff7facfc4...	0x61016060	10896304	55 mins ago	0xd1f68a3b433f02c3640...	IN Create: Raffle	0 Ether	0.003343211255

[ Download CSV Export ]

```
.encryptedKey.json
# vscode
.vscode
```

```
# hardhat
artifacts
cache
deployments
node_modules
coverage
coverage.json
typechain
```

```
# don't push the environment vars!
.env
```

```
# Built application files
.DS*
*.apk
*.ap_
*.aab
```

```
# Files for the ART/Dalvik VM
*.dex
```

```
# Java class files
*.class
```

```
# Generated files
bin/
gen/
out/
# Uncomment the following line in case you need and you don't have the release build type files in your app
# release/
```

```
# Gradle files
.gradle/
build/
```

```
# Local configuration file (sdk path, etc)
local.properties
```

```
# Proguard folder generated by Eclipse
proguard/
```

```
# Log Files
*.log
```

Github a yüklemek için .gitignore dosyası oluşturun.

```
# Android Studio Navigation editor temp files
.navigation/
```

```
# Android Studio captures folder
captures/
```

```
# IntelliJ
*.iml
.idea/workspace.xml
.idea/tasks.xml
.idea/gradle.xml
.idea/assetWizardSettings.xml
.idea/dictionaries
.idea/libraries
# Android Studio 3 in .gitignore file.
.idea/caches
.idea/modules.xml
# Comment next line if keeping position of elements in Navigation Editor is relevant for you
.idea/navEditor.xml
```

```
# Keystore files
# Uncomment the following lines if you do not want to check your keystore files in.
#*.jks
#*.keystore
```

```
# External native build folder generated in Android Studio 2.2 and later
.externalNativeBuild
```

```
# Google Services (e.g. APIs or Firebase)
google-services.json
```

```
# Freeline
freeline.py
freeline/
freeline_project_description.json
```

```
# fastlane
fastlane/report.xml
fastlane/Preview.html
fastlane/screenshots
fastlane/test_output
fastlane/readme.md
```

```
# Version control
vcs.xml
```

```
# lint
lint/intermediates/
lint/generated/
lint/outputs/
lint/tmp/
# lint/reports/
```

```
gas-report.txt
```

# Completed Hardhat Basics!





soooooooooooooon



git init --b main ( bu bazen çalışmayabilir. Vs code da git bölümünden yapılabilir. Sonrasında git status çalışıyor.

git status

git add .

git commit -m 'intial commit'

"github ta repo oluştur ve adresini aşağıya ekle"

git remote add origin <https://github.com/celalaksu/hardhat-fund-me-cc.git>

git remote -v

git branch -M main

git push -u origin main

( vs code giriş izni isteyecektir. Yada önceden github kullanıcı adı ve şifre tanımlanmış olmalıdır.

TYPESCRIPT

16.32.39