

# 한 번에 끝내는 블록체인, dApp 개발의 모든 것

---

## Chapter 4

### Lottery 컨트랙트 v2 개발

Chapter 4

Lottery 컨트랙트 v2 개발

# Chainlink VRF Consumer 컨트랙트 파악하기

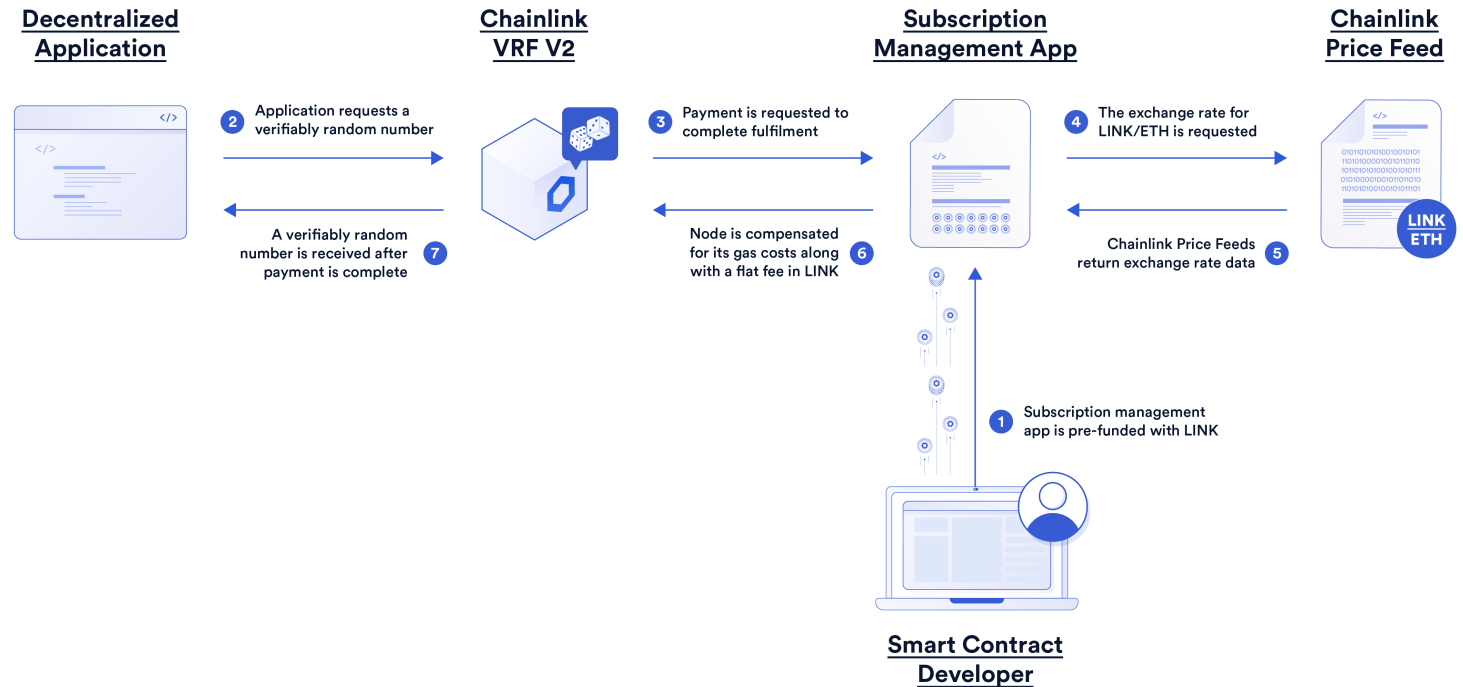


# Chainlink VRF 구조 파악하기

VRF 컨트랙트 동작 구조

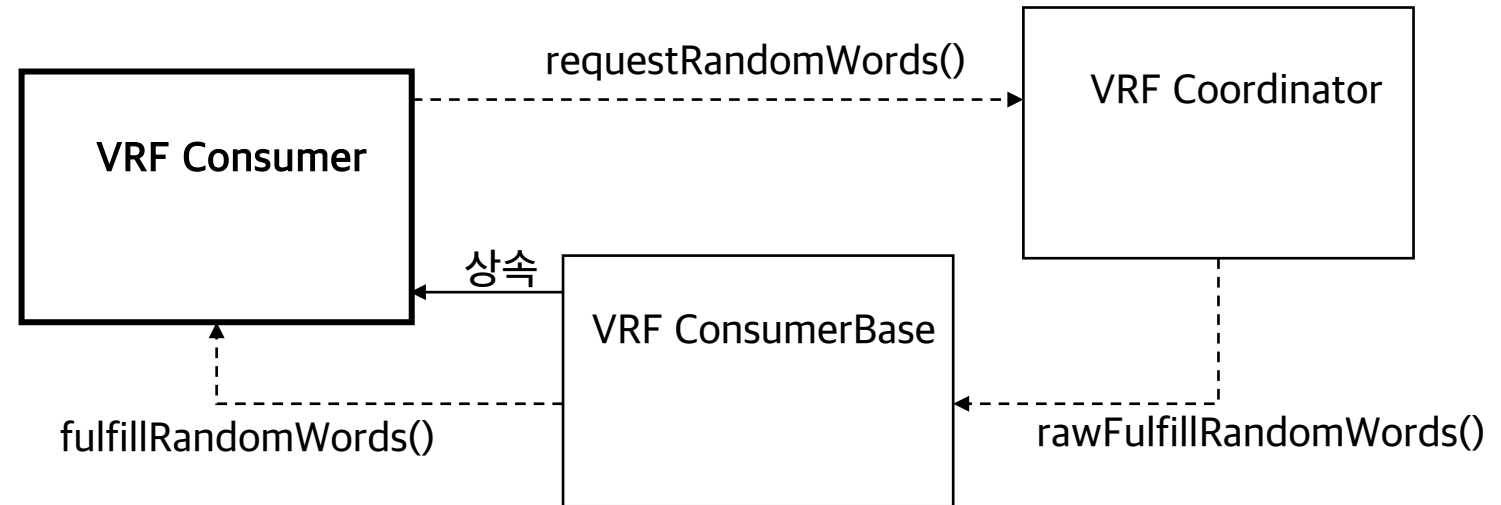
# Chainlink VRF (v2) 동작 방식

1. 스마트 컨트랙트 개발자가 Chainlink의 subscription management app에 LINK 토큰 예치 및 랜덤 값 요청할 컨트랙트 등록
2. 컨트랙트에서 Chainlink VRF로 랜덤 값 요청
3. Chainlink VRF는 해당 컨트랙트에 대해 Subscription management app에 예치되어있는 LINK 토큰을 사용하여 랜덤 값 생성 및 검증
4. 검증된 랜덤 값을 컨트랙트로 전송



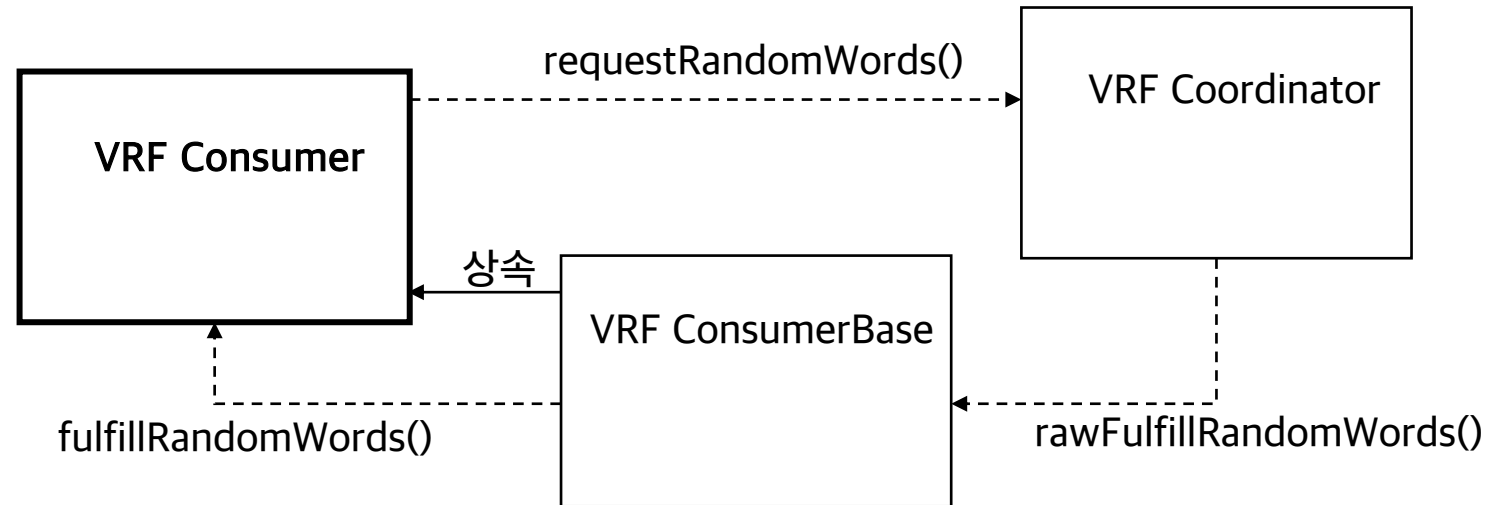
# Chainlink VRF 컨트랙트 동작 구조

- VRF Consumer: Chainlink VRF에 랜덤 값을 요청하는 컨트랙트 (즉, 우리가 구현할 컨트랙트)
- VRF Coordinator: consumer 컨트랙트로부터 랜덤 값 생성을 요청 받으면 랜덤 값을 생성하여 반환해주는 컨트랙트



# Chainlink VRF 컨트랙트 동작 구조

- requestRandomWords(): VRF Consumer 컨트랙트에서 VRF Coordinator 컨트랙트로 랜덤 값 생성 요청
- rawFulfillRandomWords(): VRF Coordinator가 랜덤 값 생성 후 VRF Consumer에서 설정한 block confirmation 기간이 지난 후에 callback으로 VRF Consumer에 랜덤 값 반환
- VRF ConsumerBase의 rawFulfillRandomWords()에서 internal 함수인 fulfillRandomWords() 호출 → 랜덤 값 받은 후의 작업 수행







# Chainlink VRF 이용 준비 하기

테스트넷 ETH 및 LINK 토큰 얻기 &  
Subscription Management App에 등록하기

# 테스트 ETH 및 LINK 토큰 얻기

- 이더리움 Goerli 테스트넷 이용
- <https://faucets.chain.link/>
- Goerli ETH 및 LINK 토큰 얻기

## Request testnet LINK

Get testnet LINK for an account on one of the supported blockchain testnets so you can create and test your own oracle and Chainlinked smart contract. [Learn more](#)



Your wallet is connected to Ethereum Görli, so you are requesting Ethereum Görli LINK/ETH.

Wallet address

0x65115eb5e9acfd27213db1874531f27f4186add3

Request type



20 test LINK



0.1 test ETH



# Subscription Management App - Create Subscription

- <https://vrf.chain.link/goerli>
- Chainlink VRF Subscription Management App

## Chainlink Verifiable Randomness Function

Chainlink VRF provides cryptographically secure randomness for your smart contracts.

Create Subscription

Go to the docs


LINK Token ⓘ

0x326c...06fb 

VRF Coordinator ⓘ

0x2ca8...734d 

Key hash ⓘ

0x79d3...0c15 

Max gas price ⓘ

30 Gwei

# Subscription Management App - Create Subscription

## 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](#)

Subscription address

0x65115eb5e9acfd27213db1874531f27f4186add3

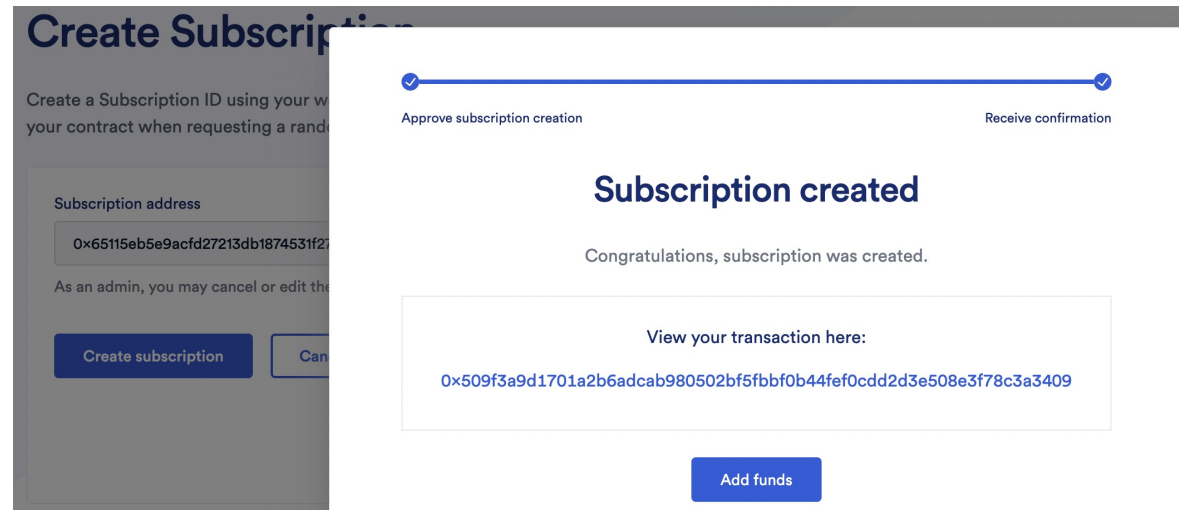
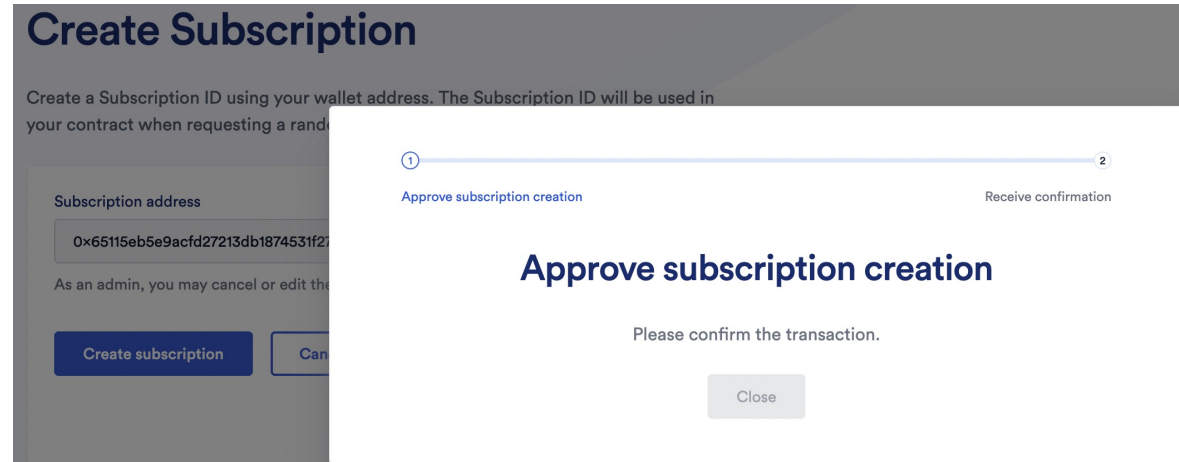
As an admin, you may cancel or edit the subscription anytime

Create subscription

Cancel

# Subscription Management App - Create Subscription

- Create Subscription 후 Add funds 클릭



# Subscription Management App - Add Funds

- 미래에 랜덤 값 생성 요청하는데 사용할 LINK 토큰을 미리 충분히 예치

## 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

# Subscription Management App - Subscription

- Home 화면에서 생성된 subscription과 함께 해당 subscription에 예치해둔 LINK 토큰양 확인 가능

## My Subscriptions

Active

ID	Created	Consumers	Balance
 1816	September 19, 2022 at 18:59 UTC	0	2 LINK



# Chainlink VRF Consumer Example 컨트랙트 파악하기

Chainlink의 VRF Consumer Example 컨트  
랙트 파악하기



# Chainlink VRF Consumer Example

- <https://remix.ethereum.org/#url=https://docs.chain.link/samples/VRF/VRfV2Consumer.sol&optimize=false&runs=200&evmVersion=null&version=soljson-v0.8.7+commit.e28d00a7.js>

```
1 // SPDX-License-Identifier: MIT
2 // An example of a consumer contract that relies on a subscription for funding.
3 pragma solidity ^0.8.7;
4
5 import "@chainlink/contracts/src/v0.8/interfaces/VRFCoordinatorV2Interface.sol";
6 import "@chainlink/contracts/src/v0.8/VRFCConsumerBaseV2.sol";
7
8 /**
9  * THIS IS AN EXAMPLE CONTRACT THAT USES HARDCODED VALUES FOR CLARITY.
10  * THIS IS AN EXAMPLE CONTRACT THAT USES UN-AUDITED CODE.
11  * DO NOT USE THIS CODE IN PRODUCTION.
12  */
13
14 contract VRfV2Consumer is VRFCConsumerBaseV2 {
15     VRFCoordinatorV2Interface COORDINATOR;
16
17     // Your subscription ID.
18     uint64 s_subscriptionId;
19
20     // Goerli coordinator. For other networks,
21     // see https://docs.chain.link/docs/vrf-contracts/#configurations
22     address vrfCoordinator = 0x2Ca8E0C643bDe4C2E08ab1fA0da3401AdAD7734D;
23
24     // The gas lane to use, which specifies the maximum gas price to bump to.
25     // For a list of available gas lanes on each network,
26     // see https://docs.chain.link/docs/vrf-contracts/#configurations
27     bytes32 keyHash = 0x79d3d8832d904592c0bf9818b621522c988bb8b0c05cdc3b15aea1b6e8db0c15;
28
29     // Depends on the number of requested values that you want sent to the
30     // fulfillRandomWords() function. Storing each word costs about 20,000 gas,
31     // so 100,000 is a safe default for this example contract. Test and adjust
32     // this limit based on the network that you select, the size of the request,
33     // and the processing of the callback request in the fulfillRandomWords()
34     // function.
35     uint32 callbackGasLimit = 100000;
36
37     // The default is 3, but you can set this higher.
38     uint16 requestConfirmations = 3;
```

# Chainlink VRF Consumer Example

- vrfCoordinator: 랜덤 값 요청할 VRF Coordinator 컨트랙트 주소
- keyHash: VRF Coordinator에 요청할 때 허용할 최대 가스비를 나타내는 식별자
  - <https://docs.chain.link/docs/vrf/v2/supported-networks/#configurations>
  - 블록체인 네트워크 별로, 허용할 최대 가스비 별로 상이한 식별자 제공
- callbackGasLimit: VRF Coordinator로부터 콜백을 통해 랜덤 값 수신시, 처리할 복잡한 작업 수행에 허용할 최대 가스 리밋
- requestConfirmations: VRF Coordinator가 랜덤 값 응답까지 기다릴 block confirmation 기간

```
1 // SPDX-License-Identifier: MIT
2 // An example of a consumer contract that relies on a subscription for funding.
3 pragma solidity ^0.8.7;
4
5 import "@chainlink/contracts/src/v0.8/interfaces/VRFCoordinatorV2Interface.sol";
6 import "@chainlink/contracts/src/v0.8/VRFConsumerBaseV2.sol";
7
8 /**
9  * THIS IS AN EXAMPLE CONTRACT THAT USES HARDCODED VALUES FOR CLARITY.
10  * THIS IS AN EXAMPLE CONTRACT THAT USES UN-AUDITED CODE.
11  * DO NOT USE THIS CODE IN PRODUCTION.
12  */
13
14 contract VRFv2Consumer is VRFConsumerBaseV2 {
15     VRFCoordinatorV2Interface COORDINATOR;
16
17     // Your subscription ID.
18     uint64 s_subscriptionId;
19
20     // Goerli coordinator. For other networks,
21     // see https://docs.chain.link/docs/vrf-contracts/#configurations
22     address vrfCoordinator = 0x2Ca8E0C643bDe4C2E08ab1fA0da3401AdAD7734D;
23
24     // The gas lane to use, which specifies the maximum gas price to bump to.
25     // For a list of available gas lanes on each network,
26     // see https://docs.chain.link/docs/vrf-contracts/#configurations
27     bytes32 keyHash = 0x79d3d8832d904592c0bf9818b621522c988bb8b0c05cdc3b15aea1b6e8db0c15;
28
29     // Depends on the number of requested values that you want sent to the
30     // fulfillRandomWords() function. Storing each word costs about 20,000 gas,
31     // so 100,000 is a safe default for this example contract. Test and adjust
32     // this limit based on the network that you select, the size of the request,
33     // and the processing of the callback request in the fulfillRandomWords()
34     // function.
35     uint32 callbackGasLimit = 100000;
36
37     // The default is 3, but you can set this higher.
38     uint16 requestConfirmations = 3;
```

# Chainlink VRF Consumer Example

- numWords: 한 번의 요청에서 받고자 하는 랜덤 값 개수
- requestRandomWords(): VRF Coordinator 에 랜덤 값 요청하는 함수
- fulfillRandomWords(): VRF Coordinator가 랜덤 값을 콜백으로 응답할 때 호출하는 함수

```
40 // For this example, retrieve 2 random values in one request.
41 // Cannot exceed VRFCoordinatorV2.MAX_NUM_WORDS.
42 uint32 numWords = 2;
43
44 uint256[] public s_randomWords;
45 uint256 public s_requestId;
46 address s_owner;
47
48 constructor(uint64 subscriptionId) VRFConsumerBaseV2(vrfCoordinator) {
49     COORDINATOR = VRFCoordinatorV2Interface(vrfCoordinator);
50     s_owner = msg.sender;
51     s_subscriptionId = subscriptionId;
52 }
53
54 // Assumes the subscription is funded sufficiently.
55 function requestRandomWords() external onlyOwner {  infinite gas
56     // Will revert if subscription is not set and funded.
57     s_requestId = COORDINATOR.requestRandomWords(
58         keyHash,
59         s_subscriptionId,
60         requestConfirmations,
61         callbackGasLimit,
62         numWords
63     );
64 }
65
66 function fulfillRandomWords()  undefined gas
67     uint256, /* requestId */
68     uint256[] memory randomWords
69 ) internal override {
70     s_randomWords = randomWords;
71 }
72
73 modifier onlyOwner() {
74     require(msg.sender == s_owner);
75     _;
76 }
77 }
```



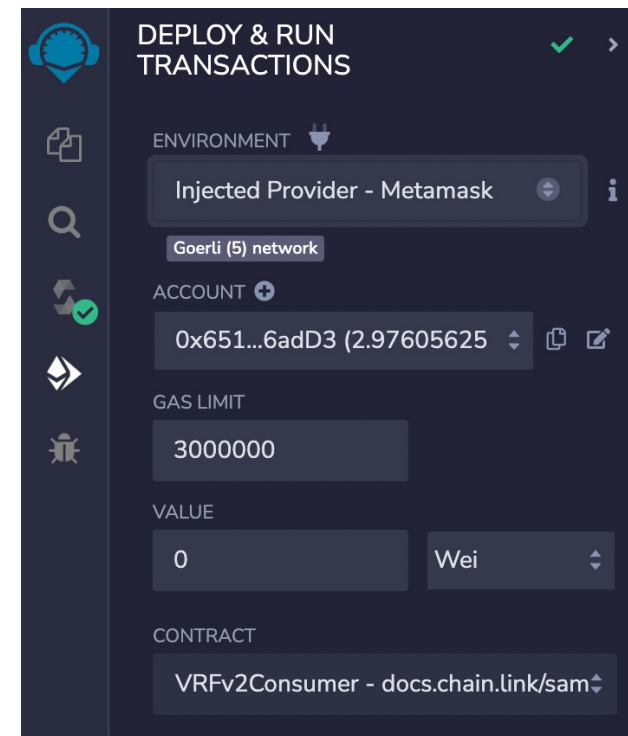
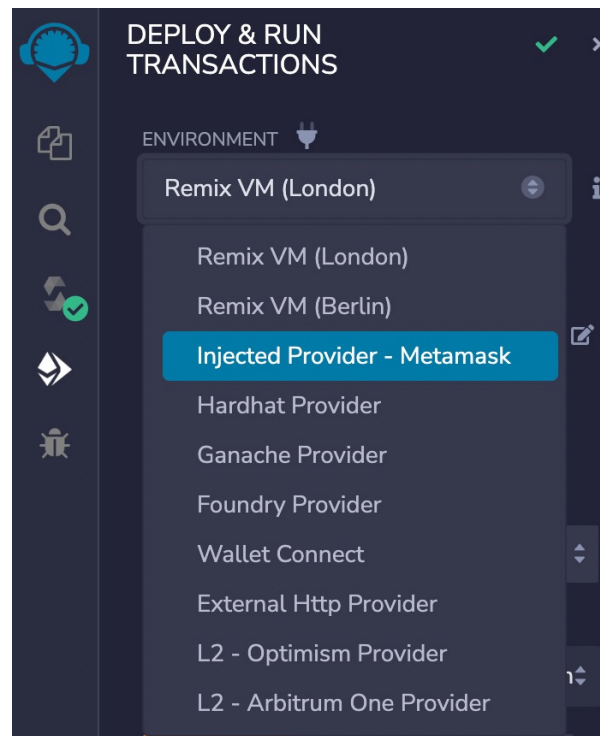


# Chainlink VRF Consumer Example 컨트랙트 테스트하기

VRF Consumer 컨트랙트 배포하고  
Subscription Management App에 등록 후  
이용하기

# VRF Consumer 컨트랙트 배포하기

- Remix가 연결할 블록체인 네트워크를  
Injected Provider - Metamask로 변경하기
- Goerli 테스트넷으로 연결됨



# VRF Consumer 컨트랙트 배포하기

- Deploy할 때 인자로, subscription management app에서 생성한 subscription id 입력 후 배포
- 컨트랙트 주소 복사

CONTRACT

VRFv2Consumer - docs.chain.link/sam↕

Deploy

1816

▼

☐ Publish to IPFS

OR

At Address

Load contract from Address

Transactions recorded **1** ⓘ >

Deployed Contracts ⓘ

> VRFV2CONSUMER AT 0X1DE...F2FB: ⓘ ✕





# Subscription Management App - Add Consumer

- Home 화면에서 subscription id 클릭

Home / Ethereum Görli /

## Subscription

Add Funds

Status	ID ⓘ	Admin ⓘ	Consumers	Fulfillments ⓘ	Balance ⓘ
● Active	1816	 0x6511...add3 	0	0	2 LINK

[Cancel Subscription](#) After canceling, the funds will be returned to the specified address

**No consumers**

Your subscription is ready. You can now add consumers.


Add consumer

# Subscription Management App - Add Consumer

- Consumer address에 배포한 VRF Consumer 컨트랙트 주소 입력 후 Add Consumer 클릭
- 이제 VRF Consumer 컨트랙트에서 랜덤 값 생성 요청 가능해진 상태

### No consumers

Your subscription is ready. You can now add consumers.

 **Important:** Your consumer contract must use the Subscription ID **1816** in the VRF request to make use of these funds.

Consumer address ⓘ




0x1De1c13b11647239C88694A5CA915036936f2FB1

Add consumer

Cancel

### Consumers

Add consumer

Address	Added	Last fulfillment	Total spent LINK	Actions
 0x1de1...2fb1 	September 19, 2022 at 19:08 UTC	-	-	

# VRF Consumer 컨 트랙트 테스트하기

- requestRandomWords() 호출: VRF  
Coordinator 컨트랙트로 랜덤 값 생성 요청



# VRF Consumer 컨트랙트 테스트하기

- requestRandomWords() 호출 후,
- s\_requestId 값이 설정됨
- 컨트랙트에서 설정한 requestConfirmations 값이 3이었으므로, block confirmation 기간으로 3 블록 기다려야함
- s\_randomWords view 함수의 인자로 0 또는 1 넣고 호출 (numWords를 2로 지정했었음): block confirmation 기간이 지난 후 VRF Coordinator로부터 받은 랜덤 값이 저장되어 있음

