
# Smart Contract Developer Bootcamp

## Day 2: Hardhat Track


# Smart Contract Developer Bootcamp

**Day 2**

Hardhat Development Environment, and the Hardhat Starter Kit

Chainlink

# Harry Papacharissiou


Developer Advocate, Chainlink Labs

harry@chainlinklabs.com

@Pappas9999

harrypapacharissiou


Chainlink

# Smart Contract Developer Bootcamp Outline


### Day 1: Introduction

Learn about blockchain, smart contracts and solidity, followed by examples of how to use Chainlink Price Feeds, Any-API and VRF in your smart contracts. Working with Solidity

### Day 2: Development Environments

Learn how to use the Hardhat development environment, followed by a guide on using our starter kits to create smart contracts that use Chainlink. Refining your project with local deployments and testing

Chainlink

# Agenda: Day 2

1. Introduction to Development Environments
2. Hardhat Development Environment
3. Chainlinks Hardhat Starter Kit
4. Working with a Local Network
5. Smart Contract Security
6. Testing Smart Contracts
7. Where Can I Learn More

# Housekeeping Rules

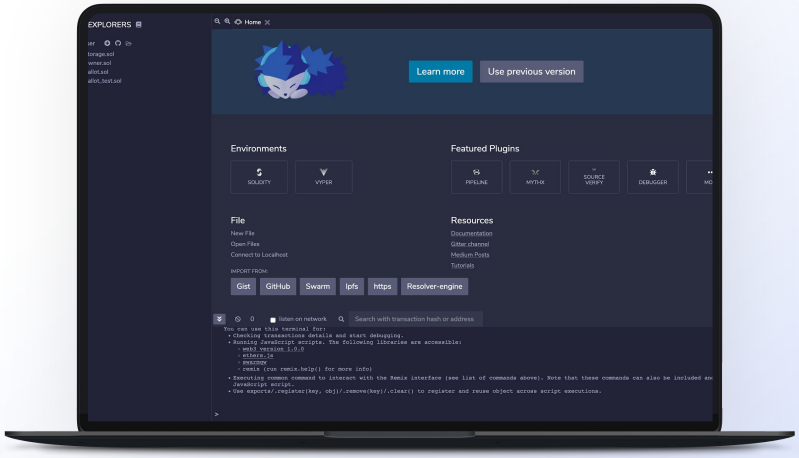
- 4 hour session, broken up into sections, with breaks
- Presentation, followed by questions, demo and exercise
- Questions can be asked at the end of each section
- Zoom: If you need help during a practical exercise, raise your hand and an instructor will eventually get to you and send you an invite to a private breakout session 
- YouTube: Technical questions can be asked in the chat or in the #developer-bootcamp channel in the Chainlink Discord

# Introduction to Development Environments

Chainlink

# Introduction to Development Environments

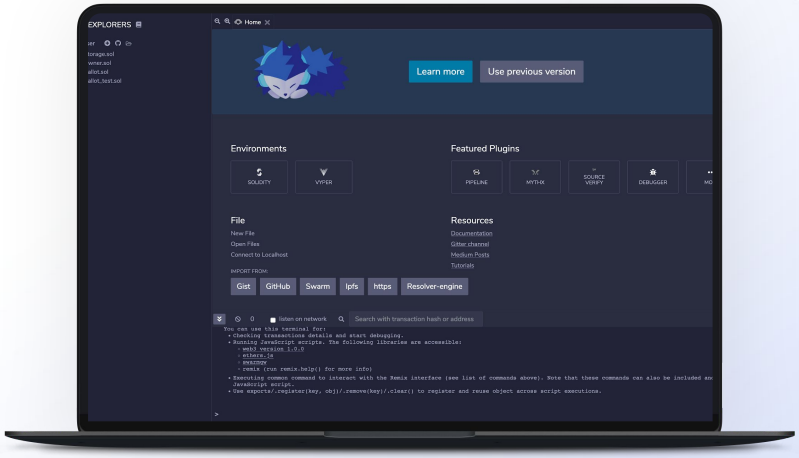
- Set of software tools and processes to develop source code
- Used to build applications
- Designed to maximize productivity and efficiency
- Can we local or web-based




Chainlink

# Remix IDE (Integrated Development Environment)

- Web-based (also a desktop version)
- Modules for testing, debugging and deploying smart contracts
- Local JavaScript VM to simulate Ethereum
- Extremely quick and easy to create and test smart contracts




Chainlink

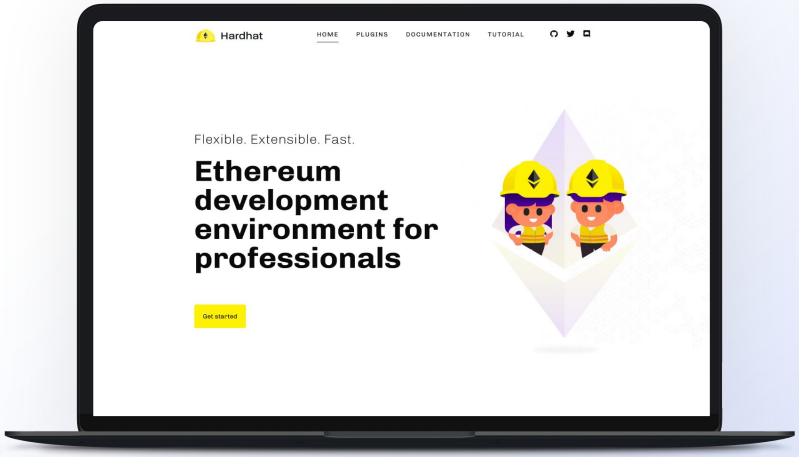
# Limitations of Remix

- Only deals with smart contracts
- Can’t integrate other parts of a project
- Limited support for tests or custom deployments
- Can’t save files locally without a plugin
- Any custom functionality requires a plugin

# Hardhat

# Hardhat Development Environment

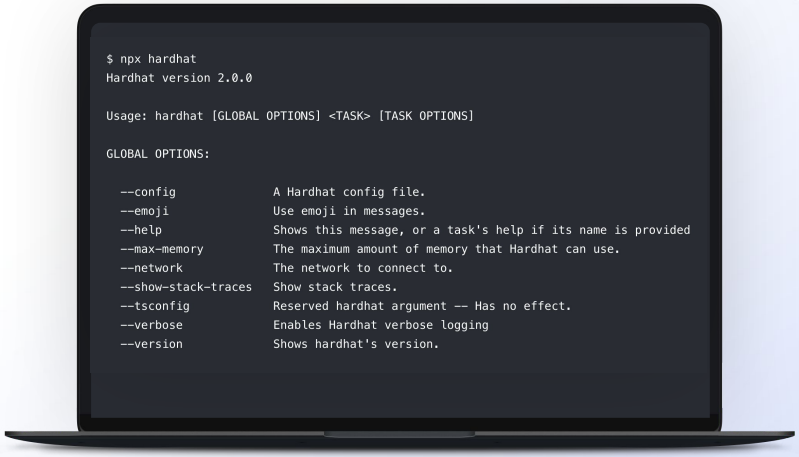
- Flexible JavaScript based Development Environment to compile, deploy, test and debug EVM based smart contracts
- Enables easy integration of code and external tools
- Local Hardhat Network to simulate Ethereum
- Extensible plugin features
- High level of debugging features




Chainlink

# Hardhat Tasks and Plugins

- JavaScript asynchronous function with associated metadata.
- Used to automate common tasks
- Everything you can do in Hardhat is defined as a task
- Can package them into plugins that can be imported by another project
- Hardhat has a large range of plugins listed, including support for web3.js and ethers.js




Chainlink

# Demo 1: Creating Your First Hardhat Project

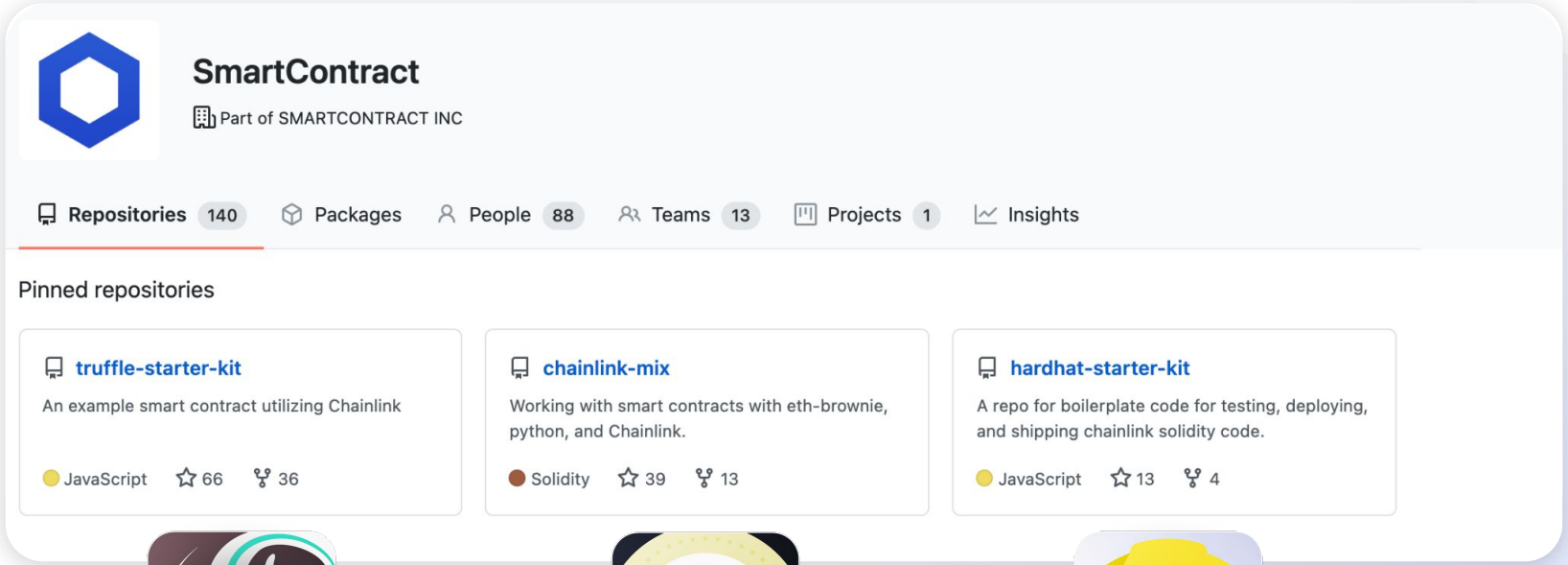
# Exercise 1: Creating Your First Hardhat Project

- Hardhat: https://hardhat.org/
- 45 mins + 5 mins break

# Hardhat Starter Kit

Chainlink

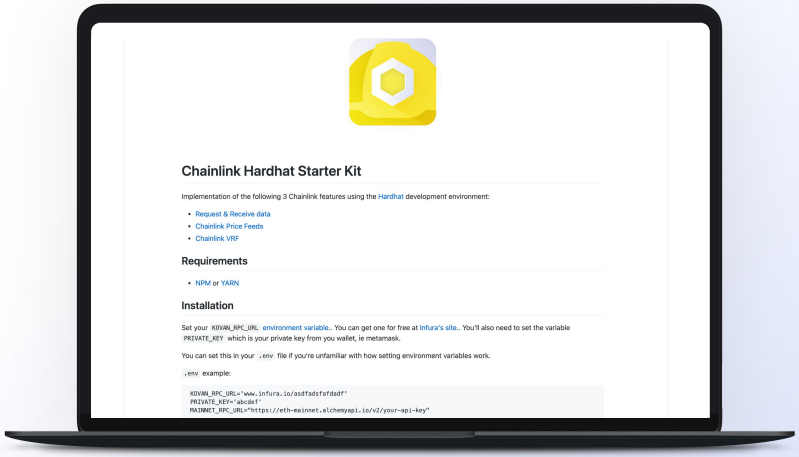
# Solidity and Smart Contract Starter Kits




Chainlink

# Hardhat Starter Kit

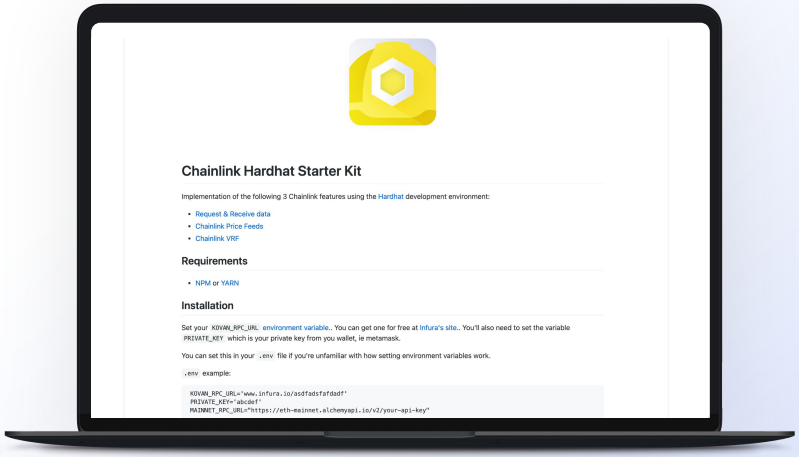
- https://github.com/smartcontractkit/hardhat-starter-kit
- Prepackaged Hardhat repository for building smart contracts that use Chainlink
  - Smart contracts
  - Deployment scripts
  - Tests
  - Tasks
  - Configuration




Chainlink

# Hardhat Starter Kit

- Allows quick building, deploying and testing of smart contracts that use Chainlink
- Can be integrated with other parts of a project
- Increases productivity via the use of custom Chainlink tasks




Chainlink

# Demo 2: Installing and Using the Hardhat Starter Kit

# Exercise 2: Installing and Using the Hardhat Starter Kit

- https://github.com/smartcontractkit/hardhat-starter-kit
- 45 minutes

# Working With a Local Network

# Hardhat Network

- Local Ethereum network
- Allows you to deploy, run and test your code
- Can be run in a standalone fashion, and connected to external clients

# Hardhat Network

- Mines a block with each transaction received, in order and with no delay
- Can be used to run tests
- Compatible with plugins
- Can be thought of as another network (like Kovan, Mainnet etc)

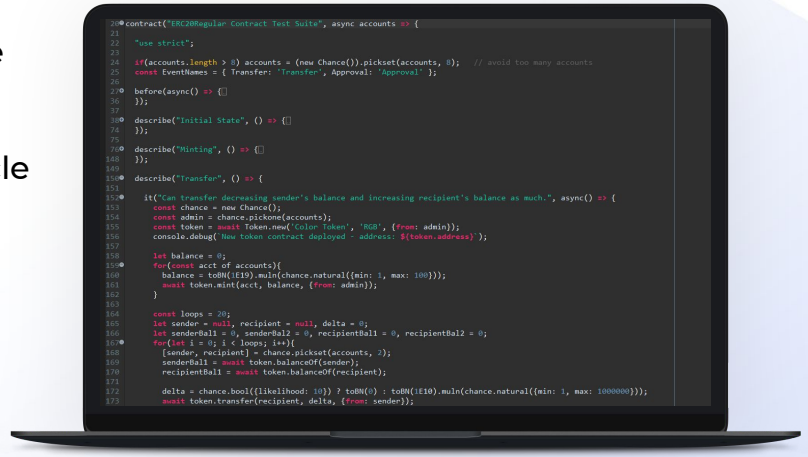
# Hardhat Network

- Can fork Mainnet!
- Initial state:
  - Genesis block
  - 20 accounts with 10000 ETH each


Chainlink

# Design Patterns for Local Deployments: Mocks

- Common design pattern involving creating a second version of something that imitates the original
- Useful for tests on local networks where state isn't the same as a public network, or where there is no integration to external services such as Chainlink oracle networks




Chainlink

# Demo 3: Deploying to a Local Network

# Exercise 3: Deploying to a Local Network

- Deploy our smart contracts to a local network
- Fork Mainnet
- 30 minutes + 5 minute break

# Smart Contract Security

Chainlink

> *Everyone here is a target for attack.*
>
> *Be paranoid.*

**- Mertin Swende**
Ethereum Foundation Security Lead

# Smart Contract Security

- Two main sources of security problems:
  - Bugs in the blockchain infrastructure
    - Protocols
    - EVM
  - Bugs or Vulnerabilities in the smart contract code
    - Developer mistakes or unanticipated code execution

Chainlink

# Smart Contract Security

- Programming smart contracts requires a different engineering mindset
- Cost of failure can be high
- Changes can be difficult
  - Similar to hardware development
- Multiple well known smart contract attacks:
  - Reentrancy attack
  - Cross-function race conditions
  - Integer overflow
  - DoS attack (revert, block gas limit)
  - Forcibly sending ether to a contract
  - TX reordering attacks
  - Timestamp dependence attack

Chainlink

# Testing Smart Contracts

# Testing Your Smart Contracts

- Executing code in order to identify any gaps, errors, or missing requirements
- Combination of subsequent, structured executions of a smart contract, validating the result each time
- Smart contracts are immutable
- Testing is even more crucial for smart contracts than for traditional applications.

Chainlink

# Testing Your Smart Contracts

- Rule of thumb: Every line of contract logic should have a corresponding test.
- The most important scenarios should be tested the most thoroughly.
- Tests can catch careless mistakes.
- Tests are great for edge cases.

Chainlink

# Bug vs Vulnerability

- A bug is when an issue leads to a planned scenario not executing, or executing incorrectly
- A vulnerability is when an issue leads to an unplanned scenario executing
- Tests do not prevent vulnerabilities.


Chainlink

# Types of Tests

- Unit tests
  - Simple tests that verify a single behavior or component within a smart contract
- Integration tests
  - More complex tests that validate interactions between multiple components, including off-chain components such as oracles

# Testing Approach

- Arrange-Act-Assert (AAA) Testing Pattern
  - Arrange the initial conditions required
  - Act, by calling the function to be tested
  - Assert the result of the action

# Writing Good Unit Tests


**Simple**

Each unit test should be evaluating a single behavior

Chainlink

# Writing Good Unit Tests


**Simple**

Each unit test should be evaluating a single behavior

**Repeatable**

Each unit test should always produce the same result

Chainlink

# Writing Good Unit Tests


**Simple**

Each unit test should be evaluating a single behavior

**Repeatable**

Each unit test should always produce the same result

**Isolated**

Each unit test should not depend upon or affect other tests

Chainlink

# Writing Good Unit Tests


**Simple**

Each unit test should be evaluating a single behavior

**Repeatable**

Each unit test should always produce the same result

**Isolated**

Each unit test should not depend upon or affect other tests

**Readable**

Someone else should easily be able to work out what's being tested

Chainlink

# Writing Good Unit Tests


**Simple**

Each unit test should be evaluating a single behavior

**Repeatable**

Each unit test should always produce the same result

**Isolated**

Each unit test should not depend upon or affect other tests

**Readable**

Someone else should easily be able to work out what's being tested

**Fast**

Unit tests should be written with speed in mind

Chainlink

# Integration Tests

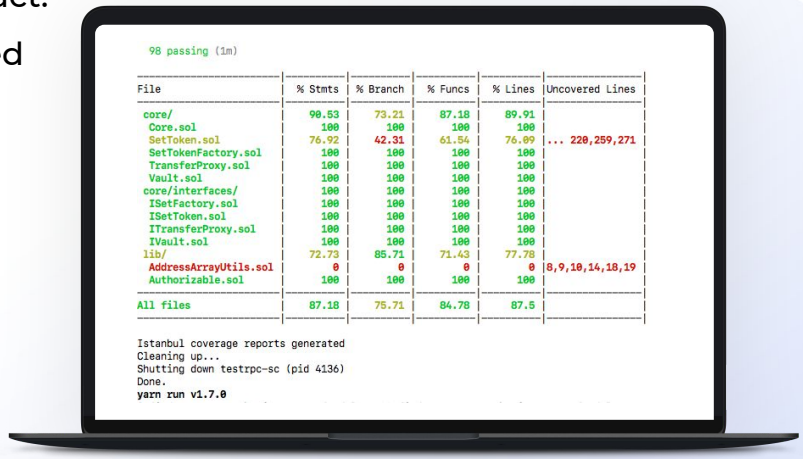
- Interactions between contracts
- Interaction between on-chain contracts and off-chain services such as Oracles
- JavaScript tests
  - Ensure contracts exhibit the right external behaviour

# Tools for Testing

- Waffle
  - Library for writing and testing smart contracts
- Mocha
  - JavaScript testing framework for browser/Node.js
- Chai
  - Assertion library for browser/NodeJS

# Tools for Testing: Testing Coverage

- https://github.com/sc-forks/solidity-coverage
- Tests should ideally ‘touch’ all of the code in a contract.
- Checks to see how much of the code has been tested




Chainlink

# Demo 4: Testing Smart Contracts

# Exercise 4: Testing Smart Contracts

- Unit Tests
- Integration Tests
- Solidity Coverage
- Publishing to GitHub
- 30 minutes

Chainlink

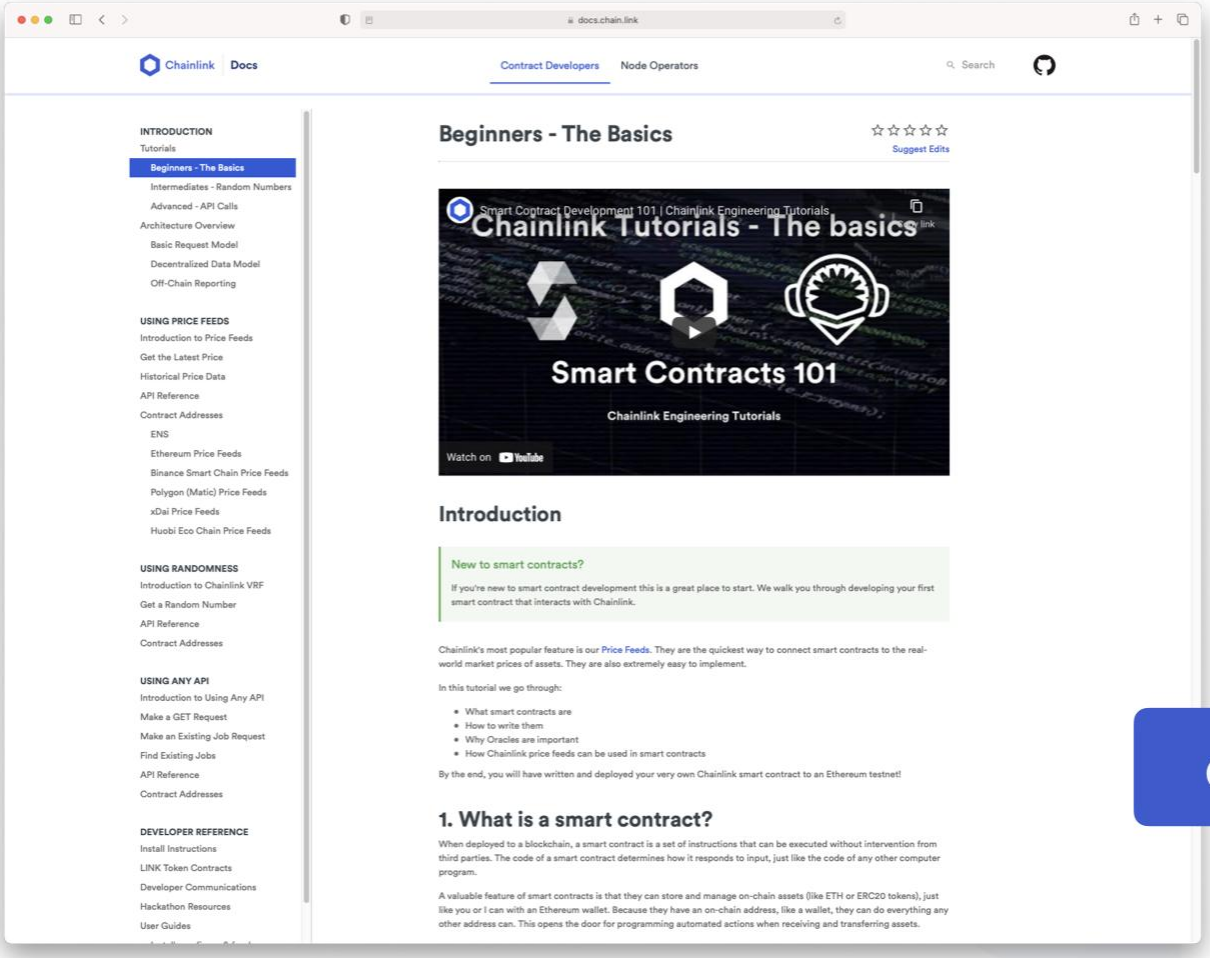
# Developer Bootcamp Day 2 Summary

- Development Environments
- Hardhat Development Environment
- Hardhat Starter Kit
- Local Development in Hardhat
- Smart Contract Security
- Unit and Integration Testing
- Publishing projects to GitHub

# Where Can I Learn More

# Solidity Resources

- CryptoZombies
- Ivan on Tech
- Dapp University
- Eat The Blocks
- Chainshot




LINK Token Contracts

Suggest Edits

LINK tokens are used to pay node operators for retrieving data for smart contracts and also for deposits placed by node operators as required by contract creators.

The LINK token is an ERC677 token that inherits functionality from the ERC20 token standard and allows token transfers to contain a data payload. Read more about the ERC677 transferAndCall token standard.

## Ethereum

### Mainnet

| Parameter | Value |
|---|---|
| ETH_CHAIN_ID | 1 |
| Address | 0x514910771af9ca656af840dff83e8264ecf986ca |
| Name | ChainLink Token |
| Symbol | LINK |
| Decimals | 18 |

### Kovan

Kovan Faucets

Testnet LINK is available from https://kovan.chain.link/
Testnet ETH is available from https://faucet.kovan.network/

| Parameter | Value |
|---|---|
| ETH_CHAIN_ID | 42 |
| Address | 0xa36085F69e2889c224210F603D836748e7dC0088 |
| Name | ChainLink Token |
| Symbol | LINK |
| Decimals | 18 |

### Rinkeby

Rinkeby Faucets

Testnet LINK is available from https://rinkeby.chain.link/
Testnet ETH is available from https://faucet.rinkeby.io/

**docs.chain.link**

### Goerli

| Parameter | Value |
|---|---|
| ETH_CHAIN_ID | 5 |
| Address | 0x326c977e6efc84e512bb9c30f76e30c160ed06fb |
| Name | ChainLink Token |
| Symbol | LINK |
| Decimals | 18 |

# blog.chain.link

blog.chain.link

Chainlink VISION ANNOUNCEMENTS EVENTS DEVELOPERS EDUCATION GRANTS CHINESE KOREAN

## Developers


DEVELOPERS

### Fetch IPFS Data in Smart Contracts Using a Chainlink External Adapter

Learn how to use a Chainlink External Adapter to fetch IPFS data and reduce Ethereum gas fees for large token distributions and airdrops on Audius and other platforms.


DEVELOPERS

### Off-Chain Computation: Statistical Analysis With Chainlink

Learn how to use Chainlink's External Adapters to enable off-chain computation of statistical weather analysis.


DEVELOPERS

### Chainlink Developers Starter Kit Showcase

Announcing the Starter Kit Showcase, a month-long initiative that encourages developers to use our Starter Kits to create smart contracts that make use of Chainlink tools.


DEVELOPERS

### How to Connect a Two-Factor Authentication (2FA) API to a Smart Contract With Chainlink

Chainlink hackathon winner Digital Bridge walks through how they built a two-factor authentication service to trigger on-chain events using Chainlink oracles.


DEVELOPERS

### Build Your Own Dynamic NFT With Hardhat

In this tutorial, learn how to use Chainlink VRF in Hardhat's popular development environment to build a next-generation NFT that changes every time it's transferred.

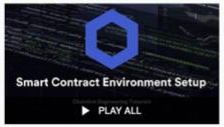

DEVELOPERS

### Craft Whiskey Crypto Payments With Chainlink Oracles

Learn how Chainlink bounty winners implemented a Chainlink Price Feed to provide whiskey buyers and sellers accurate and tamper-proof on-chain prices of craft whiskey bottles.

# Chainlink Youtube

## Chainlink Engineering Tutorials & Developer Workshops




Chainlink Engineering Tutorials

16 videos • 8,933 views • Last updated on May 4, 2021

Chainlink

1. Set Up Your Environment for Success | Chainlink Engineering Tutorials — Chainlink — 4:15
2. Smart Contract Development 101 | Chainlink Engineering Tutorials — Chainlink — 10:57
3. Getting A Random Number with Chainlink VRF | Chainlink Engineering Tutorials — Chainlink — 15:30
4. Chainlink API Call | Chainlink Engineering Tutorials — Chainlink — 11:17
5. What is Ethereum? | Chainlink Engineering Tutorials — Chainlink — 7:00
6. What is a Blockchain Oracle? What is the Oracle Problem? | Chainlink Engineering Tutorials — Chainlink — 5:47
7. Truffle, Hardhat, and Defi - Chainlink Hackathon Workshop — Chainlink — 47:34
8. Brownie, Python, and Defi - Chainlink Hackathon Workshop — Chainlink — 56:21
9. Testing Smart Contracts & Multi-Chains | Chainlink Hackathon Workshop — Chainlink — 1:12:55
10. Dynamic NFT Creation - Code-Along | Chainlink Hackathon Workshop — Chainlink — 1:30:49
11. Building and Using External Adapters | Chainlink Engineering Tutorials — Chainlink — 15:18


Chainlink Developer Workshops

43 videos • 2,383 views • Last updated on May 4, 2021

Chainlink

1. Chainlink API Call | Chainlink Engineering Tutorials — Chainlink — 11:17
2. Data Providers | Chainlink Hackathon Workshop — Chainlink — 24:49
3. Running a local Chainlink Node | Chainlink Hackathon Workshop — Chainlink — 1:05:01
4. Introduction to Smart Contracts, Blockchain, & Solidity - Chainlink Hackathon Workshop — Chainlink — 1:04:11
5. Introduction to Chainlink - Chainlink Hackathon Workshops — Chainlink — 1:38:56
6. Brownie, Python, and Defi - Chainlink Hackathon Workshop — Chainlink — 56:21
7. Truffle, Hardhat, and Defi - Chainlink Hackathon Workshop — Chainlink — 47:34
8. Ultimate Customization with External Adapters | Chainlink Hackathon Workshop — Chainlink — 57:57
9. Dynamic NFT Creation - Code-Along | Chainlink Hackathon Workshop — Chainlink — 1:30:49
10. Testing Smart Contracts & Multi-Chains | Chainlink Hackathon Workshop — Chainlink — 1:12:55
11. Getting A Random Number with Chainlink VRF | Chainlink Engineering Tutorials — Chainlink — 15:30

# Using Chainlink Nodes In Your Project

- Market.link: Check job status with Node Operators
- Market.link: Verification currently being re-worked
- Need a job or an adapter hosted? Post in **#operator-requests** channel in the Chainlink Discord
- Anything can be realized by building customized DoNs (decentralized oracle networks)

# Learn more!

**Discord Chat**
http://chn.lk/chainlink-discord

**Starter Kits**
https://github.com/smartcontractkit

**Documentation:**
http://docs.chain.link

**Hackathon Resources**
http://chn.lk/hack-resources

Chainlink

Chainlink

# Congratulations, you've completed the Summer 2021 Smart Contract Developer Bootcamp