**CODELANDCS BLOCKCHAIN DEVELOPMENT SYLLABUS**

**WEEK 6**

**DAY 3**

**STORAGE PROJECT USING HARDHAT FRAMEWORK**

**GETTING STARTED**

 On the terminal, initialize a new project with ***yarn init --y***or

package manager of choice.

 Next run ***yarn add --dev hardhat***

 Run ***yarn hardhat***to set up the development environment

and follow the instructions

Run ***yarn hardhat***to see lists of tasks that can be used in hardhat.

**COMPILING SMART CONTRACTS IN HARDHAT**

To compile a smart contract in hardhat, use the ***yarn hardhat***

***compile***task. This would generate two folders, namely:-

 Artifacts

 Cache

The Artifacts folder contains all the necessary information about

the compiled contract. (run ***yarn hardhat clean***to remove the

cache and artifacts folders).

**DEPLOYING SMART CONTRACTS IN HARDHAT**

As aforementioned, the scripts folder is responsible for

containing deploy scripts. To **deploy a contract**, we have to first

get hold of the compiled contract and to do this, we can use the

***hardhat runtime environment***which is available on the **global**

**scope,** intsead of working with the ethers library directly.

The *hardhat runtime environment* is more flexible in that it

**keeps track of all the compiled contracts**, **deployments** and **scripts** in our program; for instance, the ***getContractFactory***

method is used to get the **compiled contract** of our choice.

To deploy a contract, we use the command ***yarn hardhat run scripts/deploy.js***

**HARDHAT NETWORK**

Hardhat has a **built in network just like ganache** and **Remix** that

is designed for testing, debugging and deploying smart contracts.

It runs on the background so anytime a contract is deployed

without specifying a network, hardhat specifies their default

network **(Hardhat Network)** and it comes automatically with an

***RPC URL*** and ***a Private Key*** for signing transactions.

**OVERRIDING THE DEFAULT NETWORK**

In the hardhat configuration file, a networks property could be

added if we want to override the default network provided for

by hardhat. Each network is an object with the following

properties:-

 url (for testnets)

 chainId - Every EVM based network has a chain Id. visit

https://chainlist.org/

 accounts (an array of private keys)

We can use the **--network flag** to specify the network we would be deploying to, since we have listed the networks in the hardhat configuration file.

**VERIFICATION OF SMART CONTRACTS**

A Smart contract can be verified programmatically **right after**

**deployment.** This would allow end users and the public to verify

and ascertain **that the smart contract actually does what it**

**purports to do.**

The easiest way to achieve this is by **using plugins**. So instead of

using Block explorer APIs like that of Etherscan to verify smart

contracts, make use of plugins. An example is the ***hardhat-***

***etherscan*** plugin created by the hardhat team.

- To get the plugin, **run *yarn add --dev @nomiclabs/hardhat-***

***etherscan***

and **import into the configuration file.**

- Next is to get an API Key by creating an account with

etherscan. visit https://etherscan.io/ to create an account.

- Add an etherscan object into the configuration object and inside

the etherscan object, **add a property called apiKey**, whose value

would be the **API key** gotten from etherscan.

- Run ***yarn hardhat* to add the verify task** in the list of tasks. This is because a new plugin was added into the configuration file.

To verify through the command line, run ***npx hardhat verify --***

***network <network> <contract address> <constructor***

***argument>*** or we can simple add a script to the package.json file.

Adding a script command requires that we define a folder that would hold an array of arguments (if any) that the smart contract got deployed with.

Smart Contracts are to be verified when they are deployed ontop a public blockchain and not a local network.