Deploy your Smart Contracts

Part 1: Deploy Scripts

```
async function main() {
 const HelloWorld = await ethers.getContractFactory("HelloWorld");
// Start deployment, returning a promise that resolves to a contract object
   const hello_world = await HelloWorld.deploy("Hello World!");
   console.log("Contract deployed to address:", hello_world.address);
}

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

A `ContractFactory` in ethers.js is an abstraction used to deploy new smart contracts, so `HelloWorld` here is a factory for instances of our hello world contract. When using the `hardhat-ethers` plugin `ContractFactory` and `Contract`, instances are connected to the first signer (owner) by default.

Calling `deploy()` on a `ContractFactory` will start the deployment, and return a `Promise` that resolves to a `Contract` object. This is the object that has a method for each of our smart contract functions.

Command Lines

```
mkdir hello-world
cd hello-world
npm init # (or npm init --yes)

version: (1.0.0)
description: hello world smart contract
entry point: (index.js)
test command:
git repository:
keywords:
author:
license: (ISC)

About to write to /Users/.../.../.../hello-world/package.json:
```

```
{
"name": "hello-world",
"version": "1.0.0",
"description": "hello world smart contract",
"main": "index.js",
"scripts": {
"test": "echo \"Error: no test specified\" && exit 1"
},
"author": "",
"license": "ISC"
}
// Inside our hello-world project run:
npm install --save-dev hardhat
npx hardhat

888    888                      888 888               888
888    888                      888 888               888
888    888                      888 888               888
8888888888  8888b.  888d888 .d88888 88888b.   8888b.  888888
888    888     "88b 888P"  d88" 888 888 "88b     "88b 888
888    888 .d888888 888    888  888 888  888 .d888888 888
888    888 888  888 888    Y88b 888 888  888 888  888 Y88b.
888    888 "Y888888 888     "Y88888 888  888 "Y888888  "Y888

👷 Welcome to Hardhat v2.0.11 👷

What do you want to do? …
Create a sample project
❯ Create an empty hardhat.config.js
Quit

mkdir contracts
mkdir scripts
npm install dotenv --save
```

> In .env file
> API_URL = "https://eth-ropsten.alchemyapi.io/v2/your-api-key"
> PRIVATE_KEY = "your-metamask-private-key"

```
npm install --save-dev @nomiclabs/hardhat-ethers "ethers@^5.0.0"
```

```
/**
* @type import('hardhat/config').HardhatUserConfig
*/
// hardhat.config.js
require('dotenv').config();
require("@nomiclabs/hardhat-ethers");

const { API_URL, PRIVATE_KEY } = process.env;
```

```
module.exports = {
   solidity: "0.7.3",
   defaultNetwork: "ropsten",
   networks: {
      hardhat: {},
      ropsten: {
         url: API_URL,
         accounts: [`0x${PRIVATE_KEY}`]
      }
   },
}
```

```
npx hardhat compile
```

```
// scripts/ folder and create a new file called deploy.js
async function main() {
   const HelloWorld = await ethers.getContractFactory("HelloWorld");

   // Start deployment, returning a promise that resolves to a contract object
   const hello_world = await HelloWorld.deploy("Hello World!");
   console.log("Contract deployed to address:", hello_world.address);
}

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

```
npx hardhat run scripts/deploy.js --network ropsten
Contract deployed to address: 0x6F34E89B84097F5CEF99d82c0a7841007331499f
// check your contract on https://ropsten.etherscan.io/
```

Two important ones to call out here are `eth_sendRawTransaction`
, which is the request to actually write our contract onto the Ropsten chain, and `eth_getTransactionByHash` which is a request to read information about our transaction given the hash (a typical pattern when sending transactions).

Submitting your Smart Contract to Etherscan

Step 1: Generate an API Key on your Etherscan account

1. Select the "My profile" button
2. Navigate to the "API-KEYs" button on the left tab bar. Then press the "Add" button, name your app whatever you wish, and then select continue.
3. Once you've followed the steps above, you should be able to view your new API key. Copy this API key to your clipboard.
4. in .env file

```
API_URL = "https://eth-ropsten.alchemyapi.io/v2/your-api-key"
API_KEY = "your-api-key"
PRIVATE_KEY = "your-private-account-address"
ETHERSCAN_API_KEY = "your-etherscan-key"
```

Step 2.1 Install the `hardhat-etherscan` plugin

First, install the `hardhat-etherscan` plugin to automatically verify your smart contract's source code and ABI on Etherscan. In your project directory run:

```
npm install --save-dev @nomiclabs/hardhat-etherscan
```

Once installed, include the following statement at the top of your `hardhat.config.js` , and add the Etherscan config options:

```
// hardhat.config.js

require('dotenv').config();
require("@nomiclabs/hardhat-ethers");
require("@nomiclabs/hardhat-etherscan");

const { API_URL, PRIVATE_KEY } = process.env;
const ETHERSCAN_API_KEY = process.env.ETHERSCAN_API_KEY;

module.exports = {
  solidity: "0.7.3",
  defaultNetwork: "ropsten",
  networks: {
      hardhat: {},
      ropsten: {
         url: API_URL,
         accounts: [`0x${PRIVATE_KEY}`]
      }
  },
  etherscan: {
    // Your API key for Etherscan
    // Obtain one at https://etherscan.io/
    apiKey: ETHERSCAN_API_KEY
  }
};
```

Step 2.2 Verify your smart contract on Etherscan!

```
npx hardhat verify --network ropsten DEPLOYED_CONTRACT_ADDRESS 'Hello World!'
```

🔥 Make sure that `DEPLOYED_CONTRACT_ADDRESS` is the address of your deployed smart contract on the Ropsten test network. Also, the last argument, `'Hello World!'` must be the same string value that you used during the deploy step in Part 1.

If all goes well, you should see the following message in your terminal:

```
Successfully submitted source code for contract
contracts/HelloWorld.sol:HelloWorld at 0xdeployed-contract-address
for verification on Etherscan. Waiting for verification result...


Successfully verified contract HelloWorld on Etherscan.
https://ropsten.etherscan.io/address/<contract-address>#contracts
```

Successfully verified contract HelloWorld on Etherscan.
https://ropsten.etherscan.io/address/0x6F34E89B84097F5CEF99d82c0a7841007331499f#code