The Project's Main Concept:

The project's main goal is to destroy the big dawg: the Boss. Let's say the big guy has 1,000,000HP. What players do is when they start the game, they mint a character NFT that has a certain amount of Attack Damage and HP. Players can order their character NFT to attack and deal damage to it. It's kinda like Pokemon.

The main goal?💪

The main goal is players need to work together in order to bring the boss's HP down to 0.

However, life isn't that easy. There's a catch 🤞 ***:***

When a player hits the BigDawg, the player is also wounded. And the players are the ones that suffer the most. Assume the BigDawg has 100,000 HP and the local player has 1,000HP. Each hit is worth 500HP. The player is then half-way through dying. If the NFT's HP goes below 0, the player's NFT dies and they can't hit the boss anymore. Players can only have one character NFT in their wallet. Once the character's NFT dies, it's game over. That means many players need to join forces to attack the boss and kill it.

The important thing to know here is that the characters themselves are NFTs.

So, when a player goes to play the game:

1. They'll connect their wallet.
2. Our game will detect they don't have a character NFT in their wallet.
3. I let them choose a character and mint their own character NFT to play the game. Each character NFT has its own attributes stored on the NFT directly like: HP, Attack Damage, the image of the character, etc. So, when the character's HP hits 0, it would say hp: 0 on the NFT itself.

Now… The exciting part: building it ⛪🧱

So first I need to get our local Ethereum Network Working. We need to compile and test our smart contract code! This basically sets up our local environment. The only twist I will have to use is the blockchain. Smartcode is a code that lives on the blockchain. It's like Heruko or AWS, except no one actually owns it. It's ran by thousands of people named 'Miners'! Pretty Cool right?

The Bigger Picture is,

1. Start will write a smart contract. That contract has all the logic around our actual game.
2. My smart contract will be deployed to the local blockchain. This means anyone can access my smart contract and run my game. (I'm giving access. I know I'm a nice guy 🙄).
3. Building the client website. The website will help people easily connect their Ethereum Wallet and play our game.

Okay… Enough Talking 🗣️

I will use HardHat. A local tool that will let us run our smart contracts and test them locally. Now let's head to the terminal.

```
mkdir epic-NFT
cd epic-NFT
npm init -y
npm install --save-dev hardhat
```

🔨 **Get sample project working**

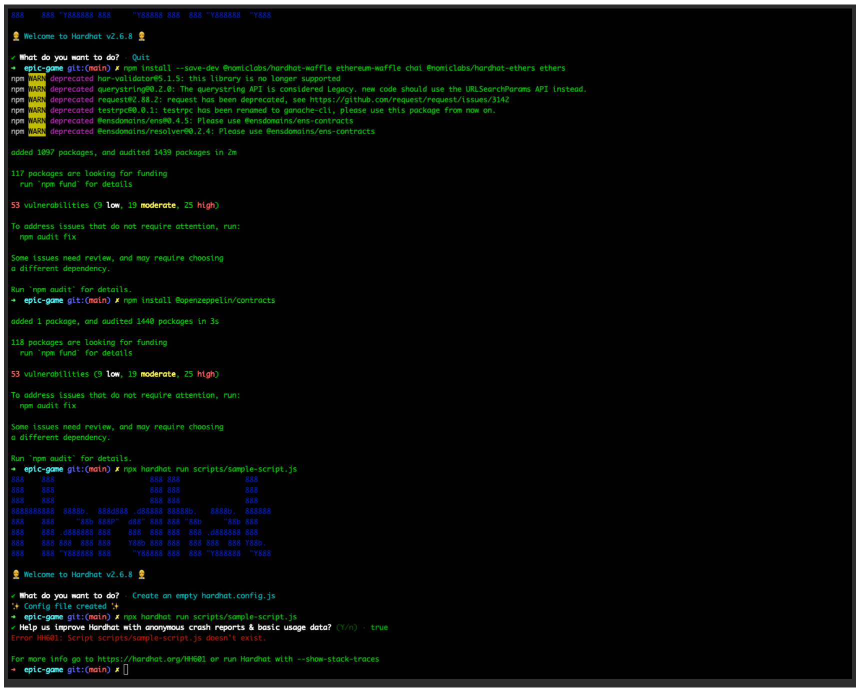
```
npx hardhat
```

You'll also want to install something called OpenZeppelin which is another library that's used a lot to develop secure smart contracts. We'll learn more about it later. For now, just install it :).

```
npm install @openzeppelin/contracts
```

Then run:

```
npx hardhat run scripts/sample-script.js
```



```
888    888 "Y888888 888     "Y88888 888  888 "Y888888  "Y888

👷 Welcome to Hardhat v2.6.8 👷

✔ What do you want to do? · Quit
➜  epic-game git:(main) ✗ npm install --save-dev @nomiclabs/hardhat-waffle ethereum-waffle chai @nomiclabs/hardhat-ethers ethers
npm WARN deprecated har-validator@5.1.5: this library is no longer supported
npm WARN deprecated querystring@0.2.0: The querystring API is considered Legacy. new code should use the URLSearchParams API instead.
npm WARN deprecated request@2.88.2: request has been deprecated, see https://github.com/request/request/issues/3142
npm WARN deprecated testrpc@0.0.1: testrpc has been renamed to ganache-cli, please use this package from now on.
npm WARN deprecated @ensdomains/ens@0.4.5: Please use @ensdomains/ens-contracts
npm WARN deprecated @ensdomains/resolver@0.2.4: Please use @ensdomains/ens-contracts

added 1097 packages, and audited 1439 packages in 2m

117 packages are looking for funding
  run `npm fund` for details

53 vulnerabilities (9 low, 19 moderate, 25 high)

To address issues that do not require attention, run:
  npm audit fix

Some issues need review, and may require choosing
a different dependency.

Run `npm audit` for details.
➜  epic-game git:(main) ✗ npm install @openzeppelin/contracts

added 1 package, and audited 1440 packages in 3s

118 packages are looking for funding
  run `npm fund` for details

53 vulnerabilities (9 low, 19 moderate, 25 high)

To address issues that do not require attention, run:
  npm audit fix

Some issues need review, and may require choosing
a different dependency.

Run `npm audit` for details.
➜  epic-game git:(main) ✗ npx hardhat run scripts/sample-script.js
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 v2.6.8 👷

✔ What do you want to do? · Create an empty hardhat.config.js
✨ Config file created ✨
➜  epic-game git:(main) ✗ npx hardhat run scripts/sample-script.js
✔ Help us improve Hardhat with anonymous crash reports & basic usage data? (Y/n) · true
Error HH601: Script scripts/sample-script.js doesn't exist.

For more info go to https://hardhat.org/HH601 or run Hardhat with --show-stack-traces
➜  epic-game git:(main) ✗ 
```

Okay Sick! Now We get our local environment working 🔨 .

<u>Now Let’s write our smart contract.</u>

Under our contract folder. I will create ***<u>MyEpicGame.sol</u>*** and type our first Basic Contract.

```
pragma solidity ^0.8.0;
import "hardhat/console.sol";
contract MyEpicGame {
  constructor() {
    console.log("THIS IS MY GAME CONTRACT. NICE.");
  }
}
```

Source code has all the Explanations behind all these codes.

<u>Now the smart Contract is completed. I need to run it. How? Let me look that up lol 😂</u>

Going into the **<u>scripts</u>** folder we need to create a file named **<u>“run.js”.</u>**

```
const main = async () => {
 //It complies our contract and generate the necessary files we need to work with our contract under the arifacts directory.
 const gameContractFactory = await hre.ethers.getContractFactory("MyEpicGame");

 //This pretty fancy. Hardhat creates a local Ethereum network for my contract only. The script runs and then destroys the local network.
 //The benifit of that is that creates a brand new fresh blockchain in order to make it easier to debug.
 const gameContract = await gameContractFactory.deploy();

 //The creates fake "miners" on your machine to try to its best to imitate the actual blockchain.
 //Minning is the process of adding transactions to the large distributed public ledger of
 //existing transactions, known as the blockchain
 await gameContract.deployed();

 console.log("Contract deployed to: ", gameContract.address);
};

const runMain = async () => {
   try {
       await main();
       process.exit(0);
   } catch (error) {
       console.log(error);
       process.exit(1);
   }
};

runMain();
```

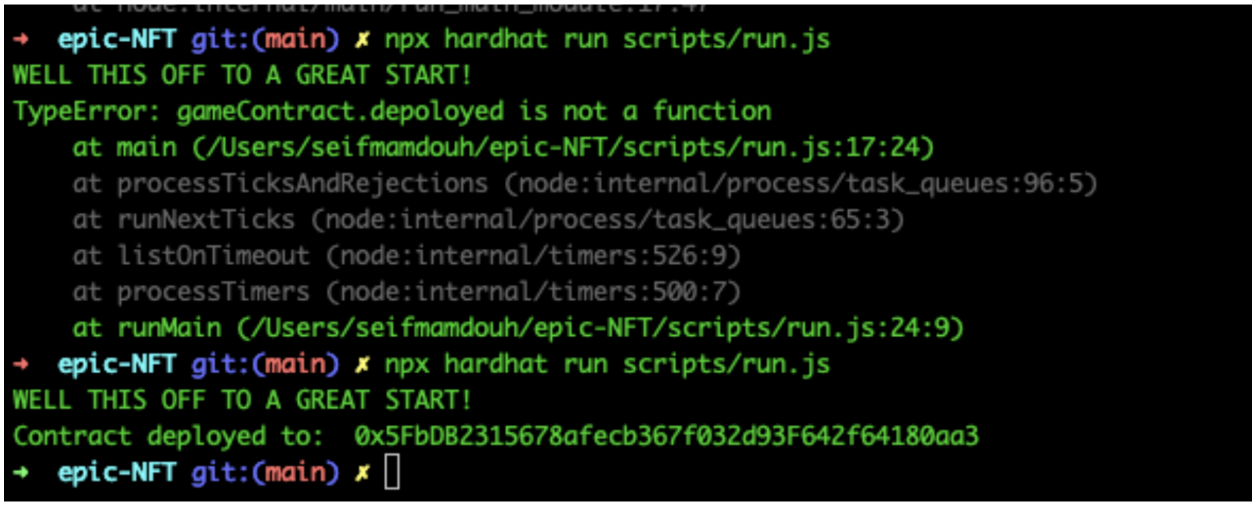
💨 Run it.

Before you run this, make sure to change solidity: "0.8.4", to solidity: "0.8.0", in your hard hat.config.js.

Let's run it! Open up your terminal and run:

```
npx hardhat run scripts/run.js
```

I should see my console.log run from within the contract!



```
→ epic-NFT git:(main) ✗ npx hardhat run scripts/run.js
WELL THIS OFF TO A GREAT START!
TypeError: gameContract.depoloyed is not a function
    at main (/Users/seifmamdouh/epic-NFT/scripts/run.js:17:24)
    at processTicksAndRejections (node:internal/process/task_queues:96:5)
    at runNextTicks (node:internal/process/task_queues:65:3)
    at listOnTimeout (node:internal/timers:526:9)
    at processTimers (node:internal/timers:500:7)
    at runMain (/Users/seifmamdouh/epic-NFT/scripts/run.js:24:9)
→ epic-NFT git:(main) ✗ npx hardhat run scripts/run.js
WELL THIS OFF TO A GREAT START!
Contract deployed to:  0x5FbDB2315678afecb367f032d93F642f64180aa3
→ epic-NFT git:(main) ✗ 
```

Holy Moly! It worked! Okay this off to a great start!

<u>Now I gotta SetUp my local NFT'S.</u> (Praying it goes smooth 🤞)

1. First, data for the NFTS.
    a. Each character will have a few attributes: an image, a name, HP value, and attack damage value. **These attributes will live directly on the NFT itself.**

Now we code:

Go into **Contracts** folder, into the **MyEpicGame.sol,** we gonna update something.

```solidity
// SPDX-License-Identifier: MIT
// This is the version of the Solidity compiler we want our contract to use.
// It basically says "when running this, I only want to use version 0.8.0 of the Solidity compiler,
// nothing lower. nNote, be sure your compiler is set to 0.8.0 in hardhat.config.js.
pragma solidity ^0.8.0;

import "hardhat/console.sol";

contract MyEpicGame {
 // We'll hold our character's attributes in a struct. Feel free to add
 // whatever you'd like as an attribute! (ex. defense, crit chance, etc).
 struct CharacterAttributes {
   uint characterIndex;
   string name;
   string imageURI;
   uint hp;
   uint maxHp;
   uint attackDamage;
 }
 // A lil array to help us hold the default data for our characters.
 // This will be helpful when we mint new characters and need to know
 // things like their HP, AD, etc.
 CharacterAttributes[] defaultCharacters;

 // Data passed in to the contract when it's first created initializing the characters.
 // We're going to actually pass these values in from from run.js.
 constructor(
   string[] memory characterNames,
   string[] memory characterImageURIs,
   uint[] memory characterHp,
   uint[] memory characterAttackDmg
 )
 {
   //Loop through all the characters, and save their values in our contract so
   //we can use them later when we mint our NFTs.
   for(uint i = 0; i < characterNames.length; i++){
     defaultCharacters.push(CharacterAttributes({
       characterIndex: i,
       name: characterNames[i],
       imageURI: characterImageURIs[i],
       hp: characterHp[i],
       maxHp: characterHp[i],
       attackDamage: characterAttackDmg[i]
     }));

     CharacterAttributes memory c = defaultCharacters[i];
     console.log("Done initializing %s w/ HP %s, img %s", c.name, c.hp, c.imageURI);
   }
 }
}
```

Okay I get it there's you are prolly confused. Honestly, so am I lmfaooo. Let's take it slow.

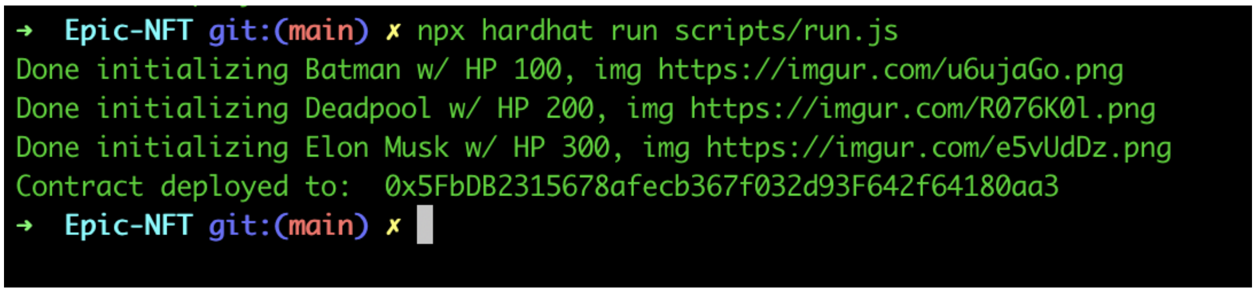
1. First we created a struct to hold all the Default attributes for all new NFTs coming into the game
2. An Array to hold the default data for our character. I actually store each character in an array called **`defaultCharacters.`** The array will be passed through the run.js file we run it. All this gives me is easy access to each character. For example, I can just do defaultCharacters[0] and get access to the default attributes of the first character. This is useful for when we mint our NFTs and need to initialize their data!
3. A for loop that will loop through all the characters and save their values in our smart contract so we can use them later when we mint out NFTS!
4. Lastly we will console.log that the program is done initializing our Default NFTs

Wooohoooo, okay one more thing. We need to update the run.js.

```
const gameContract = await gameContractFactory.deploy(
     ["Batman", "Deadpool", "Elon Musk"], //Names of chacters
     [
       "https://imgur.com/u6ujaGo.png", // Images
       "https://imgur.com/R076K0l.png",
       "https://imgur.com/e5vUdDz.png",
     ],
     [100, 200, 300], // HP values
     [100, 50, 25] //Attack damage values
   );
```

Don't worry, nothing fancy is. All I did was add characters, their id, name, image, hp value, and attack value, in the **Const gameContract function.**

Okay Sick so now we gotta Run `npx hardhat run scripts/run.js`



```
➜  Epic-NFT git:(main) ✗ npx hardhat run scripts/run.js
Done initializing Batman w/ HP 100, img https://imgur.com/u6ujaGo.png
Done initializing Deadpool w/ HP 200, img https://imgur.com/R076K0l.png
Done initializing Elon Musk w/ HP 300, img https://imgur.com/e5vUdDz.png
Contract deployed to:  0x5FbDB2315678afecb367f032d93F642f64180aa3
➜  Epic-NFT git:(main) ✗ 
```

Our images of our NFT are on the blockchain! Pretty cool huh?