

Lesson 16

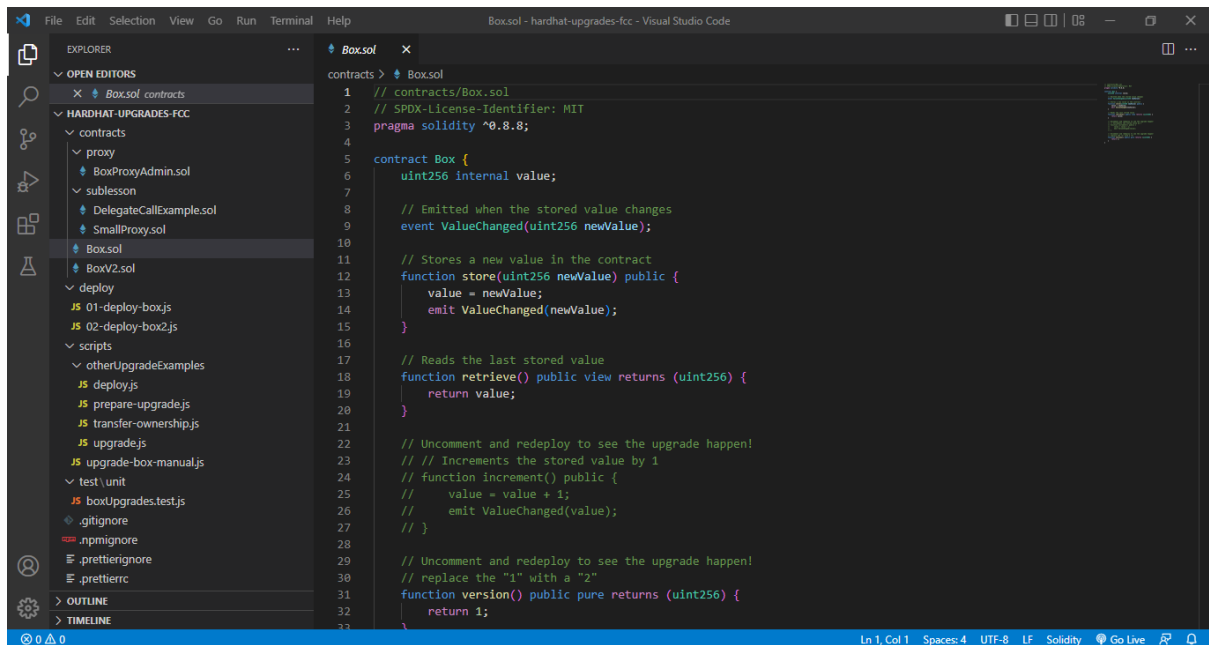
The image displays two screenshots of a Visual Studio Code editor window, showing JavaScript code for Hardhat upgrades.

Top Screenshot: `upgrade-box-manual.js`

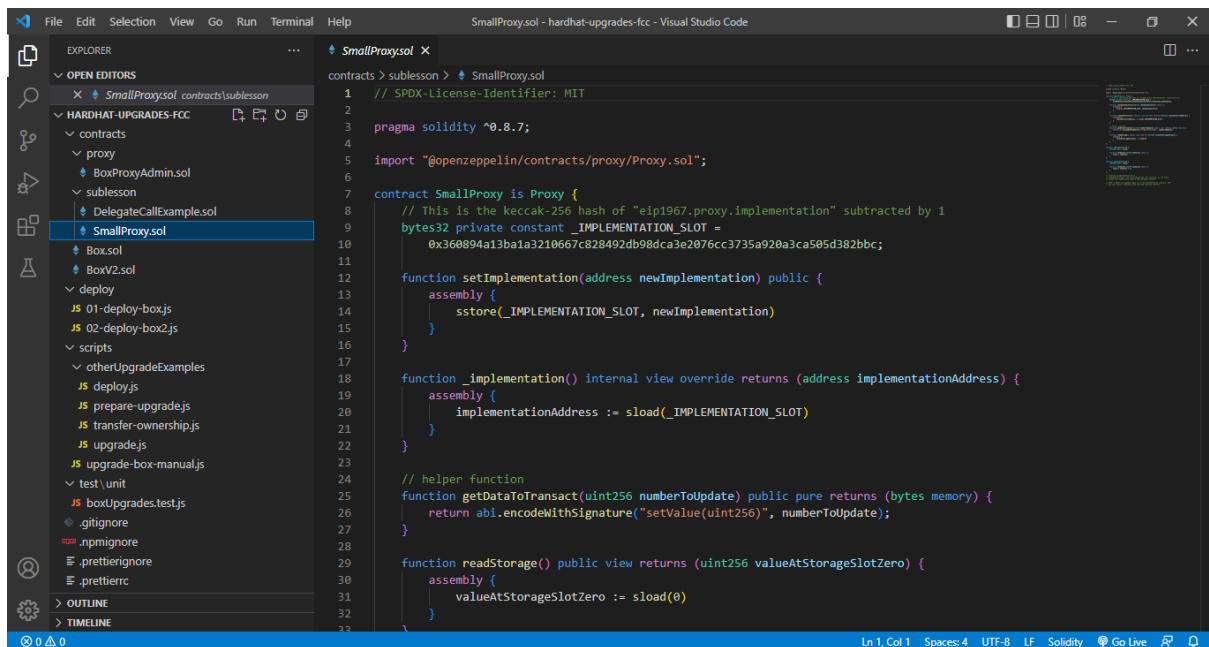
```
1 const { developmentChains, VERIFICATION_BLOCK_CONFIRMATIONS } = require("../helper-hardhat-config")
2 const { network, deployments, deployer } = require("hardhat")
3
4 async function main() {
5   const { deploy, log } = deployments
6   const { deployer } = await getNamedAccounts()
7
8   const waitBlockConfirmations = developmentChains.includes(network.name)
9     ? 1
10     : VERIFICATION_BLOCK_CONFIRMATIONS
11
12   log("-----")
13
14   const boxV2 = await deploy("BoxV2", {
15     from: deployer,
16     args: [],
17     log: true,
18     waitConfirmations: waitBlockConfirmations,
19   })
20
21   // Verify the deployment
22   if (!developmentChains.includes(network.name) && process.env.ETHERSCAN_API_KEY) {
23     log("Verifying...")
24     await verify(boxV2.address, arguments)
25   }
26
27   // Upgrade!
28   // Not "the hardhat-deploy way"
29   const boxProxyAdmin = await ethers.getContract("BoxProxyAdmin")
30   const transparentProxy = await ethers.getContract("Box_Proxy")
31   const upgradeTx = await boxProxyAdmin.upgrade(transparentProxy.address, boxV2.address)
32   await upgradeTx.wait(1)
33   const proxyBox = await ethers.getContractAt("BoxV2", transparentProxy.address)
```

Bottom Screenshot: `01-deploy-box.js`

```
1 const { developmentChains, VERIFICATION_BLOCK_CONFIRMATIONS } = require("../helper-hardhat-config")
2
3 const { network } = require("hardhat")
4
5 module.exports = async ({ getNamedAccounts, deployments }) => {
6   const { deploy, log } = deployments
7   const { deployer } = await getNamedAccounts()
8
9   const waitBlockConfirmations = developmentChains.includes(network.name)
10     ? 1
11     : VERIFICATION_BLOCK_CONFIRMATIONS
12
13   log("-----")
14
15   const box = await deploy("Box", {
16     from: deployer,
17     args: [],
18     log: true,
19     waitConfirmations: waitBlockConfirmations,
20     proxy: {
21       proxyContract: "OpenZeppelinTransparentProxy",
22       viaAdminContract: {
23         name: "BoxProxyAdmin",
24         artifact: "BoxProxyAdmin",
25       },
26     },
27   })
28
29   // Be sure to check out the hardhat-deploy examples to use UUPS proxies!
30   // https://github.com/wighawag/template-ethereum-contracts
31
32   // Verify the deployment
33   if (!developmentChains.includes(network.name) && process.env.ETHERSCAN_API_KEY) {
```



```
contracts > Box.sol
1 // contracts/Box.sol
2 // SPDX-License-Identifier: MIT
3 pragma solidity ^0.8.8;
4
5 contract Box {
6     uint256 internal value;
7
8     // Emitted when the stored value changes
9     event ValueChanged(uint256 newValue);
10
11     // Stores a new value in the contract
12     function store(uint256 newValue) public {
13         value = newValue;
14         emit ValueChanged(newValue);
15     }
16
17     // Reads the last stored value
18     function retrieve() public view returns (uint256) {
19         return value;
20     }
21
22     // Uncomment and redeploy to see the upgrade happen!
23     // // Increments the stored value by 1
24     // function increment() public {
25     //     value = value + 1;
26     //     emit ValueChanged(value);
27     // }
28
29     // Uncomment and redeploy to see the upgrade happen!
30     // replace the "1" with a "2"
31     function version() public pure returns (uint256) {
32         return 1;
33     }
34 }
```



```
contracts > sublesson > SmallProxy.sol
1 // SPDX-License-Identifier: MIT
2
3 pragma solidity ^0.8.7;
4
5 import "@openzeppelin/contracts/proxy/Proxy.sol";
6
7 contract SmallProxy is Proxy {
8     // This is the keccak-256 hash of "eip1967.proxy.implementation" subtracted by 1
9     bytes32 private constant _IMPLEMENTATION_SLOT =
10         0x360894a13ba1a3210667c828492db98dca3e2076cc3735a920a3ca505d382bbc;
11
12     function setImplementation(address newImplementation) public {
13         assembly {
14             sstore(_IMPLEMENTATION_SLOT, newImplementation)
15         }
16     }
17
18     function _implementation() internal view override returns (address implementationAddress) {
19         assembly {
20             implementationAddress := sload(_IMPLEMENTATION_SLOT)
21         }
22     }
23
24     // helper function
25     function getDataToTransact(uint256 numberToUpdate) public pure returns (bytes memory) {
26         return abi.encodeWithSignature("setValue(uint256)", numberToUpdate);
27     }
28
29     function readStorage() public view returns (uint256 valueAtStorageSlotZero) {
30         assembly {
31             valueAtStorageSlotZero := sload(0)
32         }
33     }
34 }
```