下半节

三种可升级合约的区别

常用的三种升级合约，透明升级合约、UUPS升级合约，BEACON升级合约，其核心是用的依然是EIP1967

> ERC-1967: Proxy Storage Slots
> A consistent location where proxies store the address of the logic contract they delegate to, as well as other proxy-specific information.
> https://eips.ethereum.org/EIPS/eip-1967

EIP1967定义了代理升级模式里面核心的数据的存放，例如：

Logic contract address

定义了业务合约的地址

Admin address

定义了管理员合约的地址

Beacon contract address

定义了贝壳合约所在的地址

三种合约的区别在于ADMIN、业务合约地址的组织形式

透明可升级合约要点：

（1）部署完成后生成proxy，admin和业务合约

（2）只有admin的owner才能通过proxyadmin升级业务合约

UUPS合约要点：

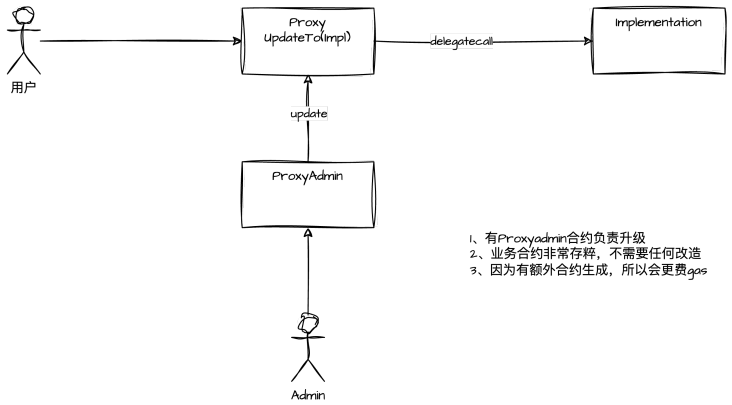
（1）业务合约必须继承UUPSUpgradeable

（2）UUPSUpgradeable提供upgradeToAndCall进行升级，通过Proxy调用

（3）业务合约需要首先_authorizeUpgrade做权限控制

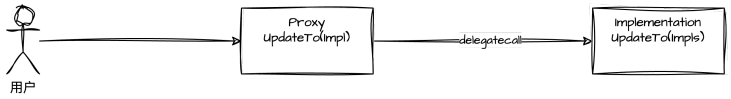
EIP1967

透明可升级合约



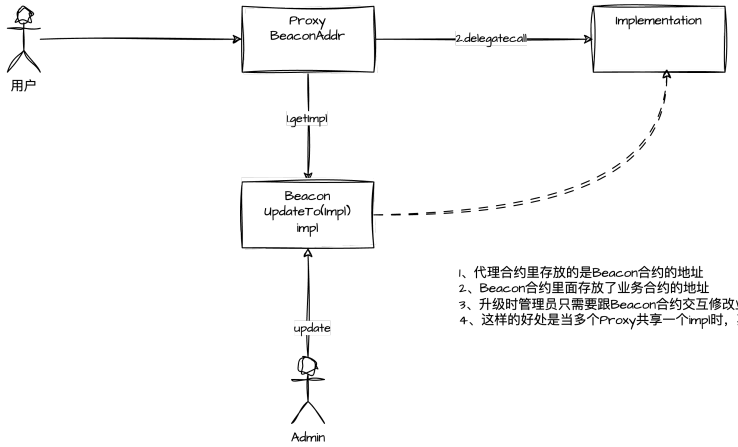
1、有Proxyadmin合约负责升级
2、业务合约非常存粹，不需要任何改造
3、因为有额外合约生成，所以会更费gas

uuPS升级合约



1、业务合约自己负责升级，所以业务合约需要提供升级能力
2、业务合约需要具备权限控制，例如实现ownableUpgradable
3、如果希望合约一直被升级，那么需要一致实现uups接口
4、短期升级更省gas，但是如果会一直升级反而费一些gas

Beacon升级合约



1、代理合约里存放的是Beacon合约的地址
2、Beacon合约里面存放了业务合约的地址
3、升级时管理员只需要跟Beacon合约交互修改业务合约地址
4、这样的好处是当多个Proxy共享一个impl时，其他模式需要每个proxy都升级，但是beacon合约只需要升级一次

代码演示

（1）透明可升级合约

合约代码

```
// SPDX-License-Identifier: MIT
pragma solidity 0.8.27;
contract BoxV1 {

  uint public  x;
    function initialize(uint _val) public  {
        x = _val; // set initial value in initializer
    }
    function cal() external    {
         x=x+1;
    }

}
// SPDX-License-Identifier: MIT
pragma solidity 0.8.27;

contract BoxV2 {

    uint public  x;
    function cal() external {
         x=x*2;
    }



}
```

部署代码

```
const hre = require("hardhat");
async function deploy() {
  //V1版本工厂
  const BOXV1 = await hre.ethers.getContractFactory("BoxV1");

  //通过V1版本部署代理
  const v1 = await hre.upgrades.deployProxy(BOXV1, [1], {
```

```
    initializer: "initialize",
  });
  await v1.waitForDeployment();

  console.log(await v1.getAddress());
  console.log(await v1.x());
  await v1.cal();
  console.log(await v1.x());

  const BOXV2 = await hre.ethers.getContractFactory("BoxV2");
  const v2 = await hre.upgrades.upgradeProxy(await v1.getAddress(), BOX
V2);
  await v2.waitForDeployment();
  console.log(await v2.getAddress());
  console.log(await v2.x());
  await v2.cal();
  console.log(await v2.x());
}

deploy();
```

(2) UUPS升级合约

合约代码

```
// SPDX-License-Identifier: MIT
pragma solidity 0.8.27;
import "@openzeppelin/contracts/proxy/utils/UUPSUpgradeable.sol";
contract BoxV1UUPS is UUPSUpgradeable{

  uint public  x;
    function initialize(uint _val) public  {
        x = _val; // set initial value in initializer
    }

    function _authorizeUpgrade(address newImplementation)   internal overr
ide {
```

```
    }
    function cal() external    {
        x=x+1;
    }

}



// SPDX-License-Identifier: MIT
pragma solidity 0.8.27;
import "@openzeppelin/contracts/proxy/utils/UUPSUpgradeable.sol";
contract BoxV2UUPS  is  UUPSUpgradeable {

    uint public  x;
    function cal() external {
        x=x*2;
    }

   function _authorizeUpgrade(address newImplementation)   internal overri
de {

    }



}
```

部署代码

```
const hre = require("hardhat");
async function deploy() {
  //V1版本工厂
  const _UUPSV1 = await hre.ethers.getContractFactory("UUPSV1");
  //V2版本工厂
  const _UUPSV2 = await hre.ethers.getContractFactory("UUPSV2");

  //通过V1版本部署代理
  const v1 = await hre.upgrades.deployProxy(_UUPSV1, [1], {
    initializer: "initialize",
```

```
    kind: "uups",
  });
  await v1.waitForDeployment();

  console.log(await v1.getAddress());
  console.log(await v1.x());
  await v1.cal();
  console.log(await v1.x());

  //升级成V2版本
  await hre.upgrades.upgradeProxy(await v1.getAddress(), _UUPSV2);
  console.log(await v1.x());
  await v1.cal();
  console.log(await v1.x());
}

deploy();
```

(3) Beacon升级合约

合约代码

沿用透明升级合约的代码

部署代码

```
const hre = require("hardhat");
async function deploy() {
  //V1版本工厂
  const BOXV1 = await hre.ethers.getContractFactory("BoxV1");

  //通过V1版本部署代理
  const beacon = await hre.upgrades.deployBeacon(BOXV1);
  await beacon.waitForDeployment();

  const proxy = await hre.upgrades.deployBeaconProxy(beacon, BOXV1,
[1]);
  await proxy.waitForDeployment();
```

```
  console.log(await proxy.getAddress());
  console.log(await proxy.x());
  await proxy.cal();
  console.log(await proxy.x());

  const BOXV2 = await hre.ethers.getContractFactory("BoxV2");
  await hre.upgrades.upgradeBeacon(await beacon.getAddress(), BOXV2);

  console.log(await proxy.getAddress());
  console.log(await proxy.x());
  await proxy.cal();
  console.log(await proxy.x());
}

deploy();
```

项目串讲

Launchpad

（1）Launchpad的主要业务流程

讲ppt，讲空投、sales、业务流程

（2）Launchpad的代码目录介绍

重点强调优化后的目录结构

（3）Launchpad的合约走读

重点讲几个合约的关系，面向接口编程与工厂模式

（4）Launchpad的简历包装

⚡ 简历优化

NFTMarket

2.2、技术方案