angry–baby

todo

手动部署OpenZeppelin可升级合约

首发：https://learnblockchain.cn/article/2758

手动部署可升级能更好的理解部署过程、原理，主要原因是本人对前端工具使用不熟。以下只是本人学习时的操作记录，仅分享。使用remix部署。

首次部署

需要部署三个合约，分别是逻辑合约，ProxyAdmin，TransparentUpgradeProxy。逻辑合约就是我们自己的业务合约，需要满足OpenZeppelin可升级合约的条件。以下业务合约以Params合约为例进行说明。

业务合约Params

```
// SPDX-License-Identifier: MIT

pragma solidity ^0.8.0;

import "@openzeppelin/contracts-upgradeable/proxy/utils/Initializable.sol";
import "@openzeppelin/contracts-upgradeable/access/OwnableUpgradeable.sol";

contract Params is Initializable,OwnableUpgradeable {
    function initialize()public initializer{
        __Context_init_unchained();
        __Ownable_init_unchained();
    }
    mapping(string => uint256) private uint256Params;

    event Uint256ParamSetted(string indexed _key,uint256 _value);

    function SetUint256Param(string memory _key,uint256 _value) external onlyOwner{
        uint256Params[_key] = _value;
        emit Uint256ParamSetted(_key,_value);
    }


    function GetUint256Param(string memory _key)public view returns(uint256){
        return uint256Params[_key];
    }
}
```

部署后地址为：0xe2899bddFD890e320e643044c6b95B9B0b84157A，先不进行初始化（initialize，本方法对应的code为 0x8129fc1c ）

ProxyAdmin 管理合约

```
// SPDX-License-Identifier: MIT

pragma solidity ^0.8.0;

import "@openzeppelin/contracts/proxy/transparent/TransparentUpgradeableProxy.sol";
import "@openzeppelin/contracts/access/Ownable.sol";

/**
 * @dev This is an auxiliary contract meant to be assigned as the admin of a
{TransparentUpgradeableProxy}. For an
 * explanation of why you would want to use this see the documentation for
{TransparentUpgradeableProxy}.
 */
contract ProxyAdmin is Ownable {
    /**
     * @dev Returns the current implementation of `proxy`.
     *
     * Requirements:
     *
     * - This contract must be the admin of `proxy`.
     */
    function getProxyImplementation(TransparentUpgradeableProxy proxy) public view
virtual returns (address) {
        // We need to manually run the static call since the getter cannot be flagged
as view
        // bytes4(keccak256("implementation()")) == 0x5c60da1b
        (bool success, bytes memory returndata) =
address(proxy).staticcall(hex"5c60da1b");
        require(success);
        return abi.decode(returndata, (address));
    }

    /**
     * @dev Returns the current admin of `proxy`.
     *
     * Requirements:
     *
     * - This contract must be the admin of `proxy`.
     */
    function getProxyAdmin(TransparentUpgradeableProxy proxy) public view virtual
returns (address) {
        // We need to manually run the static call since the getter cannot be flagged
as view
        // bytes4(keccak256("admin()")) == 0xf851a440
        (bool success, bytes memory returndata) =
address(proxy).staticcall(hex"f851a440");
        require(success);
        return abi.decode(returndata, (address));
    }

    /**
     * @dev Changes the admin of `proxy` to `newAdmin`.
     *
     * Requirements:
     *
     * - This contract must be the current admin of `proxy`.
     */
    function changeProxyAdmin(TransparentUpgradeableProxy proxy, address newAdmin)
public virtual onlyOwner {
        proxy.changeAdmin(newAdmin);
    }

    /**
     * @dev Upgrades `proxy` to `implementation`. See {TransparentUpgradeableProxy-
upgradeTo}.
     *
     * Requirements:
     *
     * - This contract must be the admin of `proxy`.
     */
    function upgrade(TransparentUpgradeableProxy proxy, address implementation) public
virtual onlyOwner {
        proxy.upgradeTo(implementation);
    }

    /**
     * @dev Upgrades `proxy` to `implementation` and calls a function on the new
implementation. See
     * {TransparentUpgradeableProxy-upgradeToAndCall}.
     *
     * Requirements:
     *
     * - This contract must be the admin of `proxy`.
     */
    function upgradeAndCall(
        TransparentUpgradeableProxy proxy,
        address implementation,
        bytes memory data
    ) public payable virtual onlyOwner {
        proxy.upgradeToAndCall{value: msg.value}(implementation, data);
    }
}
```

部署后地址为：0x1c91347f2A44538ce62453BEBd9Aa907C662b4bD

TransparentUpgradeableProxy 代理合约，DAPP直接交互的合约地址

```
// SPDX-License-Identifier: MIT

pragma solidity ^0.8.0;

import "@openzeppelin/contracts/proxy/ERC1967/ERC1967Proxy.sol";

/**
 * @dev This contract implements a proxy that is upgradeable by an admin.
 *
 * To avoid https://medium.com/nomic-labs-blog/malicious-backdoors-in-ethereum-proxies-
62629adf3357[proxy selector
 * clashing], which can potentially be used in an attack, this contract uses the
 * https://blog.openzeppelin.com/the-transparent-proxy-pattern/[transparent proxy
pattern]. This pattern implies two
 * things that go hand in hand:
 *
 * 1. If any account other than the admin calls the proxy, the call will be forwarded
to the implementation, even if
 * that call matches one of the admin functions exposed by the proxy itself.
 * 2. If the admin calls the proxy, it can access the admin functions, but its calls
will never be forwarded to the
 * implementation. If the admin tries to call a function on the implementation it will
fail with an error that says
 * "admin cannot fallback to proxy target".
 *
 * These properties mean that the admin account can only be used for admin actions like
upgrading the proxy or changing
 * the admin, so it's best if it's a dedicated account that is not used for anything
else. This will avoid headaches due
 * to sudden errors when trying to call a function from the proxy implementation.
 *
 * Our recommendation is for the dedicated account to be an instance of the
{ProxyAdmin} contract. If set up this way,
 * you should think of the `ProxyAdmin` instance as the real administrative interface
of your proxy.
 */
contract TransparentUpgradeableProxy is ERC1967Proxy {
    /**
     * @dev Initializes an upgradeable proxy managed by `_admin`, backed by the
implementation at `_logic`, and
     * optionally initialized with `_data` as explained in {ERC1967Proxy-constructor}.
     */
    constructor(
        address _logic,
        address admin_,
        bytes memory _data
    ) payable ERC1967Proxy(_logic, _data) {
        assert(_ADMIN_SLOT == bytes32(uint256(keccak256("eip1967.proxy.admin")) - 1));
        _changeAdmin(admin_);
    }

    /**
     * @dev Modifier used internally that will delegate the call to the implementation
unless the sender is the admin.
     */
    modifier ifAdmin() {
        if (msg.sender == _getAdmin()) {
            _;
        } else {
            _fallback();
        }
    }

    /**
     * @dev Returns the current admin.
     *
     * NOTE: Only the admin can call this function. See {ProxyAdmin-getProxyAdmin}.
     *
     * TIP: To get this value clients can read directly from the storage slot shown
below (specified by EIP1967) using the
     * https://eth.wiki/json-rpc/API#eth_getstorageat[`eth_getStorageAt`] RPC call.
     * `0xb53127684a568b3173ae13b9f8a6016e243e63b6e8ee1178d6a717850b5d6103`
     */
    function admin() external ifAdmin returns (address admin_) {
        admin_ = _getAdmin();
    }

    /**
     * @dev Returns the current implementation.
     *
     * NOTE: Only the admin can call this function. See {ProxyAdmin-
getProxyImplementation}.
     *
     * TIP: To get this value clients can read directly from the storage slot shown
below (specified by EIP1967) using the
     * https://eth.wiki/json-rpc/API#eth_getstorageat[`eth_getStorageAt`] RPC call.
     * `0x360894a13ba1a3210667c828492db98dca3e2076cc3735a920a3ca505d382bbc`
     */
    function implementation() external ifAdmin returns (address implementation_) {
        implementation_ = _implementation();
    }

    /**
     * @dev Changes the admin of the proxy.
     *
     * Emits an {AdminChanged} event.
     *
     * NOTE: Only the admin can call this function. See {ProxyAdmin-changeProxyAdmin}.
     */
    function changeAdmin(address newAdmin) external virtual ifAdmin {
        _changeAdmin(newAdmin);
    }

    /**
     * @dev Upgrade the implementation of the proxy.
     *
     * NOTE: Only the admin can call this function. See {ProxyAdmin-upgrade}.
     */
    function upgradeTo(address newImplementation) external ifAdmin {
        _upgradeToAndCall(newImplementation, bytes(""), false);
    }

    /**
     * @dev Upgrade the implementation of the proxy, and then call a function from the
new implementation as specified
     * by `data`, which should be an encoded function call. This is useful to
initialize new storage variables in the
     * proxied contract.
     *
     * NOTE: Only the admin can call this function. See {ProxyAdmin-upgradeAndCall}.
     */
    function upgradeToAndCall(address newImplementation, bytes calldata data) external
payable ifAdmin {
        _upgradeToAndCall(newImplementation, data, true);
    }

    /**
     * @dev Returns the current admin.
     */
    function _admin() internal view virtual returns (address) {
        return _getAdmin();
    }

    /**
     * @dev Makes sure the admin cannot access the fallback function. See {Proxy-
_beforeFallback}.
     */
    function _beforeFallback() internal virtual override {
        require(msg.sender != _getAdmin(), "TransparentUpgradeableProxy: admin cannot
fallback to proxy target");
        super._beforeFallback();
    }
}
```

部署需要参数，如下:

- _LOGIC:逻辑合约地址，这里为 0xe2899bddFD890e320e643044c6b95B9B0b84157A
- ADMIN_：管理合约地址，这里为 0x1c91347f2A44538ce62453BEBd9Aa907C662b4bD
- _DATA：逻辑合约初始化方法调用数据，这里为 `0x8129fc1c` （只调用initialize方法，initialize方法没有入参，如果有参数也是支持的）

部署后地址为：0x93f8dddd876c7dBE3323723500e83E202A7C96CC

以上，可升级合约部署完毕，DAPP可以调用 `0x93f8dddd876c7dBE3323723500e83E202A7C96CC` 访问Params合约。

测试用例：调用SetUint256Param，设置_key=K,_VALUE=1，并通过GetUint256Param进行验证。

合约升级

修改Params合约，并部署

```
function GetUint256Param(string memory _key)public view returns(uint256){
        uint256 v = uint256Params[_key];
        return v+1;
    }
```

直接部署Params合约，部署后地址为：0x406AB5033423Dcb6391Ac9eEEad73294FA82Cfbc

调用ProxyAdmin进行升级

ProxyAdmin提供两个方法进行升级

- upgrade，需要传入proxy地址，新的逻辑实现地址
- upgradeAndCall，需要传入roxy地址，新的逻辑实现地址，初始化调用数据

本例中，由于数据是保存在代理合约中，这份数据已经初始化过了，不需要再初始化，所以调用upgrade方法即可，参数如下:

- proxy:0x93f8dddd876c7dBE3323723500e83E202A7C96CC
- implementation:0x406AB5033423Dcb6391Ac9eEEad73294FA82Cfbc

至此，合约升级完毕。调用GetUint256Param方法进行验证，获得_key=K的value为2，合约升级成功。

注意事项

- 可升级合约的存储不能乱，即：只能新增存储项，不能修改顺序
- 没有构造函数，使用initialize替代
- 继承的父合约也需要能满足升级，本例中的Ownable采用OwnableUpgradeable，支持升级
- 可使用OpenZeppelin插件验证合约是否为可升级合约，以及升级时是否有冲突

分类: other

好文要顶 关注我 收藏该文

angry-baby
粉丝 - 6 关注 - 0
+加关注

0 推荐 0 反对

« 上一篇： solidity“abi.encode/abi.encodePacked”使用golang编码
» 下一篇： library Checkpoints:Transform openzeppelin

posted on 2021-07-20 09:50 angry-baby 阅读(2614) 评论(0) 编辑 收藏 举报

刷新评论 刷新页面 返回顶部

登录后才能查看或发表评论，立即 登录 或者 逛逛 博客园首页

编辑推荐：
· 闲来无事-树莓派控制风扇启停
· ASP.NET Core 6框架揭秘实例演示[40]：基于角色的授权
· 记一次字符串末尾空白丢失的排查，MySQL 是会玩的！
· 记一次 .NET 某旅行社审批系统 崩溃分析
· C#/.Net的多播委托到底是啥？彻底剖析下

阅读排行：
· C# 实现 Linux 视频聊天、远程桌面（源码，支持信创国产化环境，银河麒麟，统信UOS）
· 11k+ Star 一款更适合中国用户的开源 BI 工具
· 发布一个Visual Studio 2022 插件,可以自动完成构造函数依赖注入代码
· 三年，能否成为一名真正的架构师
· 记一次.net加密神器 Eazfuscator.NET 2023.2 最新版 使用尝试

公告

昵称： angry-baby
园龄： 7年1个月
粉丝： 6
关注： 0
+加关注

导航

博客园
首页
新随笔
联系
订阅 XML
管理

<			2023年6月			>
日	一	二	三	四	五	六
28	29	30	31	1	2	3
4	5	6	7	8	9	10
11	12	13	14	15	16	17
18	19	20	21	22	23	24
25	26	27	28	29	30	1
2	3	4	5	6	7	8

统计

随笔 - 59
文章 - 0
评论 - 3
阅读 - 67095

搜索

找找看
谷歌搜索

常用链接

我的随笔
我的评论
我的参与
最新评论
我的标签

我的标签

golang(5)
micro(5)
freeswitch(4)
定时器(2)
Timer(2)
nsq(2)
nats(2)
thrift(2)
tls(2)
cosmos(2)
更多

随笔分类

C/C++(16)
Golang(24)
other(16)
Python(1)
solidity(2)

随笔档案

2023年5月(1)
2022年12月(1)
2022年6月(1)
2021年7月(1)
2020年10月(1)
2020年9月(3)
2020年5月(1)
2018年12月(1)
2018年11月(6)
2018年9月(1)
2018年7月(1)
2018年5月(1)
2018年3月(1)
2018年1月(2)
2017年12月(37)

阅读排行榜

1. solidity“abi.encode/abi.encodePacked”使用golang编码(5860)
2. 使用NATS替换NSQ为后台服务解耦(5408)
3. Golang微服务：Micro限流、熔断(4425)
4. Golang微服务：Micro Trace使用opentracing jaeger(3346)
5. 使用vendor管理go第三方包(3278)

评论排行榜

1. golang 六宫格、九宫格头像生成(2)
2. mod_conference ESL控制一（原理）(1)

推荐排行榜

1. 【坑】https证书链不完整的坑(1)
2. mod_conference ESL控制一（原理）(1)

最新评论

1. Re:golang 六宫格、九宫格头像生成
@唐奋 任何实现io.Writer都可以，这里是一个内存实现。...
--angry-baby
2. Re:golang 六宫格、九宫格头像生成
fOut := memory.NewWriter()
这个库怎么引的
--唐奋
3. Re:mod_conference ESL控制一（原理）
你好，这种方式发起会议，会议的ID怎么看？ originate user/1001 3001 #呼叫user/1001...
--f_heaven