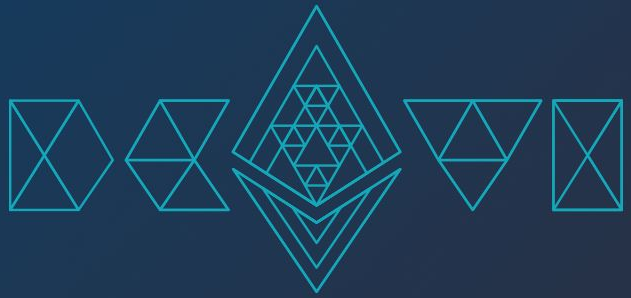


# Five smart contract patterns

## I wish more devs would use

Hadrien Croubois

OpenZeppelin

A lot of the ecosystem's value is the result of smart contract composability,

Be creative with your usage of existing interfaces

# 1. Turn ownable contracts into NFT.

**EIP-173** (Ownable.sol)

- Ownability is often use for access restriction
- Ownership transfer is not supported by many tools

**EIP-721** (ERC721.sol)

- Token represent ownership of an “asset”
- NFT transfer is supported by many tools (OpenSea, …)

```
function addressToUint256(address a) pure returns (uint256) {
    return uint256(uint160(a));
}

abstract contract RegistryOwnable {
    /// @custom:oz-upgrades-unsafe-allow state-variable-immutable
    IERC721 public immutable ownershipRegistry;

    modifier onlyOwner() {
        require(owner() == msg.sender, "RegistryOwnable: caller is not the owner");
        _;
    }

    /// @custom:oz-upgrades-unsafe-allow constructor
    constructor(address ownershipRegistry_)
    {
        ownershipRegistry = IERC721(ownershipRegistry_);
    }

    function owner()
        public
        view
        virtual
        returns (address)
    {
        return ownershipRegistry.ownerOf(addressToUint256(address(this)));
    }

    function transferOwnership(address newOwner) public virtual onlyOwner {
        ownershipRegistry.transferFrom(owner(), newOwner, addressToUint256(address(this)));
    }
}
```

```
contract VestingFactory is ERC721("Vestings", "Vestings"), Multicall
{
    address public immutable template = address(new VestingTemplate(address(this)));

    function newVesting(
        address beneficiaryAddress,
        uint64 startTimestamp,
        uint64 cliffDuration,
        uint64 vestingDuration
    )
        external
        returns (address)
    {
        address instance = Clones.clone(template);
        VestingTemplate(payable(instance)).initialize(startTimestamp, cliffDuration, vestingDuration);
        _mint(beneficiaryAddress, addressToUint256(instance));
        return instance;
    }

    function _isApprovedOrOwner(address spender, uint256 tokenId)
        internal
        view
        override
        returns (bool)
    {
        return super._isApprovedOrOwner(spender, tokenId) || addressToUint256(spender) == tokenId;
    }
}
```

# 2. What about a universal registry for ethereum address ownership?

```
contract AddressOwnershipRegistry is ERC721("Eth Address", "EthAddr"), Multicall
{
    function _ownerOf(uint256 tokenId) internal view override returns (address) {
        require(tokenId <= type(uint160).max);

        address o = super._ownerOf(tokenId);
        return o == address(0) ? address(uint160(tokenId)) : o;
    }

    function balanceOf(address owner) public view override returns (uint256) {
        unchecked {
            return super.balanceOf(owner) + 1;
        }
    }

    function _isApprovedOrOwner(address spender, uint256 tokenId)
        internal
        view
        override
        returns (bool)
    {
        return super._isApprovedOrOwner(spender, tokenId)
            || (spender.code.length > 0 && uint256(uint160((spender))) == tokenId);
    }

    function _burn(uint256 tokenId) internal override {
        _transfer(ownerOf(tokenId), address(uint160(tokenId)), tokenId);
    }
}
```

# 3. Combine Ownable & AccessControl

**EIP-173** (Ownable.sol)

- One single owner
- Ownership transfer (NFT-like)
- “onlyOwner” modifier

AccessControl.sol

- Many roles
- Many members for each role
- Roles administer other roles
- Grant / Revoke / Renounce
- “onlyRole(bytes32)” modifier

```
abstract contract AccessControlOwnable is AccessControl, Ownable
{
    function hasRole(bytes32 role, address account) public view override returns (bool)
    {
        return role == DEFAULT_ADMIN_ROLE
            ? account == owner()
            : super.hasRole(role, account);
    }

    function _grantRole(bytes32 role, address account) internal override
    {
        require(role != DEFAULT_ADMIN_ROLE, "Admin role is managed by owner");
        super._grantRole(role, account);
    }

    function _revokeRole(bytes32 role, address account) internal override
    {
        require(role != DEFAULT_ADMIN_ROLE, "Admin role is managed by owner");
        super._revokeRole(role, account);
    }
}
```

The ecosystem is constantly changing,

Try to be ahead of the curve

# 4. Let prepare for the next hard-fork

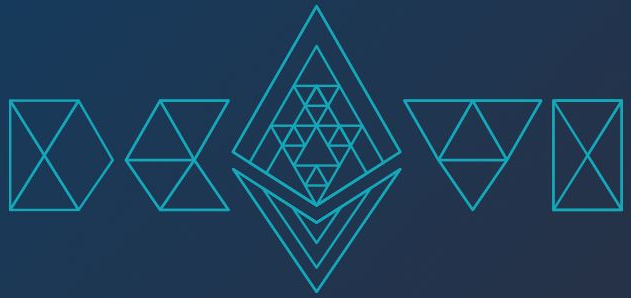
```
abstract contract ForkProtection {
    uint256 private immutable INITIAL_CHAIN_ID = block.chainid;

    modifier onlyInitialChain() {
        require(_isInitialChain());
        _;
    }

    modifier onlyForkedChain() {
        require(!_isInitialChain());
        _;
    }

    function _isInitialChain() internal view returns (bool) {
        return block.chainid == INITIAL_CHAIN_ID;
    }
}

contract SomeBridge is ForkProtection, Ownable {
    function exit(/* args */) external onlyInitialChain() {
        //... do stuff
    }

    function recover(/* args */) external onlyForkedChain() onlyOwner() {
        //... do stuff
    }
}
```

# 5. Use Multicall to empower your users and UIs

```
/**
 * @dev Provides a function to batch together multiple calls in a single external call.
 *
 * _Available since v4.1._
 */
abstract contract Multicall {
    /**
     * @dev Receives and executes a batch of function calls on this contract.
     */
    function multicall(bytes[] calldata data) external virtual returns (bytes[] memory results) {
        results = new bytes[](data.length);
        for (uint256 i = 0; i < data.length; i++) {
            results[i] = Address.functionDelegateCall(address(this), data[i]);
        }
        return results;
    }
}
```

```
contract VestingFactory is ERC721("Vestings", "Vestings"), Multicall
{
    address public immutable template = address(new VestingTemplate(address(this)));

    function newVesting(
        address beneficiaryAddress,
        uint64 startTimestamp,
        uint64 cliffDuration,
        uint64 vestingDuration
    )
        external
        returns (address)
    {
        address instance = Clones.clone(template);
        VestingTemplate(payable(instance)).initialize(startTimestamp, cliffDuration, vestingDuration);
        _mint(beneficiaryAddress, addressToUint256(instance));
        return instance;
    }

    function _isApprovedOrOwner(address spender, uint256 tokenId)
        internal
        view
        override
        returns (bool)
    {
        return super._isApprovedOrOwner(spender, tokenId) || addressToUint256(spender) == tokenId;
    }
}
```

```
(async () => {

    const factory = await attach("VestingFactory", "0x3c912349aB2AcA8D6a573a34acfA9Ff26D49B7f9"); // goerli
    const encoded = config.map(entry => factory.interface.encodeFunctionData("newVesting", [
        entry.addr,
        toTimestamp(entry.start),
        entry.cliff,
        entry.duration
    ]));

    await factory.multicall(encoded)
        .then(tx => tx.wait())
        .then(receipt => console.log({
            tx:    receipt.transactionHash,
            addrs: receipt.events
                .filter(ev => ev.eventSignature == 'Transfer(address,address,uint256)')
                .map(ev => ethers.utils.hexlify(ethers.utils.zeroPad(ev.args.tokenId, 20))),
        }));

})().catch(console.error);
```



# Thank you!

Hadrien Croubois

OpenZeppelin

@Amxx