Rollup-OP-Stack：
官网地址：https://www.optimism.io/
文档地址：https://stack.optimism.io/
github地址：https://github.com/ethereum-optimism
浏览器地址：https://optimistic.etherscan.io/；
生态：https://www.optimism.io/apps/all
介绍：https://github.com/ethereum-optimism/optimism/blob/develop/specs/introduction.md
op-reth：https://github.com/paradigmxyz/reth/pull/4377；https://github.com/anton-rs/op-reth/tree/clabby/op-reth

什么是乐观主义？

Optimism是一个致力于扩展以太坊技术并扩展其协调世界各地人们建立有效的去中心化经济和治理系统的能力的项目。

Optimism Collective构建用于运行 L2 区块链的开源软件，旨在解决更广泛的加密货币生态系统中的关键治理和经济挑战。

乐观主义的运作原则是**影响=利润**，即对集体产生积极影响的个人应该获得按比例的利润奖励。**改变激励措施，你就改变了世界。**

在此存储库中，您会发现 OP Stack 的许多核心组件，OP Stack 是由 Optimism Collective 维护的去中心化软件堆栈，为 Optimism 提供动力并构成OP Mainnet和Base等区块链的支柱。

OP Stack 旨在“积极开源”，鼓励您根据需要探索、修改、扩展和测试代码。

虽然此处并未包含 OP 堆栈的所有元素，但可以在此存储库中找到其许多基本组件。

通过在自由、开放软件和共享标准方面进行合作，Optimism Collective 旨在防止孤立的软件开发并快速加速以太坊生态系统的发展。

一起贡献力量，建设未来，重新定义力量。

文档

- 如果您想在 OP 主网之上进行构建，请参阅Optimism 社区中心：https://community.optimism.io/
- 如果您想构建自己的基于 OP Stack 的区块链，请参阅OP Stack 文档：https://stack.optimism.io/
- 如果您想为 OP 堆栈做出贡献，请查看协议规范

Paradigm、OP Labs、Base 合作开发 OP Stack 高性能客户端 OP Reth

OP Reth 已合并，意味着其高性能以太坊节点 Reth 现在能在 OP Stack 中不做修改即可使用。

这是通过与 OP Labs、Base 团队和 Paradigm 自己的开源团队的合作实现的。

OP Reth 是一个实现了 Optimistic Rollup 设计的 OP Stack，该设计旨在尽可能接近以太坊。

核心模块：
1、op-batcher：L2-Batch Submitter，将批次捆绑提交到 L1；Ethereum-DA单独启动一个进程
Service for generating and submitting L2 tx batches to L1
code：https://github.com/ethereum-optimism/optimism/tree/129032f15b76b0d2a940443a39433de931a97a44/op-batcher/cmd

```
  Main()
{
    return CLIConfig{
    /* Required Flags */
    L1EthRpc:        ctx.String(flags.L1EthRpcFlag.Name),
    L2EthRpc:        ctx.String(flags.L2EthRpcFlag.Name),
    RollupRpc:       ctx.String(flags.RollupRpcFlag.Name),
    SubSafetyMargin: ctx.Uint64(flags.SubSafetyMarginFlag.Name),
    PollInterval:    ctx.Duration(flags.PollIntervalFlag.Name),


    /* Optional Flags */
    MaxPendingTransactions: ctx.Uint64(flags.MaxPendingTransactionsFlag.Name),
    MaxChannelDuration:     ctx.Uint64(flags.MaxChannelDurationFlag.Name),
    MaxL1TxSize:            ctx.Uint64(flags.MaxL1TxSizeBytesFlag.Name),
    Stopped:                ctx.Bool(flags.StoppedFlag.Name),
    TxMgrConfig:            txmgr.ReadCLIConfig(ctx),
    RPCConfig:              rpc.ReadCLIConfig(ctx),
    LogConfig:              oplog.ReadCLIConfig(ctx),
    MetricsConfig:          opmetrics.ReadCLIConfig(ctx),
    PprofConfig:            oppprof.ReadCLIConfig(ctx),
    CompressorConfig:       compressor.ReadCLIConfig(ctx),
}}
```

2、op-node：rollup共识层客户端，
2.1、Rollup-driver：主要负责从L1块（及其相关联的收据）派生出L2链。
驱动程序的任务是管理派生过程：跟踪L1区块头、跟踪L2链同步进度、当新的输入可用时，迭代派生步骤。

2.2、Derivation派生；此过程分为三个步骤：
2.2.1、从最新一个L2区块顶部的L1链中选择输入：区块列表，包含交易、相关数据和收据。
2.2.2、读取L1信息、存款和排序批，以便生成有效负载属性（本质上是一个没有输出属性的块）。
2.2.3、将有效载荷属性传递给执行引擎，以便可以计算L2块（包括输出块属性）。
虽然这个过程在概念上是从L1链到L2链的纯函数，但在实践中是增量的。
每当新的L1块被添加到L1链时，L2链被扩展。类似地，每当L1链重新组织时，L2链就会重新组织。
L2链衍生自L1链。特别地，每个L1块被映射到包括多个L2块的L2排序epoch。epoch编号被定义为等于对应的L1块编号。

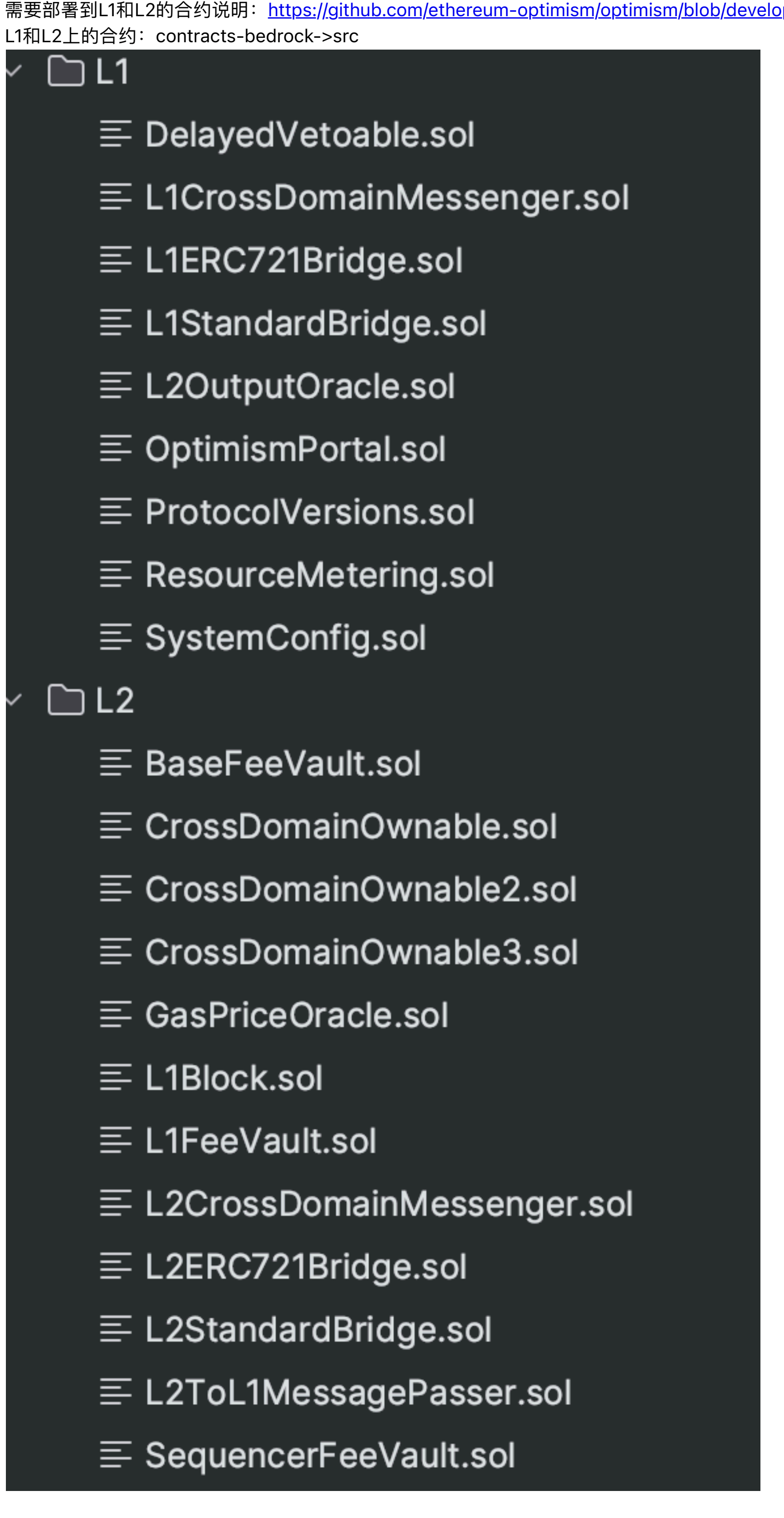
2.3 p2p模块：用户多个rollup-node之间同步数据
有可选的对等（P2P）网络服务，以在不依赖单个集中式端点的情况下，通过绕过L1来改善定序器视图与网络其余部分之间的延迟。
这也使得能够通过提供要同步的块头来引导更快的历史同步，并且与一次处理一个块的所有内容相比，只需要将L2链输入与L1数据进行比较。

3、op-proposer：L2-Output Submitter，向L1提交提案：Service for generating and submitting L2 Output checkpoints to the L2OutputOracle contract
specs介绍：https://github.com/ethereum-optimism/optimism/blob/develop/specs/proposals.md
4、cannon：欺诈证明的构建和验证：Although fault proof construction and verification is implemented in Cannon
5、op-program：欺诈证明验证程序，单独的任务，连接L1和L2 client：through the rollup state-transition to verify an L2 output from L1 inputs

目录结构

```
├── docs：包括审计和事后分析的文件集合
├── op-bindings：Bedrock 智能合约的 Go 绑定。
├── op-batcher : L2-Batch Submitter，将批次捆绑提交到 L1
├── op-bootnode : 独立操作节点发现bootnode
├── op-chain-ops : 国家手术实用程序
├── op-challenger : 争议游戏挑战代理
├── op-e2e：Go 中所有基岩组件的端到端测试
├── op-exporter : Prometheus 导出器客户端
├── op-heartbeat : 心跳监控服务
├── op-node : rollup共识层客户端
├── op-program : 防错程序，单独的任务：through the rollup state-transition to verify an L2 output from
L1 inputs
├── op-proposer : L2-Output Submitter，向L1提交提案：单独的任务通过RPC调用node-client
├── op-service : 通用代码库实用程序
├── op-signer : 客户端签名者
├── op-wheel : 数据库实用程序
├── ops-bedrock : Bedrock 开发网络工作
├── packages
│ ├── chain-mon : 链监控服务
│ ├── common-ts : 在 TypeScript 中构建应用程序的常用工具
│ ├── Contract-ts : ABI 和地址常量
│ ├── Contracts-bedrock : Bedrock 智能合约
│ ├── core-utils : 使构建 Optimism 更容易的低级实用程序
│ └── sdk : 提供了一套与 Optimism 交互的工具
├── proxyd : 可配置的RPC请求路由器和代理
└──规格：从基岩升级开始的汇总规格
```

需要部署到L1和L2的合约说明：https://github.com/ethereum-optimism/optimism/blob/develop/specs/predeploys.md
L1和L2上的合约：contracts-bedrock->src



基于op stack构建op rollup的文档：
https://stack.optimism.io/docs/build/getting-started/

optimism的L1相关合约：https://github.com/ethereum-optimism/optimism/tree/develop/packages/contracts-bedrock/deployments/mainnet

op在L1-ETH上的合约地址：

Contract name	Address
L1CrossDomainMessenger	0x25ace71c97B33Cc4729CF772ae268934F7ab5fA1(opens new window)
L1ERC721Bridge	0x5a7749f83b81B301cAb5f48EB8516B986DAef23D(opens new window)
L1StandardBridge	0x99C9fc46f92E8a1c0deC1b1747d010903E884bE1(opens new window)
L2OutputOracle	0xdfe97868233d1aa22e815a266982f2cf17685a27(opens new window)
OptimismMintableERC20Factory	0x75505a97BD334E7BD3C476893285569C4136Fa0F(opens new window)
OptimismPortal	0xbEb5Fc579115071764c7423A4f12eDde41f106Ed(opens new window)
PortalSender	0x0A893d9576b9cFD9EF78595963dc973238E78210(opens new window)
ProxyAdmin	0x543bA4AADBAb8f9025686Bd03993043599c6fB04(opens new window)
SystemConfig	0x229047fed2591dbec1eF1118d64F7aF3dB9EB290(opens new window)
SystemDictator	0xB4453CEb33d2e67FA244A24acf2E50CEF31F53cB(opens new window)
Batch Inbox Address	0xff00000000000000000000000000000000000010

L1部署的合约：

```
Op项目在L1的合约部署：部署合约的账户：0x354F3f4ECdcA5E0A7acE08d71348cdC1Dab48960
1、ProxyAdmin合约：
        code：packages/src/universal/
        L1-ETH:https://etherscan.io/address/0x543ba4aadbab8f9025686bd03993043599c6fb04#code

2、Proxy合约：合约实现：L2OutputOracle：0xd2e67b6a032f0a9b1f569e63ad6c38f7342c2e00
        code：packages/src/universal/
         L1-ETH：https://etherscan.io/address/0xdfe97868233d1aa22e815a266982f2cf17685a27#code

3、Proxy合约：合约实现：OptimismPortal：0x28a55488fef40005309e2da0040dbe9d300a64ab
        code：packages/src/universal/
         L1-ETH：https://etherscan.io/address/0xbeb5fc579115071764c7423a4f12edde41f106ed#code

4、Proxy合约：合约实现：OptimismMintableERC20Factory：0xae849efa4bcfc419593420e14707996936e365e2
        code：packages/src/universal/
         L1-ETH：https://etherscan.io/address/0x75505a97bd334e7bd3c476893285569c4136fa0f#code

5、Proxy代理合约：合约实现：SystemConfig：0x5efa852e92800d1c982711761e45c3fe39a2b6d8
        code：packages/src/universal/
         L1-ETH：https://etherscan.io/address/0x229047fed2591dbec1ef1118d64f7af3db9eb290#code

6、Proxy代理合约：合约实现：SystemDictator：0x09e040a72fd3492355c5aeedbc3154075f83488a
                负责协调完整Bedrock系统的部署,旨在支持新的网络部署和现有基岩前系统的升级。包含如下合约：
                    import { L2OutputOracle } from "../L1/L2OutputOracle.sol";
                    import { OptimismPortal } from "../L1/OptimismPortal.sol";
                    import { L1CrossDomainMessenger } from "../L1/L1CrossDomainMessenger.sol";
                    import { L1ERC721Bridge } from "../L1/L1ERC721Bridge.sol";
                    import { L1StandardBridge } from "../L1/L1StandardBridge.sol";
                    import { L1ChugSplashProxy } from "../legacy/L1ChugSplashProxy.sol";
                    import { AddressManager } from "../legacy/AddressManager.sol";
                    import { Proxy } from "../universal/Proxy.sol";
                    import { ProxyAdmin } from "../universal/ProxyAdmin.sol";
                    import { OptimismMintableERC20Factory } from
"../universal/OptimismMintableERC20Factory.sol";
                    import { PortalSender } from "./PortalSender.sol";
                    import { SystemConfig } from "../L1/SystemConfig.sol";
                    import { ResourceMetering } from "../L1/ResourceMetering.sol";
                    import { Constants } from "../libraries/Constants.sol";
        code：packages/src/universal/
         L1-ETH：https://etherscan.io/address/0xb4453ceb33d2e67fa244a24acf2e50cef31f53cb#code

7、L1CrossDomainMessenger合约：
        code：packages/contracts-bedrock/src/l1
        L1-ETH：https://etherscan.io/address/0x2150bc3c64cbfddbac9815ef615d6ab8671bfe43#code

8、L1StandardBridge合约：
        code：packages/contracts-bedrock/src/l1
        L1-ETH：https://etherscan.io/address/0xbfb731cd36d26c2a7287716de857e4380c73a64a#code

9、L2OutputOracle合约：使用代理合约
        code：packages/contracts-bedrock/src/l1
        L1-ETH：https://etherscan.io/address/0xd2e67b6a032f0a9b1f569e63ad6c38f7342c2e00#code

10、OptimismPortal合约：使用代理合约
        code：packages/contracts-bedrock/src/l1
        L1-ETH：https://etherscan.io/address/0x28a55488fef40005309e2da0040dbe9d300a64ab#code

11、OptimismMintableERC20Factory合约：使用代理合约
        code：packages/contracts-bedrock/src/l1
        L1-ETH：https://etherscan.io/address/0xae849efa4bcfc419593420e14707996936e365e2#code

12、PortalSender合约：是一个简单的中间合约，它将在基岩迁移期间将L1StandardBridge的余额转移到OptimismPortal
        code：packages/contracts-bedrock/src/l1
        L1-ETH：https://etherscan.io/address/0x0a893d9576b9cfd9ef78595963dc973238e78210#code

13、SystemConfig合约：使用代理合约
        code：packages/contracts-bedrock/src/l1
        L1-ETH：https://etherscan.io/address/0x5efa852e92800d1c982711761e45c3fe39a2b6d8#code

14、SystemDictator合约：使用代理合约
        code：packages/contracts-bedrock/src/l1
        L1-ETH：https://etherscan.io/address/0x09e040a72fd3492355c5aeedbc3154075f83488a#code

15、SystemDictator合约：使用代理合约
        code：packages/contracts-bedrock/src/l1
        L1-ETH：https://etherscan.io/address/0x09e040a72fd3492355c5aeedbc3154075f83488a#code

16、调用Proxy代理合约重新绑定实现合约SystemDictator
        L1-ETH：
https://etherscan.io/tx/0x9de8c0ce147021777b7818807bafcdeafeabb320c231105869babd2306ee77ac


17、L1ERC721Bridge合约：
        code：packages/contracts-bedrock/src/l1
        L1-ETH：https://etherscan.io/address/0x4afdd3a48e13b305e98d9eead67b1b5867e370df#code

18、调用SystemDictator的Proxy代理合约修改admin为多签地址：
0x9BA6e03D8B90dE867373Db8cF1A58d2F7F006b3A
        L1-ETH：
https://etherscan.io/tx/0x565fd4b31b7c0e98d91a4282d148b5920fe1f81de68c1bb30e863139a355908e

19、调用ProxyAdmin合约修改admin多钱地址：0x9BA6e03D8B90dE867373Db8cF1A58d2F7F006b3A
        L1-ETH：
https://etherscan.io/tx/0xc1816e47f3792ffad9bebac6e252c844ad10c5a089c4645a44b8c05659392108

20、ProtocolVersions合约：使用代理合约
        code：packages/contracts-bedrock/src/l1
        L1-ETH：https://etherscan.io/address/0x42f0bd8313ad456a38061308857b2383fe2c72a0#code

21、Proxy合约：合约实现：ProtocolVersions：0x42f0bd8313ad456a38061308857b2383fe2c72a0
        code：packages/contracts-bedrock/src/l1
        L1-ETH：https://etherscan.io/address/0xb0c67dd862aa24b7a5d716db06c2887e11937d7b#code

22、调用Proxy代理合约重新绑定实现合约ProtocolVersions
        L1-ETH：
https://etherscan.io/tx/0x3b55f5d09b4631b09c92c8e645e9cf243f9a585f06eb2192b7a76e46bdbadab4

23、调用ProtocolVersions的Proxy代理合约修改admin为ProxyAdmin代理合约账户：
0x543bA4AADBAb8f9025686Bd03993043599c6fB04
        L1-ETH：
https://etherscan.io/tx/0x9add200cbdc1a4e752ab03d59e307c753ecf0f592b6305f758c31e029a91d224


op-rollup提交batch数据到L1的接收账户地址：
https://etherscan.io/address/0xff00000000000000000000000000000000000010
```

optimism的L2合约：https://github.com/ethereum-optimism/optimism/tree/develop/packages/contracts-bedrock/deployments/optimism-mainnet

L2-op-mainnet.json：https://github.com/ethereum-optimism/optimism/blob/develop/packages/contracts-bedrock/deploy-config/mainnet.json

```
{
  "finalSystemOwner": "0x9BA6e03D8B90dE867373Db8cF1A58d2F7F006b3A",
  "portalGuardian": "0x9BA6e03D8B90dE867373Db8cF1A58d2F7F006b3A",
  "l1StartingBlockTag": "0x438335a20d98863a4c0c97999eb2481921ccd28553eac6f913af7c12aec04108",
  "l1ChainID": 1,
  "l2ChainID": 10,
  "l2BlockTime": 2,
  "l1BlockTime": 12,
  "maxSequencerDrift": 600,
  "sequencerWindowSize": 3600,
  "channelTimeout": 300,
  "p2pSequencerAddress": "0xAAAA45d9549EDA09E70937013520214382Ffc4A2",
  "batchInboxAddress": "0xff00000000000000000000000000000000000010",
  "batchSenderAddress": "0x6887246668a3b87F54DeB3b94Ba47a6f63F32985",
  "l2OutputOracleSubmissionInterval": 1800,
  "l2OutputOracleStartingTimestamp": 1686068903,
  "l2OutputOracleStartingBlockNumber": 105235063,
  "l2OutputOracleProposer": "0x473300df21D047806A082244b417f96b32f13A33",
  "l2OutputOracleChallenger": "0x9BA6e03D8B90dE867373Db8cF1A58d2F7F006b3A",
  "finalizationPeriodSeconds": 604800,
  "proxyAdminOwner": "0x7871d1187A97cbbE40710aC119AA3d412944e4Fe”,// 迁移后是
0x9BA6e03D8B90dE867373Db8cF1A58d2F7F006b3A
  "baseFeeVaultRecipient": "0xa3d596EAfaB6B13Ab18D40FaE1A962700C84ADEa",
  "l1FeeVaultRecipient": "0xa3d596EAfaB6B13Ab18D40FaE1A962700C84ADEa",
  "sequencerFeeVaultRecipient": "0xa3d596EAfaB6B13Ab18D40FaE1A962700C84ADEa",
  "baseFeeVaultMinimumWithdrawalAmount": "0x8ac7230489e80000",
  "l1FeeVaultMinimumWithdrawalAmount": "0x8ac7230489e80000",
  "sequencerFeeVaultMinimumWithdrawalAmount": "0x8ac7230489e80000",
  "baseFeeVaultWithdrawalNetwork": 0,
  "l1FeeVaultWithdrawalNetwork": 0,
  "sequencerFeeVaultWithdrawalNetwork": 0,
  "enableGovernance": true,
  "governanceTokenName": "Optimism",
  "governanceTokenSymbol": "OP",
  "governanceTokenOwner": "0x5C4e7Ba1E219E47948e6e3F55019A647bA501005",
  "l2GenesisBlockGasLimit": "0x1c9c380",
  "l2GenesisBlockBaseFeePerGas": "0x3b9aca00",
  "gasPriceOracleOverhead": 188,
  "gasPriceOracleScalar": 684000,
  "eip1559Denominator": 50,
  "eip1559Elasticity": 6,
  "l2GenesisRegolithTimeOffset": "0x0",
  "systemConfigStartBlock": 17422444,
  "requiredProtocolVersion": "0x0000000000000000000000000000000000000003000000010000000000000000",
  "recommendedProtocolVersion": "0x0000000000000000000000000000000000000003000000010000000000000000"
}
```

L2预部署合约地址：op-chain-ops/cmd/check-l2，在创始txn中先部署如下合约实现，然后绑定到对应的代理合约

```
const (
    L2ToL1MessagePasser          = "0x4200000000000000000000000000000000000016"
    DeployerWhitelist            = "0x4200000000000000000000000000000000000002"
    WETH9                        = "0x4200000000000000000000000000000000000006"
    L2CrossDomainMessenger       = "0x4200000000000000000000000000000000000007"
    L2StandardBridge             = "0x4200000000000000000000000000000000000010"
    SequencerFeeVault            = "0x4200000000000000000000000000000000000011"
    OptimismMintableERC20Factory = "0x4200000000000000000000000000000000000012"
    L1BlockNumber                = "0x4200000000000000000000000000000000000013"
    GasPriceOracle               = "0x420000000000000000000000000000000000000F"
    L1Block                      = "0x4200000000000000000000000000000000000015"
    GovernanceToken              = "0x4200000000000000000000000000000000000042"
    LegacyMessagePasser          = "0x4200000000000000000000000000000000000000"
    L2ERC721Bridge                = "0x4200000000000000000000000000000000000014"
    OptimismMintableERC721Factory = "0x4200000000000000000000000000000000000017"
    ProxyAdmin                   = "0x4200000000000000000000000000000000000018"
    BaseFeeVault                 = "0x4200000000000000000000000000000000000019"
    L1FeeVault                   = "0x420000000000000000000000000000000000001a"
    SchemaRegistry               = "0x4200000000000000000000000000000000000020"
    EAS                          = "0x4200000000000000000000000000000000000021"
)
```

op L2创建合约的账户：0x53A6eecC2dD4795Fcc68940ddc6B4d53Bd88Bd9E

```
1、L2ERC721Bridge合约：L2创始交易创建预部署合约，GENESISat txn GENESIS
        code：packages/contracts-bedrock/src/l1
        L2-OP:https://optimistic.etherscan.io/address/0xc0d3c0d3c0d3c0d3c0d3c0d3c0d3c0d3c0d30014#code

        1.1、创建Proxy代理合约：
        L2-OP  https://optimistic.etherscan.io/address/0x4200000000000000000000000000000000000014#code

        1.2 创建一个新的L2ERC721Bridge
        L2-op：https://optimistic.etherscan.io/address/0x5a7749f83b81b301cab5f48eb8516b986daef23d

        1.3、将L2ERC721Bridge合约的实现实例绑定到代理合约：0x4200000000000000000000000000000000000014
        L2-op：
https://optimistic.etherscan.io/tx/0x02985bade8b066202067f455d11aa0996a23c022fcd923e113af5d9997ddb5ce

        1.4、修改代理合约的admin为GnosisSafeProxy合约账户：0x2501c477D0A35545a387Aa4A3EEe4292A9a8B3F0

 https://optimistic.etherscan.io/tx/0xb910226e77e73720ab1a42fe21da681145673701c381295090a6b10199de8c3f

2、OptimismMintableERC20Factory合约：
        2.1、创世txn交易创建实现：
        https://optimistic.etherscan.io/address/0xc0d3c0d3c0d3c0d3c0d3c0d3c0d3c0d3c0d30012#code
        2.2、创建OptimismMintableERC721Factory合约的实现
        L2-OP:https://optimistic.etherscan.io/address/0x69d67c1caa8d0717dffa6d2e5b1f7f19926e5ef0#code
        2.3、创建代理合约Proxy
        L2-op：https://optimistic.etherscan.io/address/0x4482b6510df4c723bdf80c4441dbdbc855ab29ac#code
        2.4、将新创建的合约实现绑定到代理合约并修改代理合约的admin：
0x2501c477D0A35545a387Aa4A3EEe4292A9a8B3F0

https://optimistic.etherscan.io/tx/0x532158060340af1a339fcf47a6ce3cb0df26ba1a0f8ce33429ad6cfe8d6e057b

https://optimistic.etherscan.io/tx/0x1c920466a16d0ed7f78d5a57ad5894c715e11449459b897c3d54772ebc84d788

3、L2ToL1MessagePasser合约
        3.1、创世txn交易创建具体实现实例
        https://optimistic.etherscan.io/address/0xc0d3c0d3c0d3c0d3c0d3c0d3c0d3c0d3c0d30016#code
        3.2、创世txn创建代理合约Proxy：0x4200000000000000000000000000000000000016
        https://optimistic.etherscan.io/address/0x4200000000000000000000000000000000000016
```

OP堆栈中有两种类型的网络参与者：
Sequencers：序列器将用户的链上/链下事务合并为块。它们提交检查点输出以及批处理事务。
Verifiers：验证程序验证汇总的完整性，并对无效断言提出异议。
一：Sequencers序列器
定序器是主要的块生产者。可能存在一个或多个使用一致协议的测序仪。
对于1.0.0，只有一个测序仪（目前在乐观主义基金会的监督下运行）。
通常，规范可能会使用“测序仪”作为多个测序仪操作的共识协议的替代术语。
测序仪：
1、接受用户链外交易
2、观察链上交易（主要是来自L1的存款事件）
3、将这两种事务合并到具有特定顺序的L2块中。
4、通过将两件事作为调用数据提交给L1，将合并的L2块传播给L1：
1、在步骤1中接受的挂起的链外交易。
2、关于链上事务的排序的足够信息，以完全通过观察L1成功地重建步骤3.中的块。

定序器还最早在步骤3.提供对块数据的访问，以便用户可以在L1确认之前访问实时状态（如果他们选择的话）。

二：Verifiers验证程序
验证程序有两个目的：
1、向用户提供汇总数据；
2、以及验证汇总的完整性并对无效断言提出异议。

为了使网络保持安全，必须至少有一名诚实的验证器，能够验证汇总链的完整性并向用户提供区块链数据。

三：用户
网络的核心是用户（我们！）。用户可以：
1、通过向以太坊主网上的合同发送数据，在L2上存款或提取任意交易。
2、通过向序列发生器发送事务，在第2层使用EVM智能合约。
3、使用网络验证器提供的块资源管理器查看事务的状态。

Op：在ETH上做一个Cosmos：