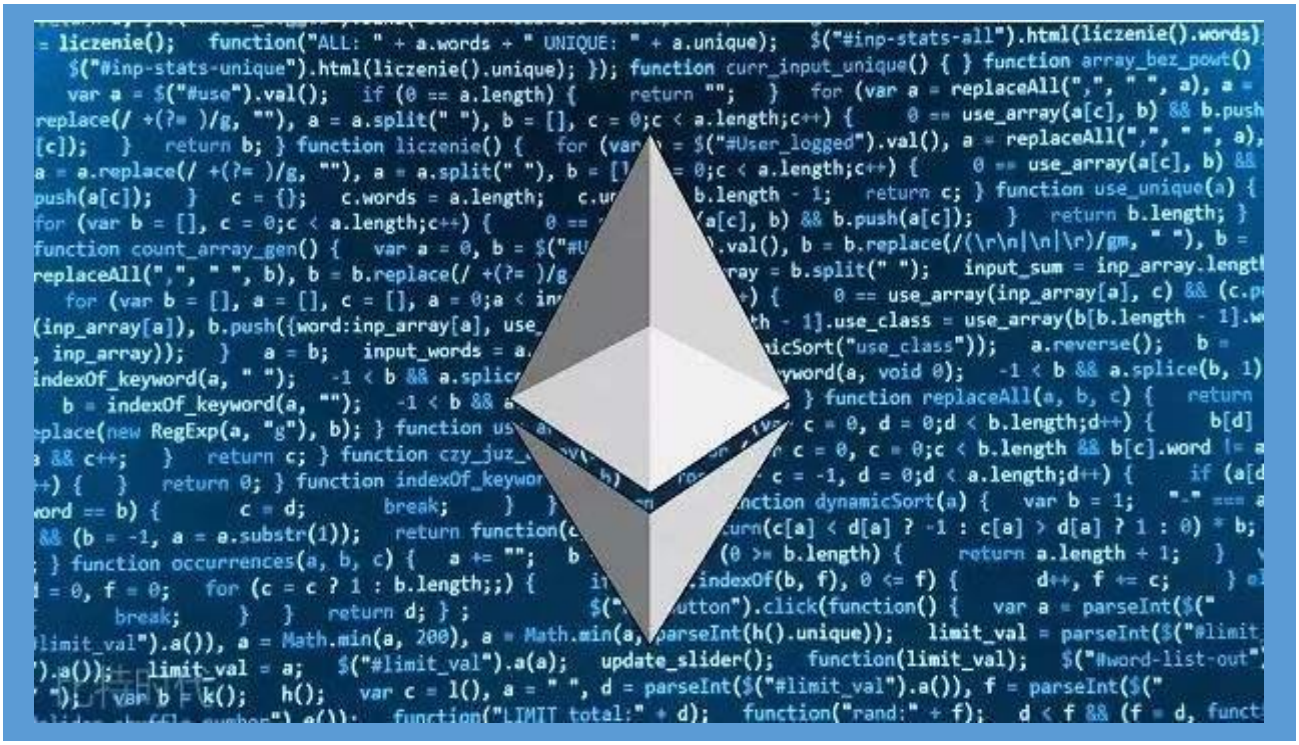
醍醐灌顶的酸爽

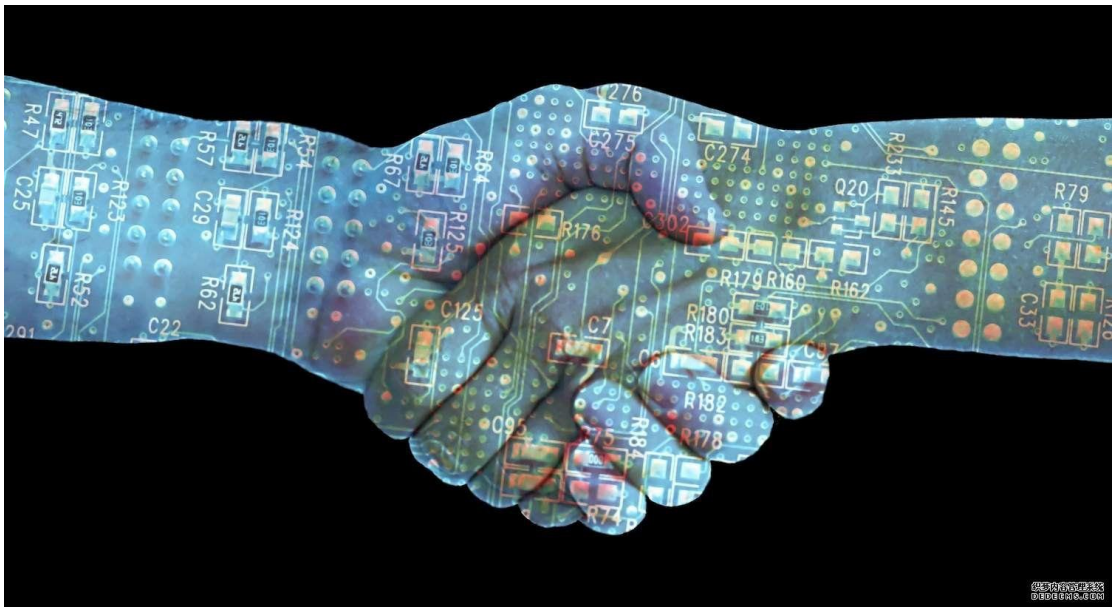
ETHEREUM 实践



目录

前言

总览

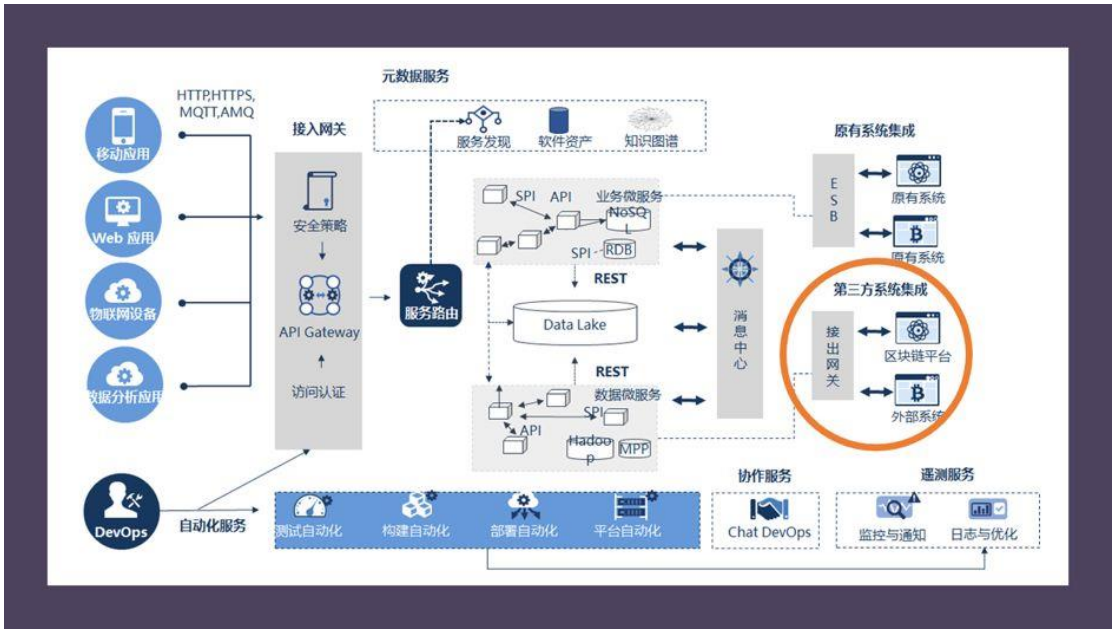


区块链本质上是一个去中心化的数据库，同时作为比特币的底层技术。区块链是一串使用密码学方法相关联产生的数据块，每一个数据块中包含了一次比特币网络交易的信息，用于验证其信息的有效性（防伪）和生成下一个区块。

如今的区块链技术概括起来是指通过去中心化和去信任的方式集体维护一个可靠数据库的技术。其实，区块链技术并不是一种单一的、全新的技术，而是多种现有技术（如加密算法、P2P 文件传输等）整合的结果，这些技术与数据库巧妙地组合在一起，形成了一种新的数据记录、传递、存储与呈现的方式。简单的说，区块链技术就是一种大家共同参与记录信息、存储信息的技术。过去，人们将数据记录、存储的工作交给中心化的机构来完成，而区块链技术则让系统中的每一个人都可以参与数据的记录、存储。区块链技术在没有中央控制点的分布式对等网络下，使用分布式集体运作的方法，构建了一个 P2P 的自组织网络。通过复杂的校验机制，区块链数据库能够保持完整性、连续性和一致性，即使部分参与人作假也无法改变区块链的完整性，更无法篡改区块链中的数据。区块链技术涉及的关键点包括：、去中心化、集体维护、时间戳、可靠数据库、去信任、非对称加密等。

以太坊开发入门，完整入门：
https://blog.csdn.net/huangshulang1234/article/details/79374085

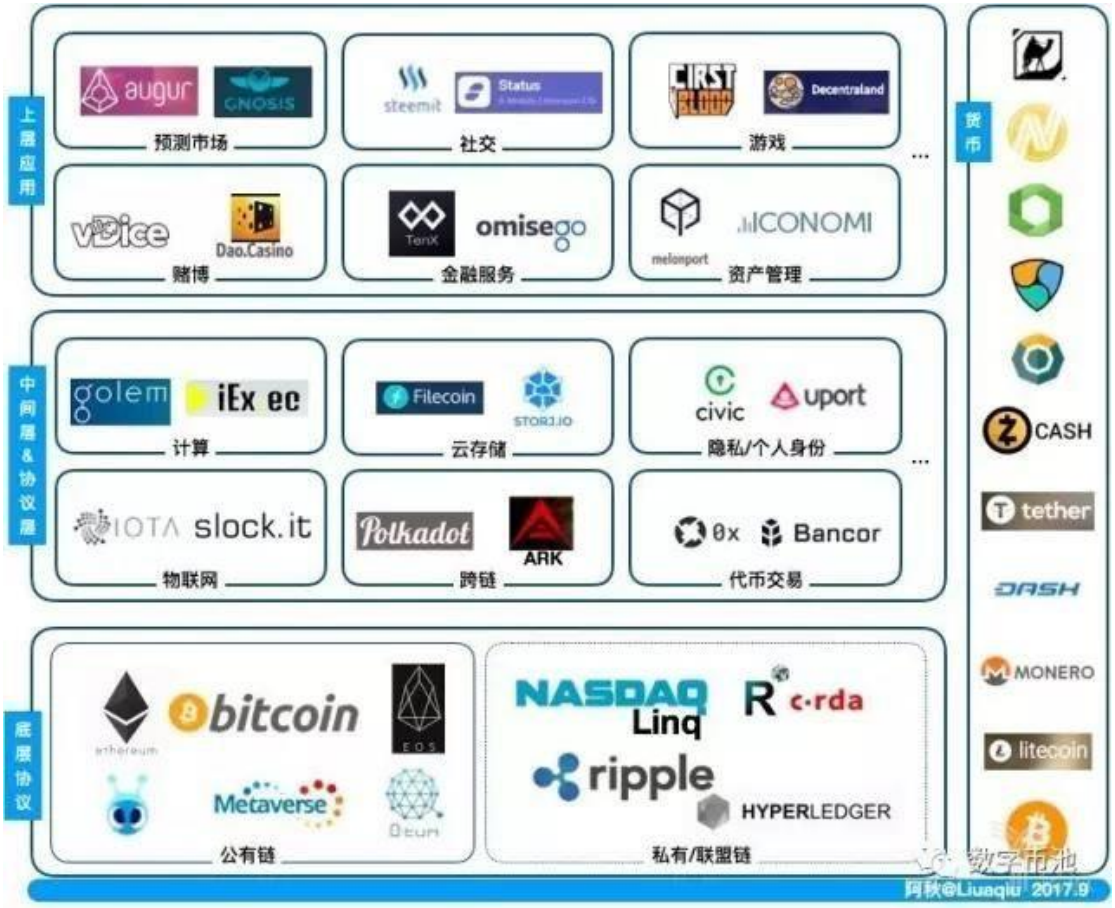
区块链的微服务化：



图说区块链
https://www.jianshu.com/p/1728c2c6fa7f

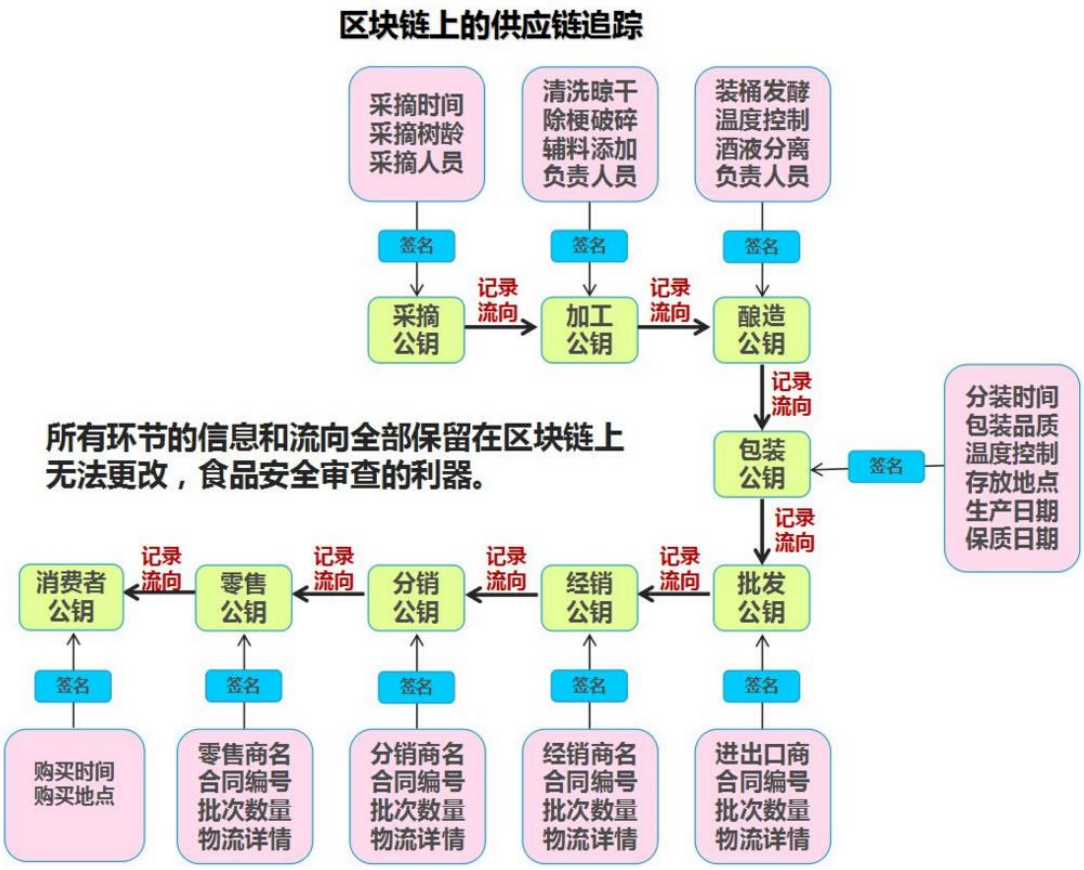
客户端软件
以太坊的两个主要的客户端软件是 Geth 和 Parity。

生态链



目前较知名的应用有：

- 去中心化创业投资：The DAO 用以太币资金创立，目标是为商企业和非营利机构创建新的去中心化营业模式、The Rudimental 让独立艺术家在区块链上进行群众募资。
- 社会经济平台：Backfeed。
- 去中心化预测市场：Augur。
- 物联网：Ethcore（一间以太坊公司）研发的客户端、Chronicled（一间区块链公司）发表了以太坊区块链的实物资产验证平台；芯片公司、物理 IP 创建者和生产者可以用植入的蓝牙或近场通信进行验证。Slock.It 开发的智能锁可以在付费后自动打开，让用户在付费后可以帮电动车充电、或是打开租屋的房门。
- 虚拟宝物交易平台：FreeMyVunk。
- 版权授权：Ujo Music 平台让创作人用智能合约发布音乐，消费者可以直接付费给创作人。伊莫珍・希普用此平台发布了一首单曲。
- 智能电网：TransActive Grid 让用户可以和邻居买卖能源。
- 去中心化期权市场：Etheropt。
- 钉住汇率的代币：DigixDAO 提供与黄金挂钩的代币，在 2016 年四月正式营运。Decentralized Capital 提供和各种货币挂钩的代币。
- 移动支付：Everex 让外劳汇款回家乡。



电力行业：
1,让每一度电都有迹可循,从根源上杜绝偷电漏电现象的发生。
2.与邻居交易剩余的电。

中国国有电力公司——国家电网公司正在探索区块链技术，以推进其“能源互联网”计划。去年 11 月，国家电网向中国国家知识产权局提交了一项名为“关于区块链的电力交易管控方法及装置”的专利申请。4 月 3 日，这项申请正式对外公布。国家电网这个能源巨头已经详细地在探索能源区块链系统，它声称可以存储和跟踪信息，例如消费者的电力消耗，以分散的方式共享数据。

政府身份证明。
目前新加坡 THEKEY 项目组已经实时联通了 66 个城市 2.1 亿人口政府的身份识别数据，BDMI（Blockchain Based Dynamic Multi-dimension Identification）是基于区块链的动态多维身份识别技术，BDMI 身份识别是用户实时身份数据、行为数据以及场景数据交叉验证的结果。生物特征：指纹、人脸、虹膜、血型、DNA 等

基本概念

分层

协议层

所谓的协议层，就是指代最底层的技术。这个层次通常是一个完整的区块链产品，类似于我们电脑的操作系统，它维护着网络节点，仅提供 Api 供调用。通常官方会提供简单的客户端（通称为钱包），这个客户端钱包功能也很简单，只能建立地址、验证签名、转账支付、查看余额等。这个层次是一切的基础，构建了网络环境、搭建了交易通道、制定了节点奖励规则，至于你要交易什么，想干什么，它一概不过问，也过问不了。典型的例子，自然是比特币，还有各种二代币，比如莱特币等，本书介绍的亿书币也是。这个层次，是现阶段开发者聚集的地方，这说明加密货币仍在起步当中。

从用到的技术来说，协议层主要包括网络编程、分布式算法、加密签名、数据存储技术等 4 个方面，其中网络编程能力是大家选择编程语言的主要考虑因素，因为分布式算法基本上属于业务逻辑上的实现，什么语言都可以做到，加密签名技术是直接简单的使用（请看书中相关的加密解密文章，不建议自由发挥，没有过多的编码逻辑），数据库技术也主要在使用层面，只有点对点网络的实现和并发处理才是开发的难点，所以对于那些网络编程能力强，对并发处理简单的语言，人们就特别偏爱。也因此，Nodejs 开发区块链应用，逐渐变得更加流行，Go 语言也在逐渐兴起。

我把这个层面进一步分成了存储层和网络层。数据存储可以相对独立，选择自由度大一些，可以单独来讨论。选择的原则无非是性能和易用性。我们知道，系统的整体性能，主要取决于网络或数据存储的 I/O 性能，网络 I/O 优化空间不大，但是本地数据存储的 I/O 是可以优化的。比如，比特币选择的是谷歌的 LevelDB，据说这个数据库读写性能很好，但是很多功能需要开发者自己实现。目前，困扰业界的一个重大问题是，加密货币交易处理量远不如现在中心化的支付系统（银行等），除了 I/O，需要全方位的突破。

分布式算法、加密签名等都要在实现点对点网络的过程中加以使用，所以自然是网络层的事情，也是编码的重点和难点，《Nodejs 开发加密货币》全书分享的基本上就是这部分的内容。当然，也有把点对点网络的实现单独分开的，把节点查找、数据传输和验证等逻辑独立出来，而把共识算法、加密签名、数据存储等操作放在一起组成核心层。无论怎么组合，这两个部分都是最核心、最底层的部分，都是协议层的内容。

扩展层

这个层面类似于电脑的驱动程序，是为了让区块链产品更加实用。目前有两类，一是各类交易市场，是法币兑换加密货币的重要渠道，实现简单，来钱快，成本低，但风险也大。二是针对某个方向的扩展实现，比如基于亿书侧链，可为第三方出版机构、论坛网站等内容生产商提供定制服务等。特别值得一提的就是大家听得最多的“智能合约”的概念，这是典型的扩展层面的应用开发。所谓“智能合约”就是“可编程合约”，或者叫做“合约智能化”，其中的“智能”是执行上的智能，也就是说达到某个条件，合约自动执行，比如自动转移证

券、自动付款等，目前还没有比较成型的产品，但不可否认，这将是区块链技术重要的发展方向。

扩展层使用的技术就没有什么限制了，可以包括很多，上面提到的分布式存储、机器学习、VR、物联网、大数据等等，都可以使用。编程语言的选择上，可以更加自由，因为可以与协议层完全分离，编程语言也可以与协议层使用的开发语言不相同。在开发上，除了在交易时与协议层进行交互之外，其他时候尽量不要与协议层的开发混在一起。这个层面与应用层更加接近，也可以理解为 B/S 架构的产品中的服务端（Server）。这样不仅在架构设计上更加科学，让区块链数据更小，网络更独立，同时也可以保证扩展层开发不受约束。

从这个层面来看，区块链可以架构开发任何类型的产品，不仅仅是用在金融行业。在未来，随着底层协议的更加完善，任何需要第三方支付的产品都可以方便的使用区块链技术；任何需要确权、征信和追溯的信息，都可以借助区块链来实现。我个人觉得，这个目标应该很快就能实现。

应用层

这个层面类似于电脑中的各种软件程序，是普通人可以真正直接使用的产品，也可以理解为 B/S 架构的产品中的浏览器端（Browser）。这个层面的应用，目前几乎是空白。市场亟待出现这样的应用，引爆市场，形成真正的扩张之势，让区块链技术快速走进寻常百姓，服务于大众。大家使用的各类轻钱包（客户端），应该算作应用层最简单、最典型的应用。很快，亿书将基于亿书网络推出文档协作工具，这个就是典型的应用层的产品。

加密算法

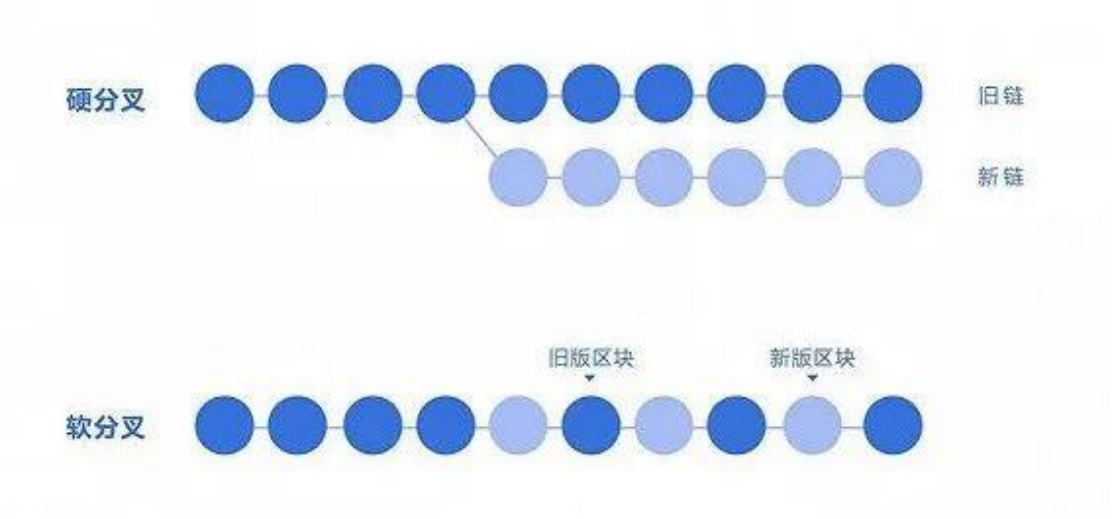
共识机制

- PoW
- PoS
- DpoS

软分叉与硬分叉

比特币每次代码升级都要获得比特币社区的一致认可，如果比特币社区无法达成一致，区块链很可能形成分叉。简单来讲，分叉是指区块链在升级时发生了意见分歧，从而导致区块链分叉。根据分叉后的区块链是否兼容旧区块，将分叉分为硬分叉和软分叉。

软分叉和硬分叉都"向后兼容"，这样才能保证新节点可以从头验证区块链。向后兼容是指新软件接受由旧软件所产生的数据或者代码



软分叉：是指区块链网络系统版本或协议升级后，旧的节点并不会意识到比特币代码发生改变，并继续接受由新节点创造的区块，新老节点始终还是在同一条链上工作。

硬分叉：是指比特币区块格式或交易格式（共识机制）发生改变时，未升级的节点拒绝验证已经升级的节点产生的区块，然后大家各自延续自己认为正确的链，所以分成两条链。

由比特大陆主导的比特币（BTC）硬分叉比特币现金（BCH），是目前为止最成功的一次硬分叉。自 2017 年 8 月 1 日硬分叉之后，BCH 已稳踞全球加密货币第四名的位置。

最具代表性的例子要属以太坊。2016 年以太坊一个知名项目 The DAO 被黑客攻击，损失了约 6000 万美元的 ETH，以太坊团队通过硬分叉的方式“追回”了被盗资产，自此分裂出两条链，原链（ETC）和新的分叉链（ETH）。

DAPP

开发概念

Solidity

Solidity 是用于编写在以太坊区块链上运行的智能合约的最流行的编程语言。 它是一种高级语言，当编译转换为 EVM 字节码。这与 Java 非常相似，其中有诸如 Scala，Groovy，Clojure，JRuby 等 JVM 语言。所有这些编译都生成在 JVM（Java 虚拟机）中运行的字节码。

Remix

以太坊官方推荐的智能合约开发 IDE，可以在浏览器中快速部署测试智能合约

Truffle

Truffle 和 Embark 是用于开发以太坊 DApps 的两个最常用的框架。 它们抽象出在区块链上编译和部署合同的许多复杂的东西。
Truffle（基于 Javascript）：
https://truffleframework.com/
http://truffle.tryblockchain.org/
功能：

- 首先对客户端做了深度集成。开发，测试，部署一行命令都可以搞定。不用再记那么多环境地址，繁重的配置更改，及记住诸多的命令。
- 它提供了一套类似 maven 或 gradle 这样的项目构建机制，能自动生成相关目录，默认是基于 Web 的。当前这个打包机制是自定义的，比较简陋，不与当前流行打包方案兼容。但自己称会弃用，与主流兼容，好在它也支持自定义打包流程。
- 提供了合约抽象接口，可以直接通过 var meta = MetaCoin.deployed();拿到合约对象后，在 Javascript 中直接操作对应的合约函数。原理是使用了基于 web3.js 封装的 Ether Pudding 工具包。简化开发流程。
- 提供了控制台，使用框架构建后，可以直接在命令行调用输出结果，可极大方便开发调试。
- 提供了监控合约，配置变化的自动发布，部署流程。不用每个修改后都重走整个流程。

Embark

- Blockchain (Ethereum)
- 去中心化 Storage (IPFS)
- 去中心化交流 (Whisper, Orbit)
- Web 技术

Meteor

许多 DApp 开发者使用的另一套开发栈由 web3.js 和 Meteor 组成，Meteor 是一套通用 webapp 开发框架（ethereum-meteor-wallet 项目提供了一个很棒的入门实例，而 SilentCiero 正在构建大量 Meteor 与 web3.js 和 DApp 集成的模板）。我下载并运行过一些不错的 DApp 是以这种方式构造的。

APIs

BlockApps.net 打算提供一套 RESTful API 给 DApp 使用以免去开发者运行本地节点的麻烦，这个中心化服务是基于以太坊 Haskell 实现的。这与 DApp 的去中心化模型背道而驰，但是在本地无法运行以太坊节点的场合非常有用，比如在你希望只有浏览器或者使用移动设备的用户也能使用你的 DApp 的时候。BlockApps 提供了一个命令行工具 bloc，注册一个开发者帐号之后就可以使用。

区块链间服务

Oraclize

Oraclize 是一种服务，旨在使智能合约可以访问来自其他区块链或者万维网的数据。该服务目前在比特币以及以太坊测试网和主网上可用。Oraclize 的特殊之处是你不需要信任它，因为它可以为所提供给智能合约的全部数据做真实性证明。

Reality Keys

RealityKeys 是关于事实的预测的加密证明，它的服务是提供自动化和人工验证的数据，将有希望提供新一代的自动化、无条件信任的信息服务。Reality Keys 监测一系列的数据来源包括汇率、加密货币交易、个人训练目标（可以在 Reality Keys 网站建立）或在维基的数百万主题，所有这些都是基于公共提供的 API 接口。除了公开的 API 外，任何人都可以往 Reality Keys 体检事实和检查存在事实的状态。Reality Keys 的这个服务是完全免费的，只是另外提供人工二次确认的收费服务，来避免数据的自动化检查之外的疏漏或变更。

环境搭建

编译安装

Ubuntu 16.04 Desktop

EOS 项目地址：
https://github.com/EOSIO/eos

EOSIO currently supports the following operating systems:

- Amazon 2017.09 and higher
- Centos 7
- Fedora 25 and higher (Fedora 27 recommended)
- Mint 18

- Ubuntu 16.04 (Ubuntu 16.10 recommended)
- Ubuntu 18.04
- MacOS Darwin 10.12 and higher (MacOS 10.13.x recommended)

如果需要编译安装，请参考：
https://www.jianshu.com/p/a200a28deae4

在 X windows 下按下[Ctrl]+[Alt]+[Backspace]可以重启 X。
切换 X windows 与命令行模式：
[Ctrl]+[Alt]+[F1]~[F6]:命令行登录 tty1~tty6 终端。
[Ctrl]+[Alt]+[F7]：返回图形界面
[root@server ~]#startx 启动 X windows
在/etc/inittab 修改运行等级为 5 是系统开机进入 X windows，运行等级为 3 是系统开机进入文本。

重试 3 次，8 小时后检出网络：
david@david-virtual-machine:~$ git clone https://github.com/EOSIO/eos --recursive
正克隆到 'eos'...
remote: Counting objects: 107700, done.
remote: Compressing objects: 100% (86/86), done.
remote: Total 107700 (delta 53), reused 63 (delta 28), pack-reused 107582
接收对象中: 100% (107700/107700), 105.33 MiB | 132.00 KiB/s, 完成.
处理 delta 中: 100% (85639/85639), 完成.
子模组 'contracts/libc++/upstream'（https://github.com/EOSIO/libcxx.git）未对路径 'contracts/libc++/upstream' 注册
子模组 'contracts/musl/upstream'（https://github.com/EOSIO/musl.git）未对路径 'contracts/musl/upstream' 注册
子模组 'externals/binaryen'（https://github.com/EOSIO/binaryen）未对路径 'externals/binaryen' 注册
子模组 'externals/magic_get'（https://github.com/EOSIO/magic_get）未对路径 'externals/magic_get' 注册
子模组 'libraries/appbase'（https://github.com/eosio/appbase）未对路径 'libraries/appbase' 注册
子模组 'libraries/chainbase'（https://github.com/eosio/chainbase）未对路径 'libraries/chainbase' 注册
子模组 'libraries/fc'（https://github.com/EOSIO/fc）未对路径 'libraries/fc' 注册
子模组 'libraries/softfloat'（https://github.com/eosio/berkeley-softfloat-3）未对路径 'libraries/softfloat' 注册
正克隆到 '/home/david/eos/contracts/libc++/upstream'...
remote: Counting objects: 73633, done.
remote: Total 73633 (delta 0), reused 0 (delta 0), pack-reused 73633
接收对象中: 100% (73633/73633), 22.75 MiB | 301.00 KiB/s, 完成.
处理 delta 中: 100% (51712/51712), 完成.
正克隆到 '/home/david/eos/contracts/musl/upstream'...

remote: Counting objects: 27733, done.
remote: Compressing objects: 100% (8/8), done.
remote: Total 27733 (delta 2), reused 2 (delta 1), pack-reused 27724
接收对象中: 100% (27733/27733), 4.35 MiB | 79.00 KiB/s, 完成.
处理 delta 中: 100% (19754/19754), 完成.
正克隆到 '/home/david/eos/externals/binaryen'...
remote: Counting objects: 51039, done.
remote: Compressing objects: 100% (8/8), done.
remote: Total 51039 (delta 3), reused 3 (delta 2), pack-reused 51029
接收对象中: 100% (51039/51039), 73.73 MiB | 50.00 KiB/s, 完成.
处理 delta 中: 100% (43890/43890), 完成.
正克隆到 '/home/david/eos/externals/magic_get'...
remote: Counting objects: 2499, done.
remote: Total 2499 (delta 0), reused 0 (delta 0), pack-reused 2499
接收对象中: 100% (2499/2499), 522.91 KiB | 41.00 KiB/s, 完成.
处理 delta 中: 100% (1776/1776), 完成.
正克隆到 '/home/david/eos/libraries/appbase'...
remote: Counting objects: 237, done.
remote: Compressing objects: 100% (30/30), done.
remote: Total 237 (delta 31), reused 51 (delta 24), pack-reused 176
接收对象中: 100% (237/237), 64.42 KiB | 254.00 KiB/s, 完成.
处理 delta 中: 100% (108/108), 完成.
正克隆到 '/home/david/eos/libraries/chainbase'...
remote: Counting objects: 322, done.
remote: Compressing objects: 100% (6/6), done.
remote: Total 322 (delta 0), reused 1 (delta 0), pack-reused 315
接收对象中: 100% (322/322), 67.77 KiB | 8.00 KiB/s, 完成.
处理 delta 中: 100% (129/129), 完成.
正克隆到 '/home/david/eos/libraries/fc'...
remote: Counting objects: 1647, done.
remote: Compressing objects: 100% (675/675), done.
remote: Total 1647 (delta 853), reused 1640 (delta 848), pack-reused 0
接收对象中: 100% (1647/1647), 909.14 KiB | 38.00 KiB/s, 完成.
处理 delta 中: 100% (853/853), 完成.
正克隆到 '/home/david/eos/libraries/softfloat'...
remote: Counting objects: 2579, done.
remote: Total 2579 (delta 0), reused 0 (delta 0), pack-reused 2579
接收对象中: 100% (2579/2579), 750.76 KiB | 492.00 KiB/s, 完成.
处理 delta 中: 100% (2361/2361), 完成.
子模组路径 'contracts/libc++/upstream'：检出 '2880ac42909d4bb29687ed079f8bb4405c3b0869'
子模组路径 'contracts/musl/upstream'：检出 '8a34536ac9764c90c86cc0b62d0cda07449fd5d8'
子模组路径 'externals/binaryen'：检出 'e13b8b74d210b882f19252dc11dc3df73294ae13'
子模组 'test/emscripten'（https://github.com/kripken/emscripten.git）未对路径

'externals/binaryen/test/emscripten' 注册
子模组 'test/spec' （https://github.com/WebAssembly/testsuite.git ）未对路径 'externals/binaryen/test/spec' 注册
子模组 'test/waterfall' （https://github.com/WebAssembly/waterfall.git ）未对路径 'externals/binaryen/test/waterfall' 注册
正克隆到 '/home/david/eos/externals/binaryen/test/emscripten'...
remote: Counting objects: 109072, done.
remote: Compressing objects: 100% (183/183), done.
remote: Total 109072 (delta 76), reused 82 (delta 49), pack-reused 108840
接收对象中: 100% (109072/109072), 158.34 MiB | 32.00 KiB/s, 完成.
处理 delta 中: 100% (75611/75611), 完成.
正克隆到 '/home/david/eos/externals/binaryen/test/spec'...
remote: Counting objects: 989, done.
remote: Compressing objects: 100% (19/19), done.
remote: Total 989 (delta 5), reused 4 (delta 0), pack-reused 970
接收对象中: 100% (989/989), 626.38 KiB | 31.00 KiB/s, 完成.
处理 delta 中: 100% (651/651), 完成.
正克隆到 '/home/david/eos/externals/binaryen/test/waterfall'...
remote: Counting objects: 2642, done.
remote: Compressing objects: 100% (44/44), done.
remote: Total 2642 (delta 42), reused 39 (delta 24), pack-reused 2574
接收对象中: 100% (2642/2642), 681.13 KiB | 196.00 KiB/s, 完成.
处理 delta 中: 100% (1921/1921), 完成.
子模组路径 'externals/binaryen/test/emscripten'：检出 '1d979a75722a513b586e6705d662ff4ee0ea832b'
子模组路径 'externals/binaryen/test/spec'：检出 '668281b3d7dfe9be6cbc3c4500b537c49d6449b9'
子模组路径 'externals/binaryen/test/waterfall'：检出 '900646fc880d526160b0df9b78bc9dd4f02dc1ab'
子模组路径 'externals/magic_get'：检出 '89fda1da702e6c76a22bfb6233e9e3d0641708ec'
子模组路径 'libraries/appbase'：检出 '44d7d88e561cc9708bcac9453686c8d0f0ffa41e'
子模组路径 'libraries/chainbase'：检出 '59d59b7089f48be93704f10340e1b5b3615a8a85'
子模组路径 'libraries/fc'：检出 'f050d0cd0f81ff2e176548ffd34917a3b1f1f87b'
子模组路径 'libraries/softfloat'：检出 '9dff375a6e3a9728a024b30afa127c88fcb40ea2'

build 设置：
cd eos
./eosio_build.sh

david@david-virtual-machine:~/eos$ ./eosio_build.sh

Beginning build version: 1.2
2018 年 06 月 26 日 星期二 07:18:43 UTC

User: david
git head id: 0b046bfa26478105247d35b439a449831bad42da
Current branch: master

ARCHITECTURE: Linux

OS name: Ubuntu
OS Version: 17.10
CPU speed: 2600.000Mhz
CPU cores: 4
Physical Memory: 3945 Mgb
Disk install: /dev/sda1
Disk space total: 35G
Disk space available: 18G
Your system must have 7 or more Gigabytes of physical memory installed.
Exiting now.

You must have at least 20GB of available storage to install EOSIO.
Exiting now.

david@david-vm-ether:~/eos$ ./eosio_build.sh

Beginning build version: 1.2
2018 年 06 月 26 日 星期二 08:24:53 UTC
User: david
git head id: 0b046bfa26478105247d35b439a449831bad42da
Current branch: master

ARCHITECTURE: Linux

OS name: Ubuntu
OS Version: 17.10
CPU speed: 2600.000Mhz
CPU cores: 4
Physical Memory: 16044 Mgb
Disk install: /dev/sda1
Disk space total: 117G
Disk space available: 103G

Checking for installed dependencies.

Package clang-4.0 NOT found.
Package lldb-4.0 NOT found.
Package libclang-4.0-dev NOT found.

Package cmake NOT found.
Package make NOT found.
Package automake NOT found.
Package libbz2-dev NOT found.
Package libssl-dev NOT found.
Package libgmp3-dev NOT found.
Package autotools-dev NOT found.
Package build-essential NOT found.
Package libicu-dev NOT found.
Package python2.7-dev NOT found.
Package python3-dev NOT found.
Package autoconf NOT found.
Package libtool NOT found.
Package curl NOT found.
Package zlib1g-dev NOT found.
Package doxygen NOT found.
Package graphviz NOT found.

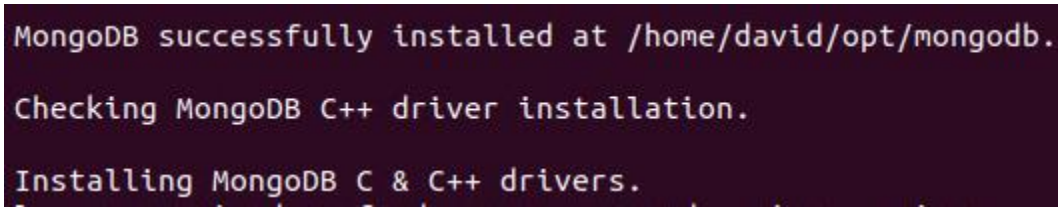
The following dependencies are required to install EOSIO.

1. clang-4.0
2. lldb-4.0
3. libclang-4.0-dev
4. cmake
5. make
6. automake
7. libbz2-dev
8. libssl-dev
9. libgmp3-dev
10. autotools-dev
11. build-essential
12. libicu-dev
13. python2.7-dev
14. python3-dev
15. autoconf
16. libtool
17. curl
18. zlib1g-dev
19. doxygen
20. graphviz

Do you wish to install these packages?

1) Yes

2) No



[100%] Built target unit_test

```
 _______    _______    _______ _________ _______
(   ____ \(    ___   )(   ____ \\__   __/(   ___   )
| (       \/| (     ) || (       \/     ) (     | (     ) |
| (__       | |     | || (_____        | |     | |     | |
|   __)      | |     | |(_____   )      | |     | |     | |
| (          | |     | |      ) |      | |     | |     | |
| (____/\| (___) |/\____) |___) (___| (___) |
(_______/(_______)\_______)\_______/(_______)
```

EOSIO has been successfully built. 01:00:33

To verify your installation run the following commands:

export PATH=${HOME}/opt/mongodb/bin:$PATH
/home/david/opt/mongodb/bin/mongod -f /home/david/opt/mongodb/mongod.conf &
cd /home/david/eos/build; make test

For more information:
EOSIO website: https://eos.io
EOSIO Telegram channel @ https://t.me/EOSProject
EOSIO resources: https://eos.io/resources/
EOSIO Stack Exchange: https://eosio.stackexchange.com
EOSIO wiki: https://github.com/EOSIO/eos/wiki

从早上八点开始，到第二天凌晨 2:06 结束。

david@david-vm-ether:~$ cat .bashrc
export PATH=${HOME}/opt/mongodb/bin:$PATH
david@david-vm-ether:~$ source .bashrc

david@david-vm-ether:~$ /home/david/opt/mongodb/bin/mongod -f /home/david/opt/mongodb/mongod.conf &
[1] 18614
david@david-vm-ether:~$ netstat -tnlp

（并非所有进程都能被检测到，所有非本用户的进程信息将不会显示，如果想看到所有信息，则必须切换到 root 用户）
激活 Internet 连接 (仅服务器)

```
Proto  Recv-Q  Send-Q  Local  Address                Foreign  Address                State
PID/Program name
tcp        0      0 0.0.0.0:22              0.0.0.0:*               LISTEN      -
tcp        0      0 127.0.0.1:631           0.0.0.0:*               LISTEN      -
tcp        0      0  127.0.0.1:27017        0.0.0.0:*               LISTEN
18614/mongod
tcp        0      0 0.0.0.0:5355            0.0.0.0:*               LISTEN      -
tcp6       0      0 :::22                   :::*                    LISTEN      -
tcp6       0      0 ::1:631                 :::*                    LISTEN      -
tcp6       0      0  ::1:27017              :::*                    LISTEN
18614/mongod
tcp6       0      0 :::5355                 :::*                    LISTEN      -
```

Make 测试：

```
# 启动 mongoDB
/usr/local/bin/mongod -f /usr/local/etc/mongod.conf &
# 运行测试
cd build
make test
```

```
david@david-vm-ether:~$ cd /home/david/eos/build; make test
Running tests...
Test project /home/david/eos/build
        Start    1: test_cypher_suites
  1/36 Test    #1: test_cypher_suites .....................   Passed       0.02 sec
        Start    2: validate_simple.token_abi
……
26/36 Test #26: unit_test_binaryen .....................   Passed    139.57 sec
        Start 27: unit_test_wavm
27/36 Test #27: unit_test_wavm .........................   Passed    584.27 sec
        Start 28: validate_deferred_test_abi
28/36 Test #28: validate_deferred_test_abi .............   Passed       0.04 sec
        Start 29: plugin_test
29/36 Test #29: plugin_test ............................   Passed       0.04 sec
        Start 30: nodeos_run_test
30/36 Test #30: nodeos_run_test ........................   Passed      58.08 sec
        Start 31: p2p_dawn515_test
31/36 Test #31: p2p_dawn515_test .......................   Passed       8.07 sec
        Start 32: distributed-transactions-test
32/36 Test #32: distributed-transactions-test ..........   Passed      64.46 sec
        Start 33: restart-scenarios-test-resync
```

33/36 Test #33: restart-scenarios-test-resync ..... Passed 134.30 sec
Start 34: restart-scenarios-test-hard_replay
34/36 Test #34: restart-scenarios-test-hard_replay ... Passed 127.91 sec
Start 35: restart-scenarios-test-none
35/36 Test #35: restart-scenarios-test-none ..... Passed 127.85 sec
Start 36: validate_dirty_db_test
36/36 Test #36: validate_dirty_db_test ..... Passed 2.39 sec

100% tests passed, 0 tests failed out of 36

Total Test time (real) = 1247.89 sec
【21 分钟】

安装可执行文件：
生成全局命令，写入/usr/local
cd build
sudo make install

david@david-vm-ether:~$ sudo make install
……
-- Installing: /usr/local/bin/nodeos
-- Installing: /usr/local/var/log/eosio
-- Installing: /usr/local/var/lib/eosio
-- Installing: /usr/local/bin/cleos
-- Installing: /usr/local/bin/keosd
-- Installing: /usr/local/bin/eosio-launcher
-- Installing: /usr/local/bin/eosio-abigen
-- Installing: /usr/local/bin/eosiocpp

启动单个节点的本地 Testnet：
至此 EOS 安装就已经完成了。现在来启动一个本地节点测试下。
nodeos -e -p eosio --plugin eosio::wallet_api_plugin --plugin eosio::chain_api_plugin --plugin eosio::account_history_api_plugin

account_history_api_plugin 开关不在起作用：
local/bin$ nodeos -e -p eosio --plugin eosio::chain_api_pluginnetstat –tnlp

nodeos 配置文件路径 ~/Library/Application\ Support/eosio/nodeos/config/，包括两个文件：
david@david-vm-ether:/$ which nodeos
/usr/local/bin/nodeos

启动报错：
version > 0: Block log was not setup properly with genesis information.
清除缓存：

david@david-vm-ether:~$ rm -rf .local/share/eosio/nodeos/data

启动报错：
Throw in function appbase::abstract_plugin &appbase
原因：MongDB 没启动。
david@david-vm-ether:~$ /home/david/opt/mongodb/bin/mongod -f /home/david/opt/mongodb/mongod.conf &

报错信息：
Failed to start a pending block, will try again later
处理：
rm -rf ~/.local/share/eosio/nodeos/data/shared_mem

david@david-vm-ether:/$ sudo find | grep config.ini
./home/david/.local/share/eosio/nodeos/config/config.ini

- config.ini 启动配置文件
- genesis.json 创世文件

如果启动时不存在配置文件，nodeos 会生成默认配置文件。生成配置文件后可以 Ctrl-C 停止 nodeos 的运行。接下来我们对配置文件做一些修改。

修改配置文件

nodeos 的配置既可以像上面一样以命令行参数的方式加载，也可以通过配置文件加载。我们将刚刚命令行中的参数，通过修改 config.ini 文件来加载

```
# Load the testnet genesis state, which creates some initial block producers with the default key
genesis-json = "genesis.json"
# Enable production on a stale chain, since a single-node test chain is pretty much always stale
enable-stale-production = true
# Enable block production with the testnet producers
producer-name = eosio
# Load the block producer plugin, so you can produce blocks
plugin = eosio::producer_plugin
# Wallet plugin
plugin = eosio::wallet_api_plugin
# As well as API and HTTP plugins
plugin = eosio::chain_api_plugin
plugin = eosio::http_plugin
```

启动命令：
nodeos
nodeos 运行时会生成两类数据文件，共享内存文件和日志文件。
文件目录：~/Library/Application\ Support/eosio/nodeos/data

```
david@david-vm-ether:~$ cleos get info
{
   "server_version": "0b046bfa",
   "chain_id": "cf057bbfb72640471fd910bcb67639c22df9f92470936cddc1ade0e2f2e7dc4f",
   "head_block_num": 2859,
   "last_irreversible_block_num": 2858,
   "last_irreversible_block_id":
"00000b2a4934f8514d7b70659b746832082ba574a83d05df760e6180dc93722b",
   "head_block_id":
"00000b2b246bc6e49abbfd0db42f99eede14b744c358ac2d2ac5f0aa09bc5e94",
   "head_block_time": "2018-06-27T03:10:21.500",
   "head_block_producer": "eosio",
   "virtual_block_cpu_limit": 3482046,
   "virtual_block_net_limit": 18291155,
   "block_cpu_limit": 199900,
   "block_net_limit": 1048576
}
```

Docker 部署

CentOS 7.4

```
[root@Docker-GP ~]# docker pull eosio/eos
Using default tag: latest
Trying to pull repository docker.io/eosio/eos ...
latest: Pulling from docker.io/eosio/eos
6b98dfc16071: Pull complete
4001a1209541: Pull complete
6319fc68c576: Pull complete
b24603670dc3: Pull complete
97f170c87c6f: Pull complete
ca8277dae3e4: Pull complete
d9de8ba6e50e: Pull complete
ae90d5bffb3f: Pull complete
1e202fb82cd4: Pull complete
6cf6989520ae: Pull complete
4e01f065c15a: Pull complete
f6882e7018de: Pull complete
Digest: sha256:9b005110e16c24fb9631046cf6b6ab192b94ea3fad922691d741c3638947265f
Status: Downloaded newer image for docker.io/eosio/eos:latest

[root@Docker-GP ~]# docker images eosio/eos
```

REPOSITORY TAG IMAGE ID CREATED SIZE
docker.io/eosio/eos latest 9a79d67130ba 2 days ago 246 MB

启动容器，因主机 8888 端口已经占用，改为 8889 端口。
[root@Docker-GP /]# docker run --name nodeeos -it -v /home/docker/dockerdata/eosdata:/opt/eos/bin/data-dir -p 8889:8888 -p 9876:9876 -p10023:22 -t eosio/eos nodeosd.sh

Port configuration：
0.0.0.0:10023 22/tcp
0.0.0.0:8889 8888/tcp
0.0.0.0:9876 9876/tcp

进入 nodeeos 容器操作：
[root@Docker-GP ~]# docker exec -it nodeeos /bin/bash

root@7b3105d46e8e:/opt/eosio/bin# ls
cleos data-dir eosio-abigen eosio-launcher eosio-s2wasm eosio-wast2wasm keosd nodeos nodeosd.sh

安装软件
root@7b3105d46e8e:/opt/eosio/bin# apt update
root@7b3105d46e8e:/opt/eosio/bin# apt install -y git
root@7b3105d46e8e:/opt/eosio/bin# apt-get -y install openssh-server

配置文件目录：
root@7b3105d46e8e:/# find | grep genesis.json
./root/eos/tutorials/bios-boot-tutorial/genesis.json
root@7b3105d46e8e:/# find | grep config.ini
./opt/eosio/bin/data-dir/config.ini
下面两个是与容器相关的配置文件
./root/eos/Docker/config.ini
./config.ini

基本使用

EOS 命令集

nodeos
启动节点（生产、测试、同步），查看块数据，同步主网等

cleos
命令行工具，创建钱包、账户（需要开启节点 nodeos）、key，绑定 key 到钱包等，一些对应命令需要 nodeos 加载对应的 plugin（可在 config.ini 中添加也可在 nodeos 后添加）

keosd
管理本地私钥，cleos 会自动启动 keosd

launcher
是用来创建多节点区块链的。nodeos 是用来创建单节点区块链。如果想要搭建多个节点的区块链，则可以通过 launcher 来配置。

eosiocpp
生成合约模板、编译合约

简单示例

1. 创建钱包
cleos wallet create

2. 执行合约
cleos -u http://192.168.1.121:8888 set contract eosio /media/yjl/32c65838-efbe-4218-97fc-fc5e9023a4c5/eos/build/contracts/eosio.bios -p eosio

3. 创建 key
cleos create key

4. 导入钱包
cleos wallet import private-key

5. 用 eosio 创建 eosio.token 账户
cleos -u http://192.168.1.121:8888 create account eosio eosio.token EOS8auHaGBT1XMbmPQWrBJyGgGp8pwVsBvyXGXqQ8RHVvHRsaEpfE EOS7fYHb5NMNwP762EvRZ5mruXvkKPJeV86fm7Grsm1QCpHXR7yyr

6. 执行 eosio.token 合约

cleos -u http://192.168.1.121:8888 set contract eosio.token /media/yjl/32c65838-efbe-4218-97fc-fc5e9023a4c5/eos/build/contracts/eosio.token -p eosio.token

7. 创建代币

cleos -u http://192.168.1.121:8888 push action eosio.token create '["eosio", "1000000000.0000 LLKJEOS", 0, 0, 0]' -p eosio.token

8. 发代币

cleos -u http://192.168.1.121:8888 push action eosio.token issue '["user", "100.0000 LLKJEOS", "memo"]' -p eosio

9. 转代币

cleos -u http://192.168.1.121:8888 push action eosio.token transfer '["user", "tester", "25.0000 LLKJEOS", "hellow"]' -p user

10. 查询余额

cleos -u http://192.168.1.121:8888 get currency balance eosio.token user

11. 查询 key 的绑定者

cleos -u http://192.168.1.121:8888 get accounts EOS7fYHb5NMNwP762EvRZ5mruXvkKPJeV86fm7Grsm1QCpHXR7yyr

12. 解锁钱包

cleos wallet unlock -n second-wallet --password PW5JLG2tSQDUhtujGzy4Tz5Wv7SZu2an97UfmPSeA13cLxSJi34mz

cleos wallet unlock --password 5KdjDfuTiWA5XVBvdSa67rc3VmosxF2LhnWX8K1UtJPwVLMjHDB

13. 修改

nodeos --config-dir /media/yjl/32c65838-efbe-4218-97fc-fc5e9023a4c5/yjl/nodeos/config
nodeos --data-dir /media/yjl/32c65838-efbe-4218-97fc-fc5e9023a4c5/yjl/nodeos/data

14. 获取账户信息

cleos get account eosio.token

15. 创建账户 eosio (eosio.token,exchange,eosio.msig)

cleos set contract eosio /media/yjl/32c65838-efbe-4218-97fc-fc5e9023a4c5/eos/build/contracts/eosio.bios -p eosio

16. 设置合约

cleos -u http://192.168.1.121:8888 set contract exchange /media/yjl/32c65838-efbe-4218-97fc-

fc5e9023a4c5/eos/build/contracts/exchange -p exchange

cleos -u http://192.168.1.121:8888 set contract eosio.msig /media/yjl/32c65838-efbe-4218-97fc-fc5e9023a4c5/eos/build/contracts/eosio.msig -p eosio.msig

17. 获取 keys
cleos wallet keys

实际操作

基础开发

eRemix 部署

root@7b3105d46e8e:~/browser-solidity# git clone https://github.com/ethereum/browser-solidity
root@7b3105d46e8e:~/browser-solidity# cd browser-solidity/
安装 node.js
root@7b3105d46e8e:~/browser-solidity# apt install -y nodejs
root@7b3105d46e8e:/# node -v
v8.10.0

安装 vi
root@7b3105d46e8e:~# apt install -y vim

root@7b3105d46e8e:~# node helloworld.js
Hello World
root@7b3105d46e8e:~# cat helloworld.js
console.log("Hello World");

安装 npm：
root@7b3105d46e8e:~# apt install -y npm
root@7b3105d46e8e:~# npm -v
3.5.2

修改 npm 源：
root@7b3105d46e8e:~# npm config set registry https://registry.npm.taobao.org
root@7b3105d46e8e:~# npm config get registry
https://registry.npm.taobao.org/

root@7b3105d46e8e:~/browser-solidity# npm install

项目显示转到新 git：
Remix-ide has been published as an npm module:
Remix-ide 已经发布为一个 npm 的模块了，难怪上面的项目没有 republish 脚本了。

直接运行：
root@7b3105d46e8e:~# npm install remix-ide -g
提示：
npm WARN optional Skipping failed optional dependency /remix-ide/chokidar/fsevents:
npm WARN notsup Not compatible with your operating system or architecture: fsevents@1.2.4

解决方案：更新 npm 版本即可，执行如下命令
root@7b3105d46e8e:~# npm i npm –g
root@7b3105d46e8e:/# npm install npm@latest -g
root@7b3105d46e8e:~# npm install remix-ide -g --no-optional

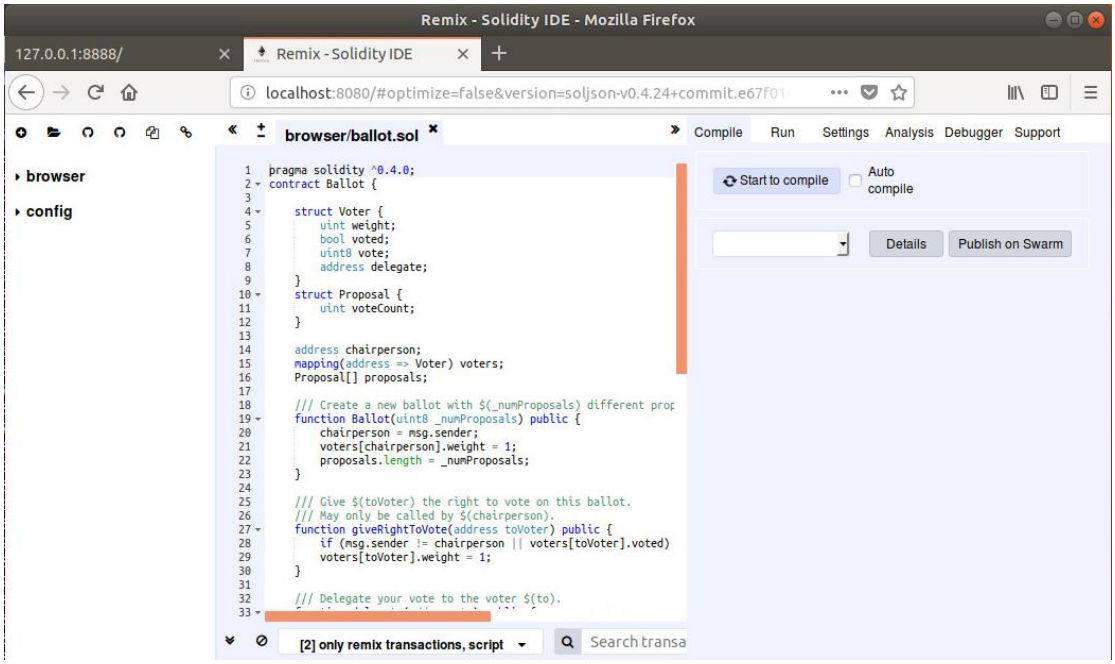
npm 安装 remix-ide 是报一些权限的问题，处理一下，软连接不能更新的问题。
Error: EACCES: permission denied, unlink '/usr/local/bin/remix-ide
david@david-vm-ether:~$ sudo rm /usr/loca/bin/remix-ide -rf

运行：
remix-ide

Error: Cannot find module './build/Release/scrypt'
david@david-vm-ether:/$ sudo vim /usr/local/lib/node_modules/remix-ide/node_modules/scrypt/index.js
将 require("./build/Release/scrypt")改为 require("scrypt")

david@david-vm-ether:/$ remix-ide
setup notifications for /
Shared folder : /
Starting Remix IDE at http://localhost:8080 and sharing /
Wed Jun 27 2018 14:59:40 GMT+0800 (CST) Remixd is listening on 127.0.0.1:65520

终于在人海中见到：



Or if you want to clone the github repository (wget need to be installed first) :
或者你需要 clone 站点，就按照下面指令操作，略过。
git clone https://github.com/ethereum/remix-ide.git
cd remix-ide
npm install
npm run setupremix　# this will clone https://github.com/ethereum/remix for you and link it to remix-ide
npm start

root@7b3105d46e8e:~# git clone https://remix.ethereum.org.

访问方式：
Run npm start and open http://127.0.0.1:8080 in your browser.

Truffle

http://truffle.tryblockchain.org/

Embark Framework

http://www.liyuechun.org/2017/11/30/embark-basic/

中级开发

高级开发

参考

官网：
https://ethereum.org/

ethereum(以太坊) - Ethereum Project
https://github.com/ethereum/
https://github.com/EOSIO/eos

官方 WIKI 文档：
https://github.com/EOSIO/eos/wiki

以太坊 Homestead 文档地址（官方）：
http://www.ethdocs.org/en/latest/index.html

以太坊网络状态地址（官方）：
https://ethstats.net/

以太坊资源网站（官方）：
http://ether.fund/
该网站提供了以太坊很多应用资源，比如市场情况、合约辅助工具、已发布的智能合约、以太坊网络、DAAP 等，方便开发和发布。

Solidity 编程文档（官方）：
http://solidity.readthedocs.io/en/latest/
该网站提供了以太坊 Solidity 语言的全面参考手册。

以太坊网络扫描（官方）：
http://etherscan.io/
该网站提供了以太坊网络的各种状态，比如帐号的详情、TOKEN 详情，难度详情、区块详情，非常方便和直观。

以太坊官方博客：
https://blog.ethereum.org/

以太坊 wiki 百科地址：
https://github.com/ethereum/wiki/wiki
在这里有白皮书、黄皮书以及开发指南，比较全面。

以太坊中文爱好者网站：
http://ethfans.org/
该网站为国内以太坊爱好者自发建立的网站，内容比较全，信息更新很快。

以太坊的 gitter 的实时交流网站：
https://gitter.im/orgs/ethereum/rooms
该网站是以太坊的 gitter chart 的网站，根据项目分了很多房间，只要你提问题，大部分有人回答，很不错，不过只能英语交流。

以太坊的官方论坛：
https://forum.ethereum.org/
官方论坛，没什么好说的，我遇到的几个问题，都是在这里找到的答案

以太坊第三方强大的 IDE：
https://live.ether.camp/
这个是第三方发布的 Solidity IDE，我看了一下，很强大，强大到还不太会用，目标是企业级 IDE，老外用的较多。

以太坊开发框架 Truffle 说明书：
http://truffle.readthedocs.io/en/latest/
以太坊目前很流行的开发框架 Truffle 的说明书，这个框架比较流行。

以太坊开发框架 dapple 说明书：
http://dapple.readthedocs.io/en/master/
这个开发框架是在 gitter chart 上看到的，感觉用的人不多，先观察

以太坊官方推荐开发框架 Meteor 说明书：
https://github.com/ethereum/wiki/wiki/Dapp-using-Meteor

REMIX 文档：
http://remix.readthedocs.io/en/latest/

欲想了解更多信息，请扫码入群：

如果您觉得受益想请鄙人吃个葱油饼，不妨打赏：

推荐使用微信支付