运行RNode的用户指南

介绍

节点是RChain网络的基础。网络层是架构中最低级别的组件，最终将支持RChain的大规模区块链操作。RNode的预发布版本可供用户浏览。RChain区块链平台的完整版本计划于2018年底发布。有关路线图，关键里程碑和发布计划的更多信息，请访问https://developer.rchain.coop。

这个指南为谁制定？

本文档是用户安装和运行软件、部署Rholang合约以及向区块链发起新区块的快速入门指南。有关用户和开发人员软件的详细信息，请访问https://github.com/rchain/rchain。

网络配置

默认情况下，RNode会不断尝试连接到其他对等节点方。成功连接到网络引导节点或网络上的其他对等节点方需要网络提供连接支持。请参阅RNode支持的网络配置并配置您的网络。

安装RNode

我们有各种安装包提供RNode软件。有关这些软件包的链接，请参阅https://developer.rchain.coop上的“入门”部分。

注意：在下面的命令示例中，您必须根据所需的RNode版本号进行更新，通过将“x”替换为安装版本中的数字。

Linux

平台	安装类型	安装信息
Debian 9 Stretch Ubuntu 16.04 LTS Ubuntu 16.04 LTS	Debian Package (.deb)	首次安装
		`apt install ./rnode_0.x.x_all.deb`
		再次安装
		`systemctl stop rnode && apt remove rnode && rm`

		`-rf /var/lib/rnode/rspace && apt install ./rnode_0.x.x_all.deb`

Fedora 27
Fedora 28
	RPM Package (.rpm)	首次安装
		`dnf install ./rnode-0.x.x-1.noarch.rpm`
		再次安装
		`systemctl stop rnode && dnf remove rnode && rm -rf /var/lib/rnode/rspace && dnf install ./rnode-0.x.x-1.noarch.rpm`

Other Linux distributions	Tarball (.tgz)	前提 ● Java - 我们推荐Open JDK 10，https：//openjdk.java.net/projects/jdk/10/ ● Libsodium - https://download.libsodium.org/doc/安装在标准前缀（/ user或/ user / local）中 首次安装 `tar -xvf rnode-0.x.x.tgz`

Mac

平台	安装类型	安装信息

Mac	Tarball (.tgz)	前提
		● Java - 我们推荐Open JDK 10，https：//openjdk.java.net/projects/jdk/10/
		● Libsodium - https://download.libsodium.org/doc/安装在标准前缀（/ user或/ user / local）中
		首次安装
		`tar -xvf rnode-0.x.x.tgz`
		`cd rnode-0.x.x`
		`./macos_install.sh`

		注意：`macos_install.sh`脚本在您的计算机上安装`Homebrew`软件包管理器，然后安装`libsodium`。 如果您已在计算机上安装了`Homebrew`，则可以参考脚本直接安装`libsodium`。

Windows

RNode不能在本机构建或Docker中运行。请参阅 RCHAIN-495 - LMDB fails to start in Windows Subsystem for Linux (WSL) **DONE** 以获取完整信息。在Windows中使用RNode的最佳方法是在Hyper-V中安装Ubuntu / Centos VM并使用Linux的安装说明。

Docker

虽然在Docker中安装RNode很简单，但了解使用Docker成功运行和与RNode的接口非常重要。

- 如果您是使用Docker的新手，请阅读Docker入门文档。
- 如果您熟悉Docker，查看或访问Docker入门文档每个部分末尾发布的备忘单很有帮助。

```
docker pull rchain/rnode
```

ARM

您可以在Raspberry pi上运行RNode。有关此示例，包括安装和部署说明，请访问https://github.com/kayvank/arm-rnode。

运行RNode

运行命令（非Docker）

连接到现有网络

```
rnode run --default-timeout 6000 --thread-pool-size 200 -b
```

```
rnode:<bootstrap node address>
```

- 引导程序地址 - 输入要连接的引导程序节点的地址。请参阅RChain核心开发团队支持的引导节点的RNode引导程序地址。
- 验证者私钥 - 如果您是测试网的验证者，或者您正在创建专用网络并且您的创世块中包含bonds file，请插入密钥。
- 网络配置 - 如果要指定端口，请在运行命令中包含--p。如果要指定主机，请在运行命令中包含--host。
- 线程池大小 - 默认情况下，RNode允许非常多的线程。使用此标志可限制节点的数量以提高性能。

运行独立节点

```
rnode run -s
```

运行命令（Docker）
创建RNode网络
如果这是您第一次运行RNode，则需要创建Docker网络以支持RNode操作。您只需要执行一次，除非您删除了您的Docker系统的信息。

```
docker network create rnode-net
```

创建数据目录
使用Docker运行RNode需要/ var / lib / rnode目录。每个Docker容器都必须拥有自己的数据目录。如果系统上有多个目录，则可以在docker run命令中单独指定它们。

```
mkdir $HOME/rnode
```

连接已知网络

```
docker run -it --rm --network rnode-net --name rnode1 -v
$HOME/var/rnode1:/var/lib/rnode rchain/rnode:latest run  --bootstrap
<bootstrap node address> --thread-pool-size 200
```

注意：Docker for Mac仅适用于静态NAT和端口转发。 network = host在Mac上不起作用。有关静态NAT和端口转发的详细信息，请参阅RNode支持的网络配置。

引导程序地址 - 输入要连接的引导程序节点的地址。请参阅RChain核心开发团队支持的引导节点的RNode引导程序地址。
验证者私钥 - 如果您是测试网的验证者，或者您正在创建专用网络并且您的创世块中包含bonds file，请插入密钥。
网络配置 - 如果要指定端口，请在运行命令中包含--p。如果要指定主机，请在运行命令中包含--host。

运行独立节点

```
docker run -u root -it --network rnode-net --name rnode-server-local -v
"$HOME/rnode":/var/lib/rnode rchain/rnode:latest run --standalone
```

在Docker中使用RNode的提示
命名容器
创建网络后，服务器容器将被放入网络，然后由客户端引用。如果为服务器容器命名，则更容易。这是命名服务器'rnode-server-local'的示例。

```
docker run --name rnode-server-local rchain/rnode:latest
```

使用--host标志
如果要创建由某些节点和引导节点组成的本地docker网络，则必须使用--host标志指定节点的地址。您要确保不将节点的IP地址用于--host flag，而是使用主机名。如果网络名为“rnode-net”并且您将docker容器命名为“rnode-server-local”，则该docker容器的主机名为“rnode-server-local.rnode-net”。

与容器共享目录
要与容器共享目录，请使用volume命令。您需要在本地系统上创建一个目录，该目录将存储所有与RNode相关的文件。创建目录后，可以使用volume命令与Docker容器共享此目录。下面是如何将volume参数指定为run命令的一部分的示例。

RNode需要在启动时有这样的路径/ var / lib / rnode。每个RNode都需要自己独立的/ var / lib / rnode目录。

```
docker run -v "path to local directory":/var
```

如何与正在运行的RNode进行对接

运行RNode后，打开一个与服务器端API接口的新终端窗口，以获取对等节点的计数，获取区块数，部署Rholang合约，并发起新区块。

检查RNode的版本
您可以要求RNode告诉您它的版本和githash。

```
curl -s localhost:40403/version
```

版本请求响应的示例

```
root@kelly:~# curl -s localhost:40403/version
RChain Node 0.6.4 (5d620e25b42f328d046cf4b1596446be7f55fe9d)
```

得到对等节点的数量
RNode提供两种类型的对等节点。 Kademlia节点可以给在节点发现中找到的对等节点数量提供参考。Peers 表示连接到RNode的对等节点数量。

以下是包含（Kademlia）节点和（RNode）`peers` status JSON响应（手动格式化）示例。

```
root@salt:~# curl -s http://localhost:40403/status
{
    "nodes": 23,
    "peers": 18,
    "version": "RChain Node 0.8.2
(41329d855c1f7a7fcc7102ceda266af99b20e974)"
}
```

得到对等节点的数量
从status JSON响应中提取`peers`字段

```
curl -s http://localhost:40403/status
```

在Docker中获取对等节点数量
从status JSON响应中提取`peers`字段

```
sudo docker exec -ti <containername> curl -s
http://localhost:40403/status
```

获取区块数和显示区块数
使用此命令显示区块链中的区块。输出的是区块和总区块数的元组空间内容。

```
rnode show-blocks
```

获取Docker中的区块数

```
docker run -it --rm --network rnode-net --name rnode -v
$HOME/var/rholang:/var/ rchain/rnode:latest --grpc-host rnode2
show-blocks
```

部署Rholang合约
Deploy是一个API调用。 deploy命令接受Rholang文件并将其发送到正在运行的RNode实例以供执行。Deploy执行Rholang代码并将Rholang代码作为下一个区块添加的一部分。Deploy需要保存在客户端系统上的Rholang文件的路径以及将执行代码部署的目标服务器。

随着https://github.com/rchain/rchain/pull/1555的合并，RNode的deploy命令发生了变化。现在，Deploy命令需要值大于0的phlo-limit和phlo-price。

部署成功后，用户可能会看到CostAccount（__ some_value __，Cost（__ some_value__））的消息。第一个__some_value__表示减少/执行其代码所需的步骤数，第二个__some_value__表示它的成本。之所以说“用户可能会看到”是取决于他们执行程序的方式。我们将在不久的将来使该消息更加通俗易懂。

部署Rholang合约

```
rnode deploy --phlo-limit <value> --pholo-price <value> <path to .rho
file>
```

在Docker里部署Rholang合约

```
docker run -it --rm --network rnode-net --name rnode-deploy1 -v
$HOME/var/rholang:/var/ rchain/rnode:latest --grpc-host rnode1 deploy
--from "0x1" --phlo-limit <value> --phlo-price <value> <path to Rholang
file>
```

如果您是锁定保证金的验证人，请向区块链发起新区块

propose触发区块添加。这是仅适用于锁定保证金的验证者的功能。您可以发起一个区块，或者其他人在您的节点上部署合约，或者您在您的节点上部署合约。

以下是服务器端的propose命令示例。

```
root@salt:~# rnode propose
21:52:06.959 [main] INFO  c.r.n.configuration.Configuration$ - Starting
with profile default
Response: Success! Block c0b68d2520... created and added.
```

以下是随着propose命令之后的控制台输出示例。日志显示了发起和添加的区块（c0b68d2520 ...）中的新分叉选项。

```
21:52:08.806 [grpc-default-executor-55] INFO
c.rchain.casper.util.comm.CommUtil$ - CASPER: Beginning send of Block #1
(c0b68d2520...) -- Sender ID 3d86379153... -- M Parent Hash
06eb7dc6ab... -- Contents 16d7c61fa6...-- Shard ID rchain to peers...
21:52:08.820 [repl-io-141] INFO  c.rchain.casper.util.comm.CommUtil$ -
CASPER: Sent c0b68d2520... to peers
21:52:08.820 [repl-io-141] INFO  c.rchain.casper.MultiParentCasper$ -
CASPER: Added c0b68d2520...
21:52:08.844 [repl-io-141] INFO  c.rchain.casper.MultiParentCasper$ -
CASPER: New fork-choice tip is block c0b68d2520....
```

发起一个新区块

```
rnode propose
```

在Docker中发起一个新区块

```
docker run -it --rm --network rnode-net --name rnode-propose1 -v
$HOME/var/rholang:/var/ rchain/rnode:latest --grpc-host rnode1 propose
```

==========小郝校对以下内容==========

获取帮助

查看可用于RNode的所有命令行选项

```
--help
```

在Docker里获取帮助

```
docker run rchain/rnode --help
```

绑定到网络以成为验证者
要参与RChain权益证明共识协议，您必须在网络上放置权益以成为权益验证者。以下说明支持绑定到网络，其中创世块包括龙头的实施。

生成公钥/私钥对

运行RNode并连接到引导节点

有关可用地址，请参阅RNode引导程序地址。

```
rnode run --default-timeout 6000 -b "<bootstrap address>"
```

```
--validator-private-key <private key>
```

生成Deploy文件

要绑定到网络，您将部署两个文件。此步骤支持生成这些文件。完成此命令后，目录中将有2个新的.rho文件：forward_filename.rho和bond_filename.rho。

```
rnode generateFaucetBondingDeploys --amount <bond amount> --private-key
<private key > --public-key <public key > -s ed25519
```

- <bond amont> - 这个时候数量是任意的所以得写 >2

将转发文件部署到绑定的验证程序

在此步骤中，您将转发文件部署到绑定的验证程序节点。您将需要此验证程序节点的IP地址。

```
rnode --grpc-host <IP address of a validator node> deploy --phlo-limit
10000000000 --phlo-price 1 <forward_filename.rho>
```

等待绑定的验证程序添加包含转发文件部署的区块

在此步骤中，您正在等待部署转发文件的节点发起包含该部署的区块。

将绑定文件部署到绑定的验证程序上

在此步骤中，您将将绑定文件部署到绑定的验证程序节点。您将需要此验证程序节点的IP地址。

```
rnode --grpc-host <IP address of a validator node> deploy --phlo-limit
10000000000 --phlo-price 1 <bond_filename.rho>
```

等待绑定验证者发起包含绑定文件部署的区块

在此步骤中，您正在等待部署绑定文件的节点发起包含该部署的区块。

确认您已被绑定

在步骤4中部署之后，使用发起的区块的哈希运行

```
rnode show-block <block hash>
```

在区块信息部分中，要么搜索<公钥>，要么<绑定金额/bond amont> - 1（例如，如果您的bond amont为500，则为499，反映加盟费用减少1 REV）。

或者，在区块内容中grep或搜索您的公钥。如果您看到公钥和保证金金额 - 您已被锁定保证金！

创建私链

您可以创建自己的区块链网络。

私有区块链的先决条件

- 1个引导程序节点
- 在网络上运行的其他2个节点实例，它们可以相互通信（彼此的对等节点）。
- 节点实例的密钥（签名和创建bond.txt文件时需要这些密钥）
- 所有节点实例均可访问的bond.txt文件。您可以提供它，也可以使用系统生成的bonds file。
- 要在整个网络中部署的Rholang文件。

创建私有区块链的步骤

- 启动引导节点。这是在独立模式下运行的1节点。
- 在对等节点节点的run命令中包含bootstrap节点的地址。

加入RChain测试网

您可以加入RChain测试网。

- 获取引导节点的版本。使用节点的IP地址和度量端口查询版本

```
http://<bootstrap IP address>:40403/version
```

- 引导到测试网上的已知节点或使用RNode引导程序地址上显示的测试网引导程序地址。
- 确认您拥有正确的版本并使用上述加入现有网络的说明加入网络。
- 要作为验证者进行绑定，请遵循上面作为验证者的绑定说明。您将部署合约到52.119.8.203或52.119.8.204。这些节点每10分钟propose一次。

监控节点

RNode与Prometheus集成。这些说明描述了使用Prometheus通过Docker-compose进行RNode指标收集和显示的方法。

可视化区块链

为了支持调试，我们从节点收集信息并在graphviz中使用它来创建DAG的可视化。以下是如何执行此操作的说明。

指导	示例/说明
在RNode上运行程序	`rnode vdag`

观察图文的生成并复制输出内容	`digraph dag {`
	`  rankdir=BT`
	`  node [width=0 height=0 margin=0.03 fontsize=8]`
	`  subgraph lvl0 {`
	`  rank=same`
	`  "0"`
	`  "14451c5c72..." [shape=Msquare]`
	`  }  subgraph lvl1 {`
	`  rank=same`
	`  "1"`
	`  "be37ade65c..." [color="#1ad3b4" shape=record label="{be37ade65c...|1ad3b44c9e...}"]`
	`  "4c98df3a00..." [color="#e6c1e8" shape=record label="{4c98df3a00...|e6c1e846bc...}"]`

```
    "48370c8117..." [color="#db430c" shape=record
label="{48370c8117...|db430c34a6...}"]
  }
  "be37ade65c..." -> "14451c5c72..." []
  "4c98df3a00..." -> "14451c5c72..." []
  "48370c8117..." -> "14451c5c72..." []
  subgraph lvl2 {
    rank=same
    "2"
    "754737e128..." [color="#1ad3b4" shape=record
label="{754737e128...|1ad3b44c9e...}"]
    "65462bd1dc..." [color="#e6c1e8" shape=record
label="{65462bd1dc...|e6c1e846bc...}"]
    "fda2951a85..." [color="#db430c" shape=record
label="{fda2951a85...|db430c34a6...}"]
  }
  "754737e128..." -> "be37ade65c..." []
  "65462bd1dc..." -> "4c98df3a00..." []
  "fda2951a85..." -> "48370c8117..." []
  subgraph lvl3 {
    rank=same
    "3"
    "e664d6d3e6..." [color="#1ad3b4" shape=record
label="{e664d6d3e6...|1ad3b44c9e...}"]
    "b2185fefe5..." [color="#e6c1e8" shape=record
label="{b2185fefe5...|e6c1e846bc...}"]
    "21e840b706..." [color="#db430c" shape=record
label="{21e840b706...|db430c34a6...}"]
  }
  "e664d6d3e6..." -> "754737e128..." []
  "b2185fefe5..." -> "65462bd1dc..." []
  "21e840b706..." -> "fda2951a85..." []
  subgraph lvl4 {
    rank=same
    "4"
    "97769ce1e5..." [color="#1ad3b4" shape=record
```

```
label="{97769ce1e5...|1ad3b44c9e...}"]
    "0cda5ecdc9..." [color="#e6c1e8" shape=record
label="{0cda5ecdc9...|e6c1e846bc...}"]
    "f7797f30ae..." [color="#db430c" shape=record
label="{f7797f30ae...|db430c34a6...}"]
  }
  "97769ce1e5..." -> "e664d6d3e6..." []
  "0cda5ecdc9..." -> "b2185fefe5..." []
  "f7797f30ae..." -> "21e840b706..." []
  subgraph lvl5 {
    rank=same
    "5"
    "4c631f0c6a..." [color="#1ad3b4" shape=record
label="{4c631f0c6a...|1ad3b44c9e...}"]
    "6202221ebb..." [color="#e6c1e8" shape=record
label="{6202221ebb...|e6c1e846bc...}"]
    "8a7c674316..." [color="#db430c" shape=record
label="{8a7c674316...|db430c34a6...}"]
  }
  "4c631f0c6a..." -> "97769ce1e5..." []
  "6202221ebb..." -> "0cda5ecdc9..." []
  "8a7c674316..." -> "f7797f30ae..." []
  subgraph lvl6 {
    rank=same
    "6"
    "14a38ca3a4..." [color="#1ad3b4" shape=record
label="{14a38ca3a4...|1ad3b44c9e...}"]
    "e5269a7c93..." [color="#e6c1e8" shape=record
label="{e5269a7c93...|e6c1e846bc...}"]
    "6301adcbae..." [color="#db430c" shape=record
label="{6301adcbae...|db430c34a6...}"]
  }
  "14a38ca3a4..." -> "4c631f0c6a..." []
  "e5269a7c93..." -> "6202221ebb..." []
  "6301adcbae..." -> "8a7c674316..." []
  subgraph lvl7 {
```

```
    rank=same
    "7"
    "f8952b5024..." [color="#e6c1e8" shape=record
label="{f8952b5024...|e6c1e846bc...}"]
    "0c87faf68d..." [color="#1ad3b4" shape=record
label="{0c87faf68d...|1ad3b44c9e...}"]
    "b2f9bf40af..." [color="#db430c" shape=record
label="{b2f9bf40af...|db430c34a6...}"]
  }
  "f8952b5024..." -> "e5269a7c93..." []
  "0c87faf68d..." -> "b2185fefe5..." []
  "0c87faf68d..." -> "14a38ca3a4..." []
  "b2f9bf40af..." -> "6301adcbae..." []
  subgraph lvl8 {
    rank=same
    "8"
    "e004b5f123..." [color="#e6c1e8" shape=record
label="{e004b5f123...|e6c1e846bc...}"]
    "cff175d457..." [color="#db430c" shape=record
label="{cff175d457...|db430c34a6...}"]
  }
  "e004b5f123..." -> "f8952b5024..." []
  "cff175d457..." -> "b2f9bf40af..." []
  subgraph lvl9 {
    rank=same
    "9"
    "a7a21222e5..." [color="#e6c1e8" shape=record
label="{a7a21222e5...|e6c1e846bc...}"]
    "7e5dad6dbf..." [color="#db430c" shape=record
label="{7e5dad6dbf...|db430c34a6...}"]
  }
  "a7a21222e5..." -> "e004b5f123..." []
  "7e5dad6dbf..." -> "cff175d457..." []
  subgraph lvl10 {
    rank=same
    "10"
```

```
    "0389e6efb7..." [color="#e6c1e8" shape=record
label="{0389e6efb7...|e6c1e846bc...}"]
    "de2f0def42..." [color="#db430c" shape=record
label="{de2f0def42...|db430c34a6...}"]
  }
  "0389e6efb7..." -> "a7a21222e5..." []
  "de2f0def42..." -> "7e5dad6dbf..." []
  subgraph lvl11 {
    rank=same
    "11"
    "65881caaeb..." [color="#e6c1e8" shape=record
label="{65881caaeb...|e6c1e846bc...}"]
    "30544ff802..." [color="#db430c" shape=record
label="{30544ff802...|db430c34a6...}"]
  }
  "65881caaeb..." -> "0389e6efb7..." []
  "30544ff802..." -> "de2f0def42..." []
  subgraph lvl12 {
    rank=same
    "12"
    "20c6d8b1a4..." [color="#e6c1e8" shape=record
label="{20c6d8b1a4...|e6c1e846bc...}"]
    "0d83313112..." [color="#db430c" shape=record
label="{0d83313112...|db430c34a6...}"]
  }
  "20c6d8b1a4..." -> "65881caaeb..." []
  "0d83313112..." -> "30544ff802..." []
  subgraph lvl13 {
    rank=same
    "13"
    "9b91555903..." [color="#e6c1e8" shape=record
label="{9b91555903...|e6c1e846bc...}"]
    "e43db8ddf5..." [color="#db430c" shape=record
label="{e43db8ddf5...|db430c34a6...}"]
  }
  "9b91555903..." -> "20c6d8b1a4..." []
```

```
  "e43db8ddf5..." -> "0d83313112..." []
  subgraph lvl14 {
    rank=same
    "14"
    "63f5b885ac..." [color="#e6c1e8" shape=record
label="{63f5b885ac...|e6c1e846bc...}"]
    "a3fd8825b2..." [color="#1ad3b4" shape=record
label="{a3fd8825b2...|1ad3b44c9e...}"]
  }
  "63f5b885ac..." -> "9b91555903..." []
  "a3fd8825b2..." -> "e43db8ddf5..." []
  subgraph lvl15 {
    rank=same
    "15"
    "87a58cbd1a..." [color="#e6c1e8" shape=record
label="{87a58cbd1a...|e6c1e846bc...}"]
  }
  "87a58cbd1a..." -> "63f5b885ac..." []
  subgraph lvl16 {
    rank=same
    "16"
    "dd11cbc8dc..." [color="#e6c1e8" shape=record
label="{dd11cbc8dc...|e6c1e846bc...}"]
  }
  "dd11cbc8dc..." -> "87a58cbd1a..." []
  subgraph lvl17 {
    rank=same
    "17"
    "9207e959d6..." [color="#e6c1e8" shape=record
label="{9207e959d6...|e6c1e846bc...}"]
  }
  "9207e959d6..." -> "dd11cbc8dc..." []
  subgraph lvl18 {
    rank=same
    "18"
    "a87be9d496..." [color="#e6c1e8" shape=record
```

```
label="{a87be9d496...|e6c1e846bc...}"]
  }
  "a87be9d496..." -> "9207e959d6..." []
  subgraph lvl19 {
    rank=same
    "19"
    "f7842ffcc1..." [color="#e6c1e8" shape=record
label="{f7842ffcc1...|e6c1e846bc...}"]
  }
  "f7842ffcc1..." -> "a87be9d496..." []
  subgraph lvl20 {
    rank=same
    "20"
    "ea4e333f1a..." [color="#e6c1e8" shape=record
label="{ea4e333f1a...|e6c1e846bc...}"]
  }
  "ea4e333f1a..." -> "f7842ffcc1..." []
  "ea4e333f1a..." -> "14a38ca3a4..." []
  subgraph lvl21 {
    rank=same
    "21"
    "615c23476b..." [color="#e6c1e8" shape=record
label="{615c23476b...|e6c1e846bc...}"]
  }
  "615c23476b..." -> "ea4e333f1a..." []
  subgraph lvl22 {
    rank=same
    "22"
    "d7105e1958..." [color="#e6c1e8" shape=record
label="{d7105e1958...|e6c1e846bc...}"]
  }
  "d7105e1958..." -> "615c23476b..." []
  subgraph lvl23 {
    rank=same
    "23"
    "67a01e7ddf..." [color="#e6c1e8" shape=record
```

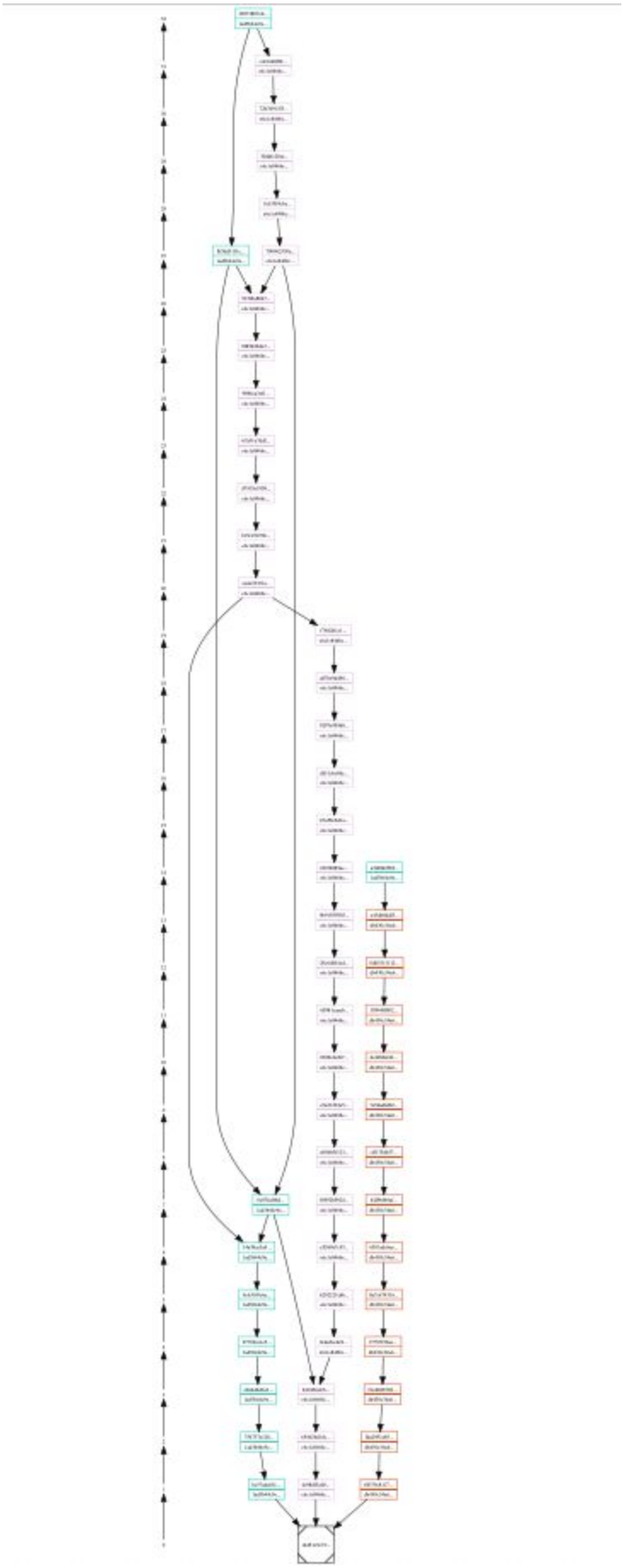
```
label="{67a01e7ddf...|e6c1e846bc...}"]
  }
  "67a01e7ddf..." -> "d7105e1958..." []
  subgraph lvl24 {
    rank=same
    "24"
    "9984ca3ef1..." [color="#e6c1e8" shape=record
label="{9984ca3ef1...|e6c1e846bc...}"]
  }
  "9984ca3ef1..." -> "67a01e7ddf..." []
  subgraph lvl25 {
    rank=same
    "25"
    "0f85d26de3..." [color="#e6c1e8" shape=record
label="{0f85d26de3...|e6c1e846bc...}"]
  }
  "0f85d26de3..." -> "9984ca3ef1..." []
  subgraph lvl26 {
    rank=same
    "26"
    "515fba8847..." [color="#e6c1e8" shape=record
label="{515fba8847...|e6c1e846bc...}"]
  }
  "515fba8847..." -> "0f85d26de3..." []
  subgraph lvl27 {
    rank=same
    "27"
    "700962330a..." [color="#e6c1e8" shape=record
label="{700962330a...|e6c1e846bc...}"]
    "8f5dd31f1c..." [color="#1ad3b4" shape=record
label="{8f5dd31f1c...|1ad3b44c9e...}"]
  }
  "700962330a..." -> "515fba8847..." []
  "700962330a..." -> "0c87faf68d..." []
  "8f5dd31f1c..." -> "515fba8847..." []
  "8f5dd31f1c..." -> "0c87faf68d..." []
```

```
  subgraph lvl28 {
    rank=same
    "28"
    "fc635f4c6a..." [color="#e6c1e8" shape=record
label="{fc635f4c6a...|e6c1e846bc...}"]
  }
  "fc635f4c6a..." -> "700962330a..." []
  subgraph lvl29 {
    rank=same
    "29"
    "30ddfc20ee..." [color="#e6c1e8" shape=record
label="{30ddfc20ee...|e6c1e846bc...}"]
  }
  "30ddfc20ee..." -> "fc635f4c6a..." []
  subgraph lvl30 {
    rank=same
    "30"
    "72b34913f3..." [color="#e6c1e8" shape=record
label="{72b34913f3...|e6c1e846bc...}"]
  }
  "72b34913f3..." -> "30ddfc20ee..." []
  subgraph lvl31 {
    rank=same
    "31"
    "cdc16d8ff0..." [color="#e6c1e8" shape=record
label="{cdc16d8ff0...|e6c1e846bc...}"]
  }
  "cdc16d8ff0..." -> "72b34913f3..." []
  subgraph lvl32 {
    rank=same
    "32"
    "10073807ed..." [color="#1ad3b4" shape=record
label="{10073807ed...|1ad3b44c9e...}"]
  }
  "10073807ed..." -> "cdc16d8ff0..." []
  "10073807ed..." -> "8f5dd31f1c..." []
```

```
subgraph timeseries {
  "0" [shape=plaintext]
  "1" [shape=plaintext]
  "2" [shape=plaintext]
  "3" [shape=plaintext]
  "4" [shape=plaintext]
  "5" [shape=plaintext]
  "6" [shape=plaintext]
  "7" [shape=plaintext]
  "8" [shape=plaintext]
  "9" [shape=plaintext]
  "10" [shape=plaintext]
  "11" [shape=plaintext]
  "12" [shape=plaintext]
  "13" [shape=plaintext]
  "14" [shape=plaintext]
  "15" [shape=plaintext]
  "16" [shape=plaintext]
  "17" [shape=plaintext]
  "18" [shape=plaintext]
  "19" [shape=plaintext]
  "20" [shape=plaintext]
  "21" [shape=plaintext]
  "22" [shape=plaintext]
  "23" [shape=plaintext]
  "24" [shape=plaintext]
  "25" [shape=plaintext]
  "26" [shape=plaintext]
  "27" [shape=plaintext]
  "28" [shape=plaintext]
  "29" [shape=plaintext]
  "30" [shape=plaintext]
  "31" [shape=plaintext]
  "32" [shape=plaintext]
  "0" -> "1" []
  "1" -> "2" []
```

```
    "2" -> "3" []
    "3" -> "4" []
    "4" -> "5" []
    "5" -> "6" []
    "6" -> "7" []
    "7" -> "8" []
    "8" -> "9" []
    "9" -> "10" []
    "10" -> "11" []
    "11" -> "12" []
    "12" -> "13" []
    "13" -> "14" []
    "14" -> "15" []
    "15" -> "16" []
    "16" -> "17" []
    "17" -> "18" []
    "18" -> "19" []
    "19" -> "20" []
    "20" -> "21" []
    "21" -> "22" []
    "22" -> "23" []
    "23" -> "24" []
    "24" -> "25" []
    "25" -> "26" []
    "26" -> "27" []
    "27" -> "28" []
    "28" -> "29" []
    "29" -> "30" []
    "30" -> "31" []
    "31" -> "32" []
  }
}
```

要查看可视化图表，请访问 http://viz-js.com/



要将图形添加到Jira ticket，另存为.txt文件并附加到ticket	

定义

Bonds.txt
bonds.txt文件是用于模拟验证者绑定的临时构造。验证者绑定的合约尚不存在，此文件提供了有关验证工具集的必要信息，以便正常工作。bonds file的结构如下：

```
"private key for validator 1" stake
"private key for validator 2" stake
```

如果使用临时验证者密钥连接到RChain支持的网络，系统将自动生成bonds.txt文件。

如果设置私有区块链网络，您可以创建并指定自己的bonds.txt文件。

引导节点

如果要连接到RChain引导程序节点，请根据您使用的软件版本选择引导程序地址。请参阅的RNode引导程序地址获取地址。

验证者密钥

如果您计划连接到RChain支持的引导程序节点，请从此处获取临时密钥对。

超越快速入门：自定义和更多信息

以下信息不局限于快速入门信息，用来帮助RNode操作员进行自定义并使用其他界面功能。

RNode服务器网络选项

RNode支持多种网络功能。默认情况下，RNode将启动并尝试连接到RChain引导程序节点。您可以使用以下选项进行自定义。

选项	描述	运行标志	示例
Host	明确地设置主机名字或这个节点的IP	--host <arg>	--host 145.34.21.7
Port	可以用的网络端口，默认使用40400沟通	--port <arg>	--port 50040
gRPC Port	Port which exposes the gRPC API. Defaults to 40401	--grpc-port <arg>	--grpc-port 50041
Standalone	独立运行（比如引导节点）	--standalone	
Bootstrap	制定一个具体的引导节点地址	--bootstrap <arg>	--b rnode://c61769b39d368cbcbc9499634e030386c79d5b02@52.119.8.108:30304
Disable UPnP	关闭 UPnP 模式，只在配置了静态 NAT的情况下使用	--no-unpnp	

RNode用户界面

Github上提供了完整的API调用列表。

调用API

RNode API是服务器端API。要在RNode运行后访问本地RNode服务器，请打开一个新窗口并使用以下命令调用RNode api：

```
rnode <API call>
```

从远程服务器调用API

如果可以通过指定主机服务器和主机服务器端口来调用远程RNode服务器的API。

选项	描述	句法	论证格式
主机服务器	主机的IP地址会收到调用信息	--grpc-host	100.10.25.75
主机服务器端口	服务器上 gRPC API 的端口会收到调用信息	--grpc-port	40401

这是对远程服务器的API调用的示例

```
./bin/rnode --grpc-host IP.Address.of.server --grpc-port 40401 repl
```

API调用列表

API	描述	调用信号
翻译器REPL	与Interpreter REPL对接以运行一行Rholang代码	repl
翻译器Eval	使用Interpreter评估Rholang文件	eval
部署	部署Rholang合约。使用Rholang代码调用以前部署的合约	deploy
添加	发起一个区块	propose
查看所有区块	查看区块链中的区块	show-blocks

通过REPL访问Rholang翻译器

RChain节点有一个支持REPL（读取，评估，打印，循环）的解释器。对本地服务器上公开的REPL API的示例调用：

```
./bin/rnode repl
```

通过EVAL评估Rholang文件

RNode Interpreter支持通过EVAL评估Rholang文件。该命令接受Rholang文件的文件名并提示它的评估。文件必须具有“.rho”的扩展名。此API将请求发送到正在运行的RNode服务器实例。这对于在将合约部署到区块链之前测试Rholang合约非常有用。

```
rnode eval <path to file to evaluate>
```

常问问题

我不得不重启我的节点。重新加入网络后，我看不到什么区块？

当您重新启动节点时，必须等待网络上的其他人提出propose，然后您的节点将赶上区块链的状态。请不要propose，直到您的节点收到propose，因为它创建一个分支并产生错误风险，如InvalidUnslashableBlock。