

Life is short, you need Spark!



从**零**开始

不需要任何基础，带领您无痛入门 Spark

云计算分布式大数据 Spark 实战高手之路

王家林著

Spark 亚太研究院系列丛书 版权所有

伴随着大数据相关技术和产业的逐步成熟，继 Hadoop 之后，Spark 技术以其无可比拟的优势，发展迅速，将成为替代 Hadoop 的下一代云计算、大数据核心技术。

本书特点

- ▶ 云计算分布式大数据 Spark 实战高手之路三部曲之第一部
- ▶ 网络发布版为图文并茂方式，边学习，边演练
- ▶ 不需要任何前置知识，从零开始，循序渐进

本书作者



Spark 亚太研究院院长和首席专家，中国目前唯一的移动互联网和云计算大数据集大成者。在 Spark、Hadoop、Android 等方面有丰富的源码、实务和性能优化经验。彻底研究了 Spark 从 0.5.0 到 0.9.1 共 13 个版本的 Spark 源码，并已完成 2014 年 5 月 31 日发布的 Spark1.0 源码研究。

Hadoop 源码级专家，曾负责某知名公司的类 Hadoop 框架开发工作，专注于 Hadoop 一站式解决方案的提供，同时也是云计算分布式大数据处理的最早实践者之一。

Android 架构师、高级工程师、咨询顾问、培训专家。

通晓 Spark、Hadoop、Android、HTML5，迷恋英语播音和健美。

“真相会使你获得自由。”

— 耶稣《圣经》约翰 8:32KJV

“所有人类的幸福都来源于不能直面事实。”

— 释迦摩尼

“道法自然”

— 老子《道德经》第 25 章

《云计算分布式大数据 Spark 实战高手之路》

系列丛书三部曲

《云计算分布式大数据 Spark 实战高手之路---从零开始》：

不需要任何基础，带领您无痛入门 Spark 并能够轻松处理 Spark 工程师的日常编程工作，内容包括 Spark 集群的构建、Spark 架构设计、RDD、Shark/SparkSQL、机器学习、图计算、实时流处理、Spark on Yarn、JobServer、Spark 测试、Spark 优化等。

《云计算分布式大数据 Spark 实战高手之路---高手崛起》：

大话 Spark 源码，全世界最有情趣的源码解析，过程中伴随诸多实验，解析 Spark 1.0 的任何一句源码！更重要的是，思考源码背后的问题场景和解决问题的设计哲学和实现招式。

《云计算分布式大数据 Spark 实战高手之路---高手之巅》：

通过当今主流的 Spark 商业使用方法和最成功的 Hadoop 大型案例让您直达高手之巅，从此一览众山小。



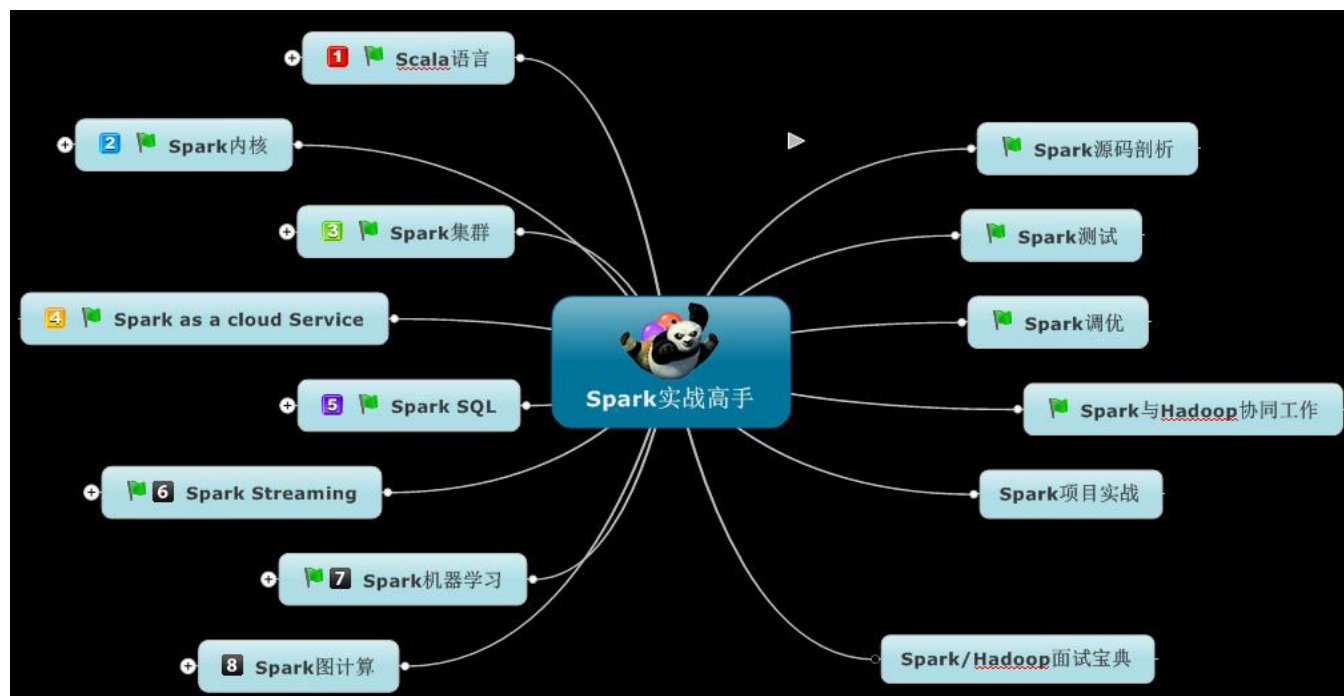
《前言》

Spark 采用一个统一的技术堆栈解决了云计算大数据的如流处理、图技术、机器学习、NoSQL 查询等方面的所有核心问题，具有完善的生态系统，这直接奠定了其一统云计算大数据领域的霸主地位；

要想成为 Spark 高手，需要经历六大阶段



Spark 实战高手之核心技能点



第一阶段：熟练的掌握 Scala 语言

1. Spark 框架是采用 Scala 语言编写的，精致而优雅。要想成为 Spark 高手，你就必须阅读 Spark 的源代码，就必须掌握 Scala；
2. 虽然说现在的 Spark 可以采用多语言 Java、Python 等进行应用程序开发，但是最快速的和支持最好的开发 API 依然并将永远是 Scala 方式的 API，所以你必须掌握 Scala 来编写复杂的和高性能的 Spark 分布式程序；
3. 尤其要熟练掌握 Scala 的 trait、apply、函数式编程、泛型、逆变与协变等；

推荐课程：“精通 Spark 的开发语言：Scala 最佳实践”

第二阶段：精通 Spark 平台本身提供给开发者 API

1. 掌握 Spark 中面向 RDD 的开发模式 掌握各种 transformation 和 action 函数的使用；
2. 掌握 Spark 中的宽依赖和窄依赖以及 lineage 机制；
3. 掌握 RDD 的计算流程，例如 Stage 的划分、Spark 应用程序提交给集群的基本过程和 Worker 节点基础的工作原理等

推荐课程：“18 小时内掌握 Spark：把云计算大数据速度提高 100 倍以上!”

第三阶段：深入 Spark 内核

此阶段主要是通过 Spark 框架的源码研读来深入 Spark 内核部分：

1. 通过源码掌握 Spark 的任务提交过程；
2. 通过源码掌握 Spark 集群的任务调度；
3. 尤其要精通 DAGScheduler、TaskScheduler 和 Worker 节点内部的工作的每一步的细节；

推荐课程：[“Spark 1.0.0 企业级开发动手：实战世界上第一个 Spark 1.0.0 课程，涵盖 Spark 1.0.0 所有的企业级开发技术”](#)

第四阶段：掌握基于 Spark 上的核心框架的使用

Spark 作为云计算大数据时代的集大成者，在实时流处理、图技术、机器学习、NoSQL 查询等方面具有显著的优势，我们使用 Spark 的时候大部分时间都是在使用其上的框架例如 Shark、Spark Streaming 等：

1. Spark Streaming 是非常出色的实时流处理框架，要掌握其 DStream、transformation 和 checkpoint 等；
2. Spark 的离线统计分析功能，Spark 1.0.0 版本在 Shark 的基础上推出了 Spark SQL，离线统计分析的功能的效率有显著的提升，需要重点掌握；
3. 对于 Spark 的机器学习和 GraphX 等要掌握其原理和用法；

推荐课程：[“Spark 企业级开发最佳实践”](#)

第五阶段：做商业级别的 Spark 项目

通过一个完整的具有代表性的 Spark 项目来贯穿 Spark 的方方面面，包括项目的架构设计、用到的技术的剖析、开发实现、运维等，完整掌握其中的每一个阶段和细节，这样就可以让您以后可以从容面对绝大多数 Spark 项目。

推荐课程：[“Spark 架构案例鉴赏：Conviva、Yahoo !、优酷土豆、网易、腾讯、淘宝等公司的实际 Spark 案例”](#)

第六阶段：提供 Spark 解决方案

1. 彻底掌握 Spark 框架源码的每一个细节；
2. 根据不同的业务场景的需要提供 Spark 在不同场景的下的解决方案；
3. 根据实际需要，在 Spark 框架基础上进行二次开发，打造自己的 Spark 框架；

推荐课程：[“精通 Spark：Spark 内核剖析、源码解读、性能优化和商业案例实战”](#)

《第一章：构建 Spark 集群》

对于 90%以上想学习 Spark 的人而言，如何构建 Spark 集群是其最大的难点之一，为了解决大家构建 Spark 集群的一切困难，家林把 Spark 集群的构建分为了四个步骤，从零起步，不需要任何前置知识，涵盖操作的每一个细节，构建完整的 Spark 集群。

从零起步，构建 Spark 集群经典四部曲：

- 第一步：搭建 Hadoop 单机和伪分布式环境；
- 第二步：构造分布式 Hadoop 集群；
- 第三步：构造分布式的 Spark 集群；
- 第四步：测试 Spark 集群；

不需任何前置知识，从零开始，循序渐进，成为 Spark 高手！



在第 1 讲中我们已经完成了第一步。

这一讲我们构建真正的 Hadoop 分布式集群环境：

- 在 VMWare 中准备第二、第三台运行 Ubuntu 系统的机器；
- 按照配置伪分布式模式的方式配置新创建运行 Ubuntu 系统的机器；
- 配置 Hadoop 分布式集群环境；
- 测试 Hadoop 分布式集群环境；

PS:为何搭建 Hadoop 分布式集群环境只用三台机器呢？原因如下：

- 三台机器可以让人人皆可成功配置 Hadoop 集群运行环境，不会因为现有的机器内存或者磁盘空间而不能够搭建集群环境，毕竟对于初学者来说，最重要的是先要让集群运行起来！
- 三台机器的集群环境配置的步骤和多台机器完全一致。
- 我们的教程是按照循序渐进的方式进行，后面会有三台机器和更多机器的集群运行环境。
- 现在开始 Hadoop 分布式集群环境搭建之旅：

目录

1. 在 VMWare 中准备第二、第三台运行 Ubuntu 系统的机器； 9
2. 按照配置伪分布式模式的方式配置新创建运行 Ubuntu 系统的机器； 9
3. 配置 Hadoop 分布式集群环境10
4. 测试 Hadoop 分布式集群环境；26

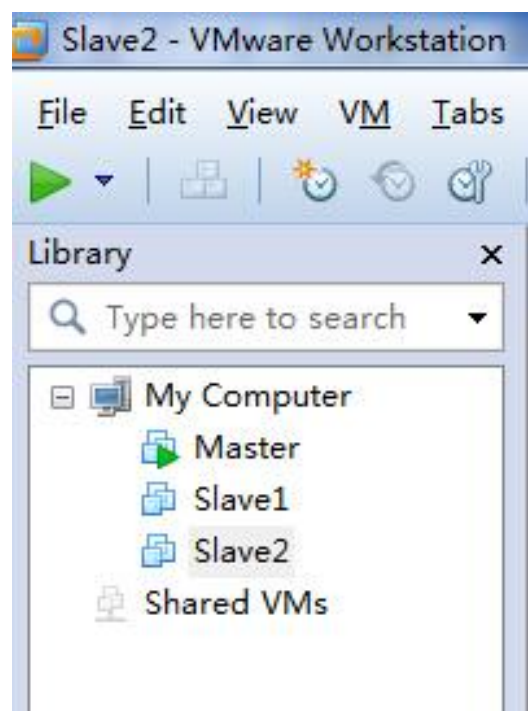
1. 在 VMWare 中准备第二、第三台运行 Ubuntu 系统的机器；

在 VMWare 中构建第二、三台运行 Ubuntu 的机器和构建第一台机器完全一样，再次不在赘述。。

与安装第一台 Ubuntu 机器不同的几点是：

第一点：我们把第二、三台 Ubuntu 机器命名为了 Slave1、Slave2，如下图所示：

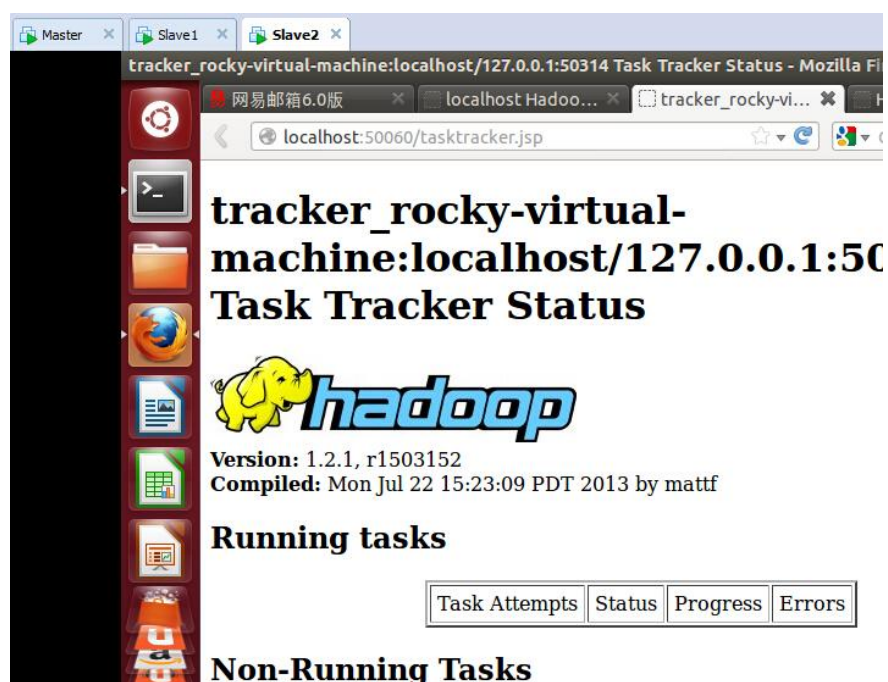
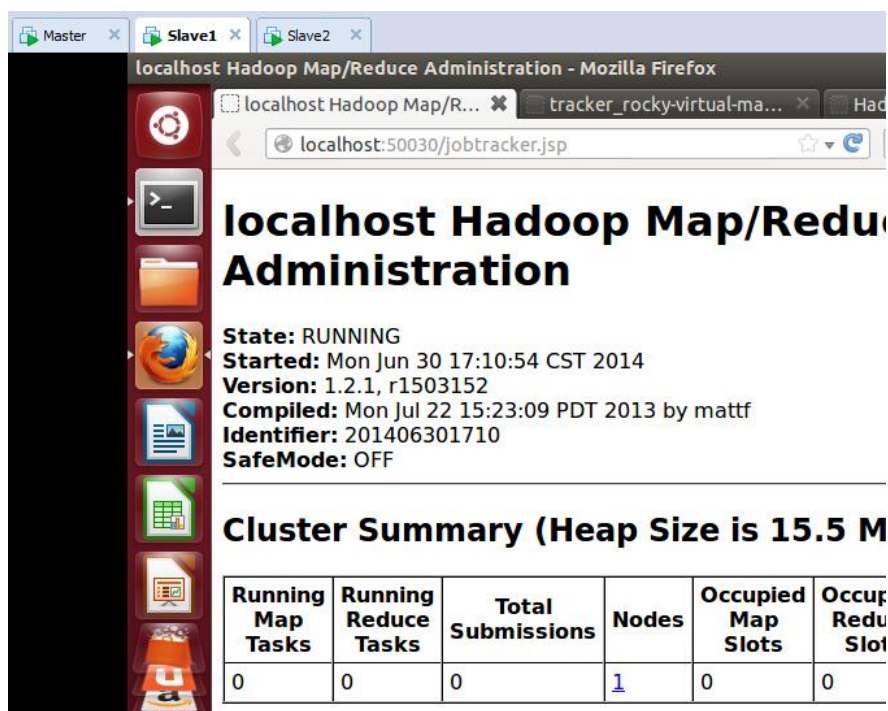
创建完的 VMware 中就有三台虚拟机了：



第二点：为了简化 Hadoop 的配置，保持最小化的 Hadoop 集群，在构建第二、三台机器的时候使用相同的 root 超级用户的方式登录系统。

2. 按照配置伪分布式模式的方式配置新创建运行 Ubuntu 系统的机器；

按照配置伪分布式模式的方式配置新创建运行 Ubuntu 系统的机器和配置第一台机器完全相同，下图是家林完全安装好后的截图：



3. 配置 Hadoop 分布式集群环境；

根据前面的配置，我们现在已经有三台运行在 VMware 中装有 Ubuntu 系统的机器，分别是：Master、Slave1、Slave2；

下面开始配置 Hadoop 分布式集群环境：

Step 1 :在/etc/hostname 中修改主机名并在/etc/hosts 中配置主机名和 IP 地址的对应关系：

我们把 Master 这台机器作为 Hadoop 的主节点，首先看一下 Master 这台机器的 IP 地址：

```
root@rocky-virtual-machine: /
root@rocky-virtual-machine:/usr/local/hadoop/hadoop-1.2.1# cd /
root@rocky-virtual-machine:/# ifconfig
eth0      Link encap:Ethernet  HWaddr 00:0c:29:44:ea:bb
          inet addr:192.168.184.133  Bcast:192.168.184.255  Mask:255.255.255.0
          inet6 addr: fe80::20c:29ff:fe44:eabb/64 Scope:Link
          UP BROADCAST RUNNING MULTICAST  MTU:1500  Metric:1
          RX packets:3209 errors:0 dropped:0 overruns:0 frame:0
          TX packets:2445 errors:0 dropped:0 overruns:0 carrier:0
          collisions:0 txqueuelen:1000
          RX bytes:2217475 (2.2 MB)  TX bytes:407523 (407.5 KB)
          Interrupt:19 Base address:0x2000

lo        Link encap:Local Loopback
          inet addr:127.0.0.1  Mask:255.0.0.0
          inet6 addr: ::1/128 Scope:Host
          UP LOOPBACK RUNNING  MTU:16436  Metric:1
          RX packets:70100 errors:0 dropped:0 overruns:0 frame:0
          TX packets:70100 errors:0 dropped:0 overruns:0 carrier:0
          collisions:0 txqueuelen:0
          RX bytes:13741725 (13.7 MB)  TX bytes:13741725 (13.7 MB)

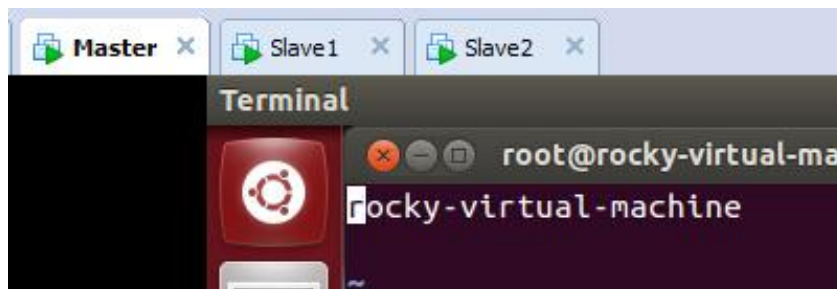
root@rocky-virtual-machine:/#
```

可以看到当前主机的 ip 地址是 “192.168.184.133” 。

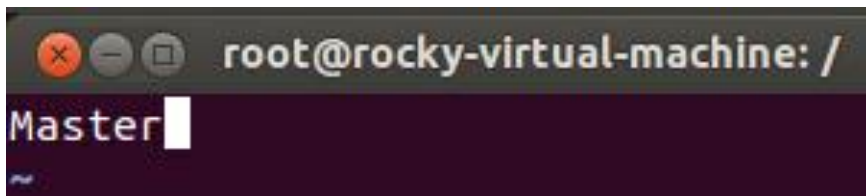
我们在/etc/hostname 中修改主机名：

```
root@rocky-virtual-machine:/# vim /etc/hostname
```

进入配置文件：

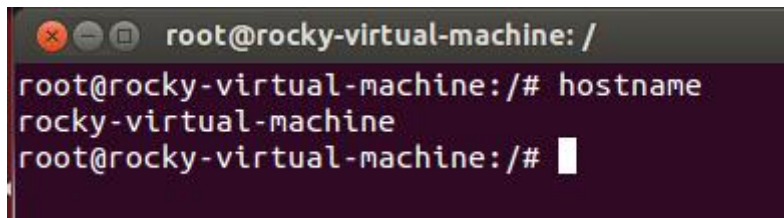


可以看到按照我们装 Ubuntu 系统时候的默认名称，配置文件中的机器的名称是 “rocky-virtual-machine”，我们把 “rocky-virtual-machine” 改为 “Master” 作为 Hadoop 分布式集群环境的主节点：



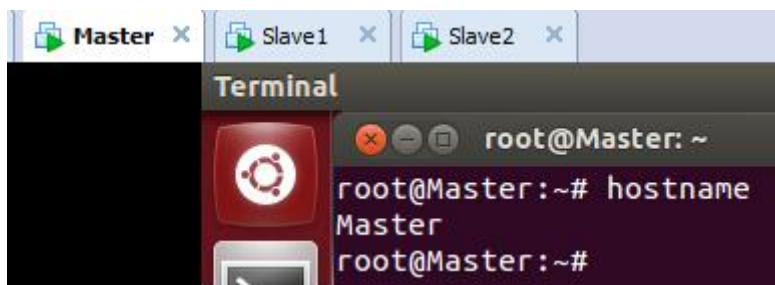
```
root@rocky-virtual-machine: /
Master
```

保存退出。此时使用以下命令查看当前主机的主机名：



```
root@rocky-virtual-machine: /
root@rocky-virtual-machine:/# hostname
rocky-virtual-machine
root@rocky-virtual-machine:/#
```

发现修改的主机名没有生效，为使得新修改的主机名生效，我们重新启动系统后再次查看主机名：



```
Master x Slave1 x Slave2 x
Terminal
root@Master: ~
root@Master:~# hostname
Master
root@Master:~#
```

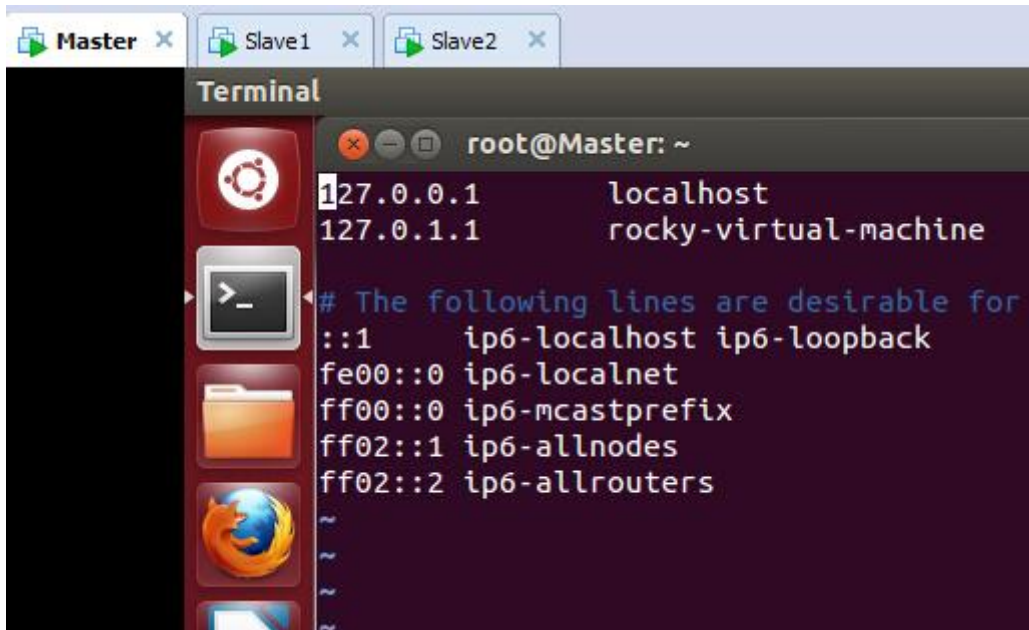
发现我们的主机名成为了修改后的“Master”，表明修改成功。

打开在/etc/hosts 文件：

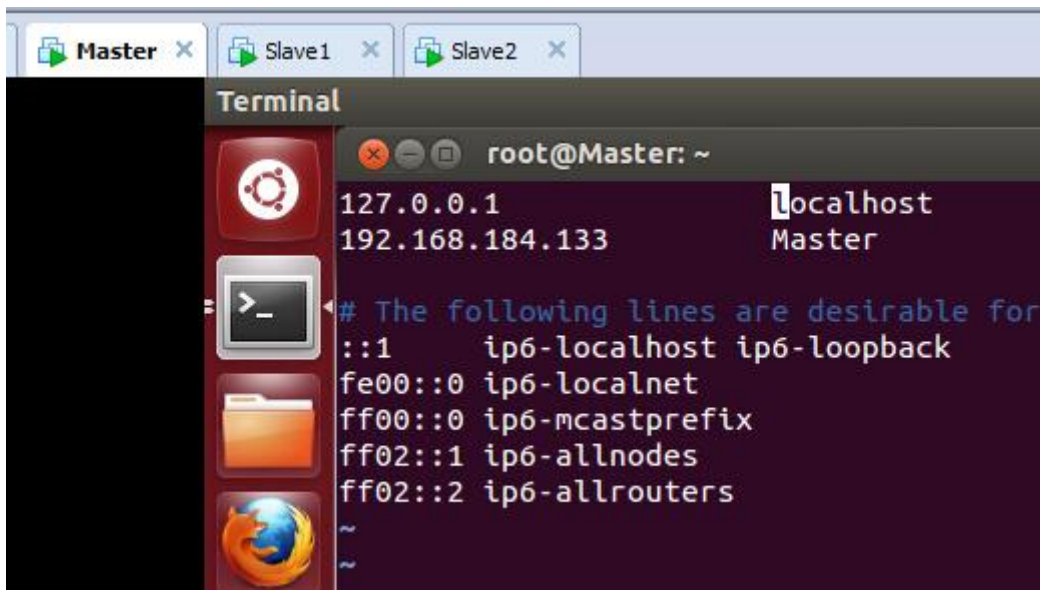


```
root@Master:~# vim /etc/hosts
```

此时我们发现文件中只有 Ubuntu 系统的原始 ip(127.0.0.1)地址和主机名(localhost)的对应关系：

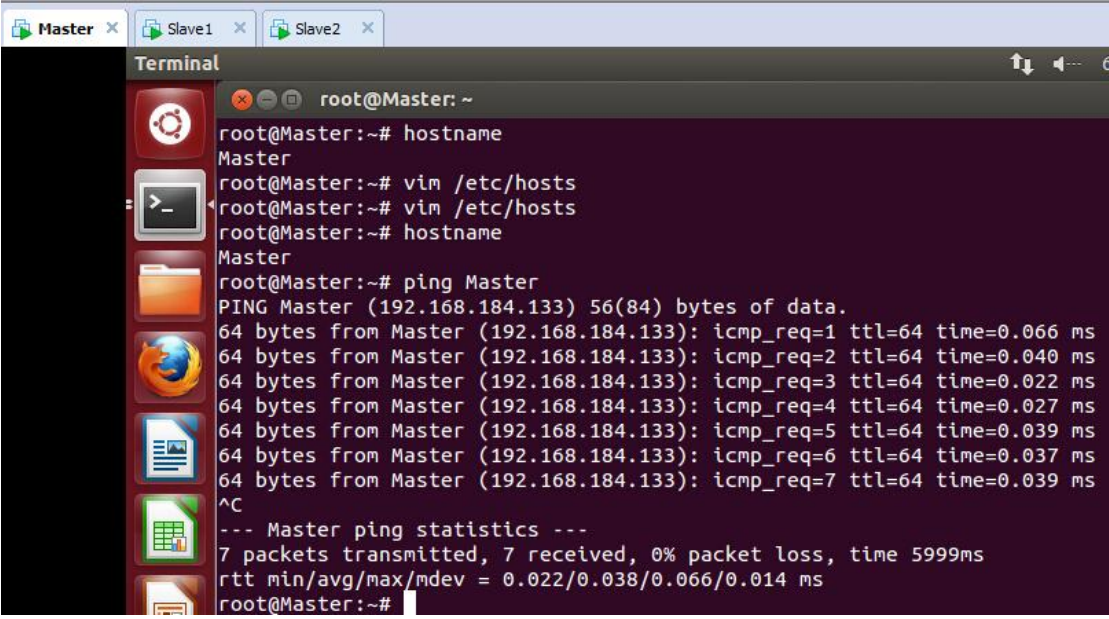


我们在/etc/hosts 中配置主机名和 IP 地址的对应关系：



修改之后保存退出。

接下来我们使用 “ping” 命令看一下主机名和 IP 地址只见的转换关系是否正确：



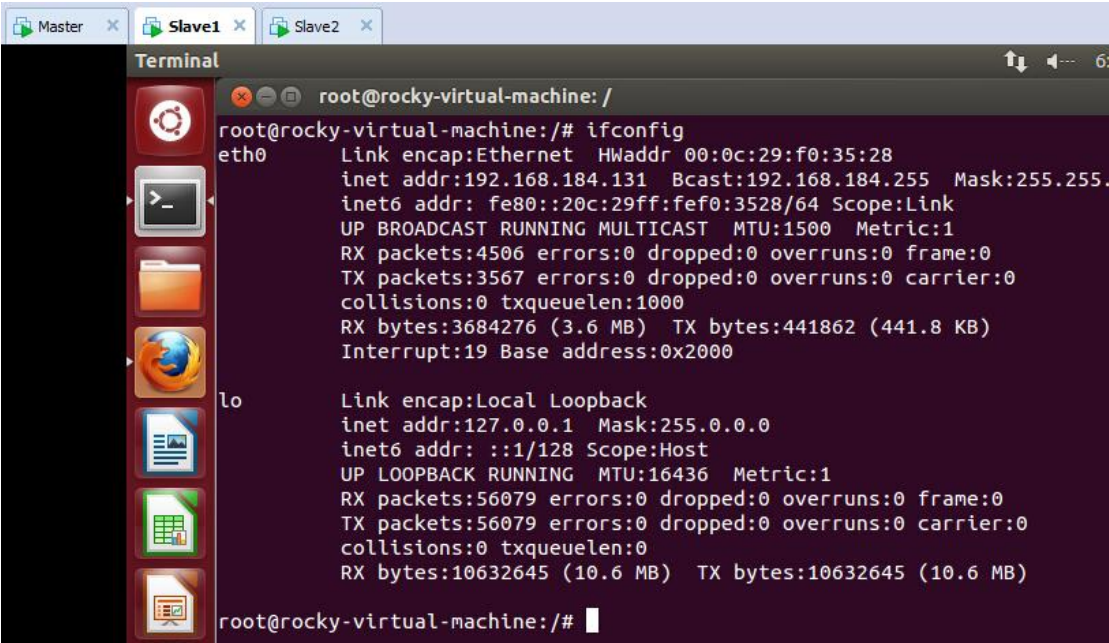
```

root@Master:~# hostname
Master
root@Master:~# vim /etc/hosts
root@Master:~# vim /etc/hosts
root@Master:~# hostname
Master
root@Master:~# ping Master
PING Master (192.168.184.133) 56(84) bytes of data.
64 bytes from Master (192.168.184.133): icmp_req=1 ttl=64 time=0.066 ms
64 bytes from Master (192.168.184.133): icmp_req=2 ttl=64 time=0.040 ms
64 bytes from Master (192.168.184.133): icmp_req=3 ttl=64 time=0.022 ms
64 bytes from Master (192.168.184.133): icmp_req=4 ttl=64 time=0.027 ms
64 bytes from Master (192.168.184.133): icmp_req=5 ttl=64 time=0.039 ms
64 bytes from Master (192.168.184.133): icmp_req=6 ttl=64 time=0.037 ms
64 bytes from Master (192.168.184.133): icmp_req=7 ttl=64 time=0.039 ms
^C
--- Master ping statistics ---
7 packets transmitted, 7 received, 0% packet loss, time 5999ms
rtt min/avg/max/mdev = 0.022/0.038/0.066/0.014 ms
root@Master:~#

```

可以看到此时我们的主机“Master”对应的IP地址是“192.168.184.133”，这表明我们的配置和运行都是正确的。

进入第二台机器，看一下这台主机的IP地址：



```

root@rocky-virtual-machine: /
root@rocky-virtual-machine:~# ifconfig
eth0      Link encap:Ethernet  HWaddr 00:0c:29:f0:35:28
          inet addr:192.168.184.131  Bcast:192.168.184.255  Mask:255.255.255
          inet6 addr: fe80::20c:29ff:fef0:3528/64 Scope:Link
          UP BROADCAST RUNNING MULTICAST  MTU:1500  Metric:1
          RX packets:4506 errors:0 dropped:0 overruns:0 frame:0
          TX packets:3567 errors:0 dropped:0 overruns:0 carrier:0
          collisions:0 txqueuelen:1000
          RX bytes:3684276 (3.6 MB)  TX bytes:441862 (441.8 KB)
          Interrupt:19 Base address:0x2000

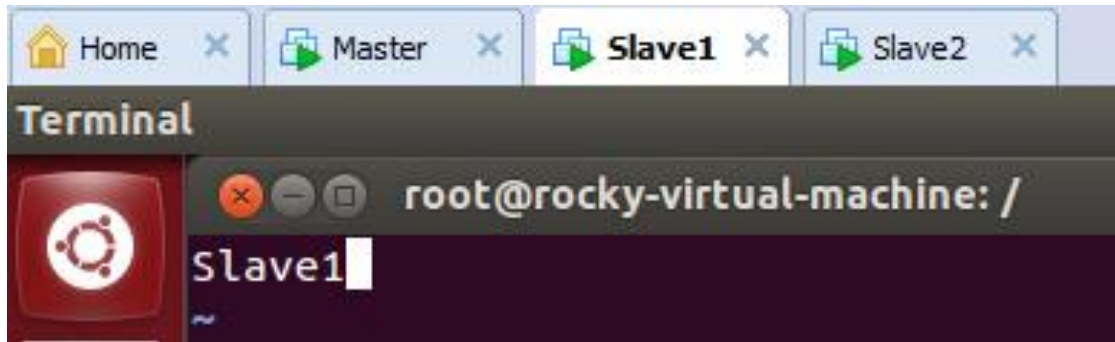
lo        Link encap:Local Loopback
          inet addr:127.0.0.1  Mask:255.0.0.0
          inet6 addr: ::1/128 Scope:Host
          UP LOOPBACK RUNNING  MTU:16436  Metric:1
          RX packets:56079 errors:0 dropped:0 overruns:0 frame:0
          TX packets:56079 errors:0 dropped:0 overruns:0 carrier:0
          collisions:0 txqueuelen:0
          RX bytes:10632645 (10.6 MB)  TX bytes:10632645 (10.6 MB)

root@rocky-virtual-machine:~#

```

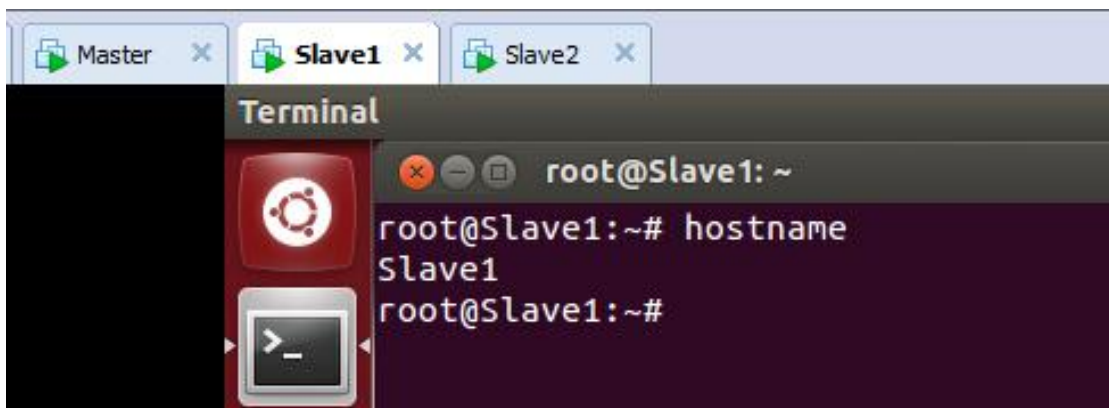
可以看出这台主机的IP地址是“192.168.184.131”。

我们在/etc/hostname 中把主机名称修改为“Slave1”：



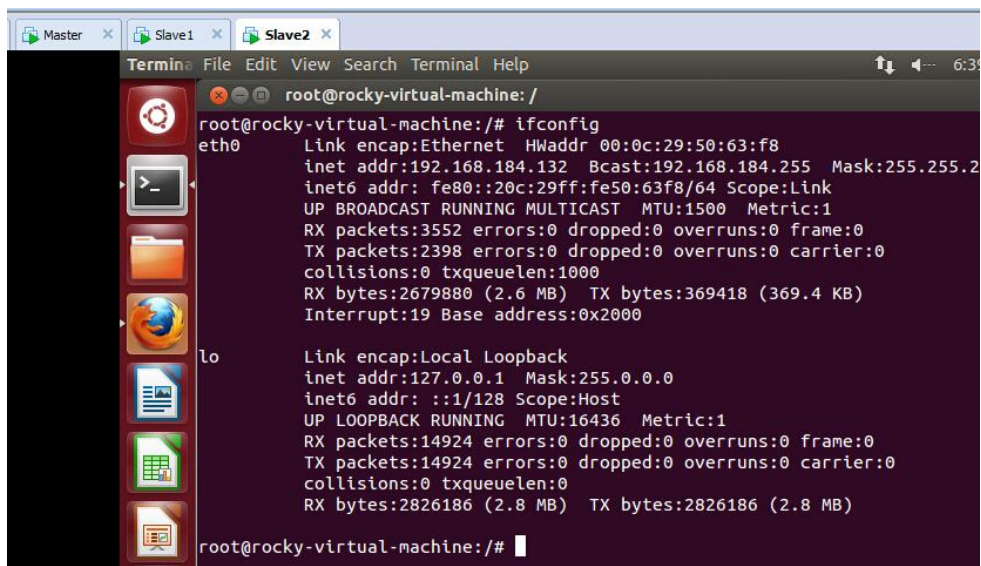
保存退出。

为了使修改生效，我们重新启动该机器，此时查看主机名：



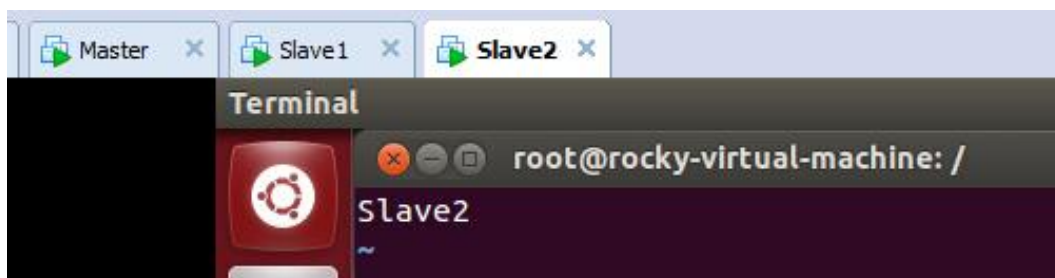
表明我们的修改生效了。

进入第三台机器，看一下这台主机的 IP 地址：



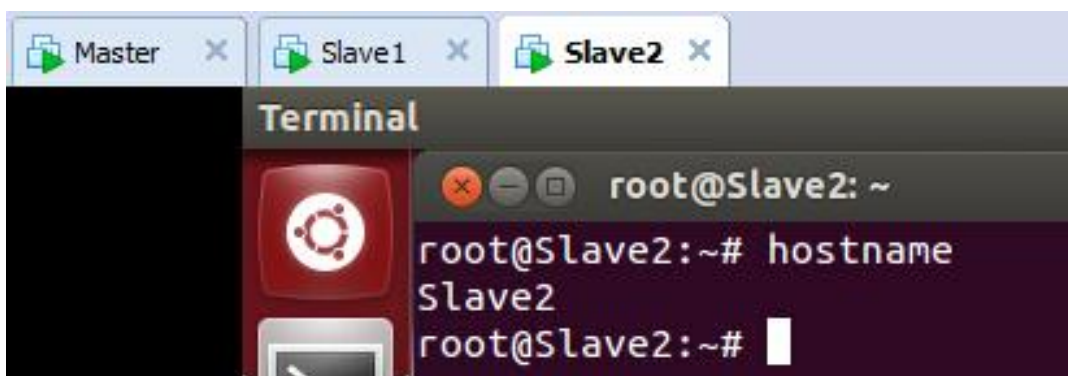
可以看出这台主机的 IP 地址是 "192.168.184.132"。

我们在/etc/hostname 中把主机名称修改为 "Slave2"：



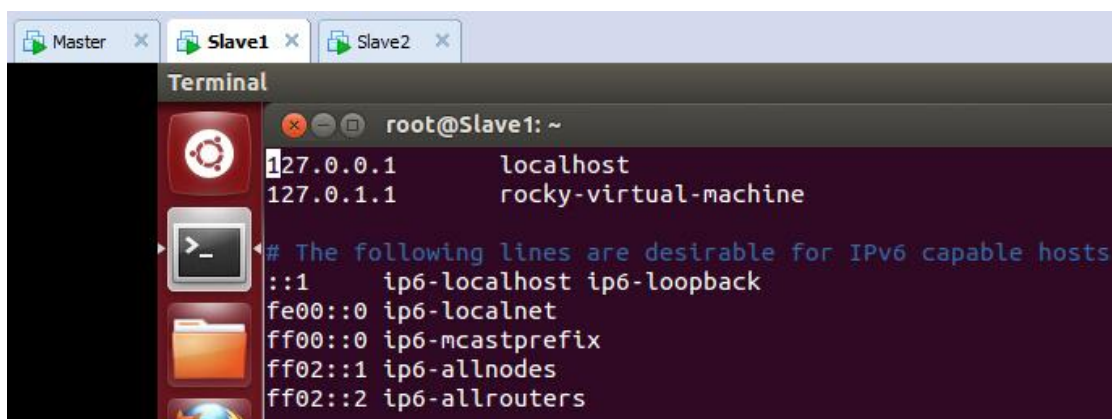
保存退出。

为了使修改生效，我们重新启动该机器，此时查看主机名：

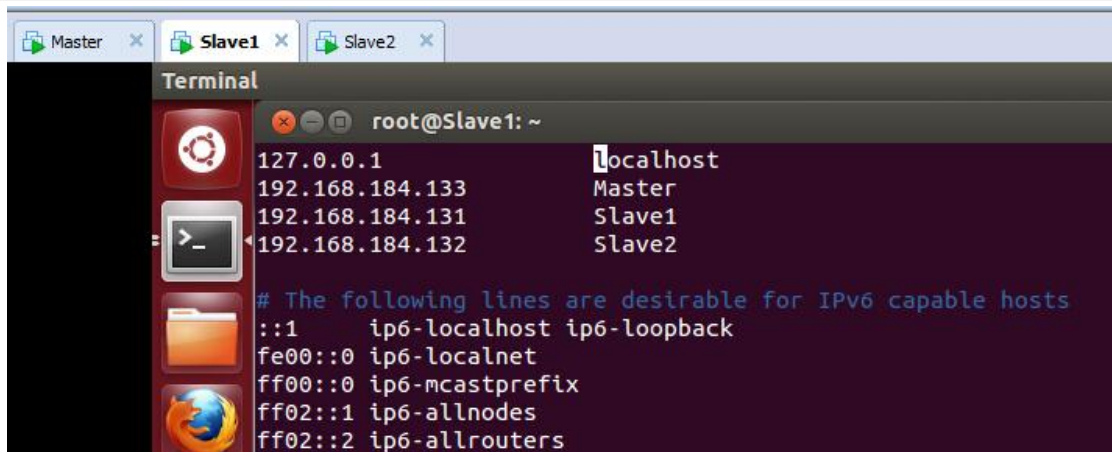


表明我们的修改生效了。

现在，Slave1 上的/etc/hosts 中配置主机名和 IP 地址的对应关系，打开后：

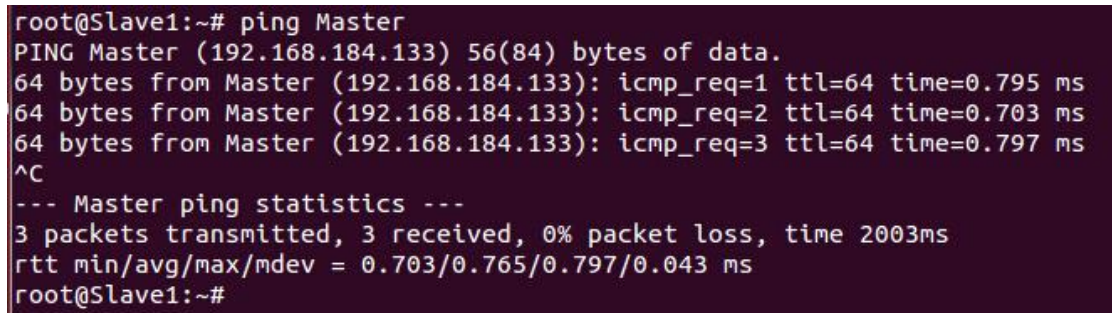


此时我们修改为配置文件为：

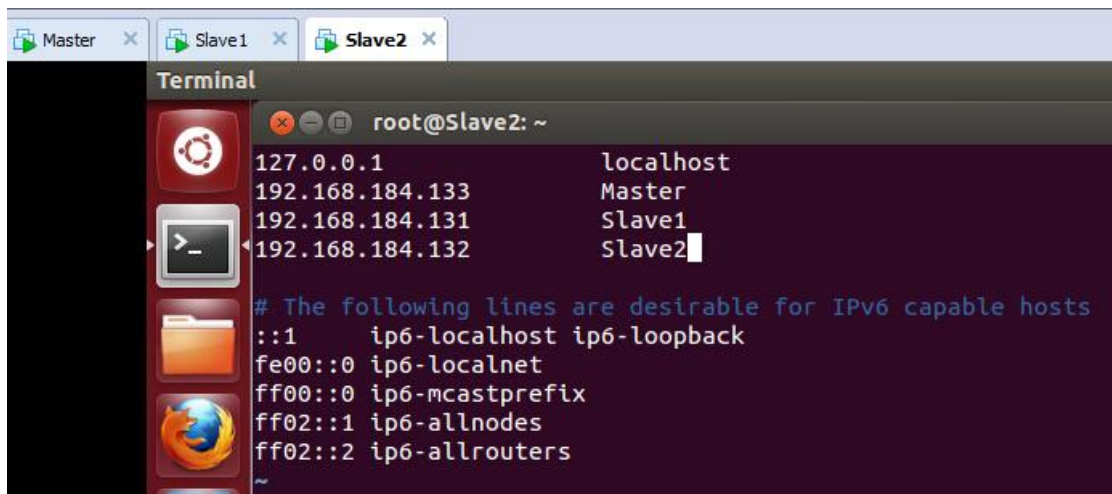


把“Master”和“Slave1”和“Slave2”的主机名和IP地址的对应关系都配置进去。保存退出。

我们此时 ping 一下 Master 这个节点发现网络访问没有问题：



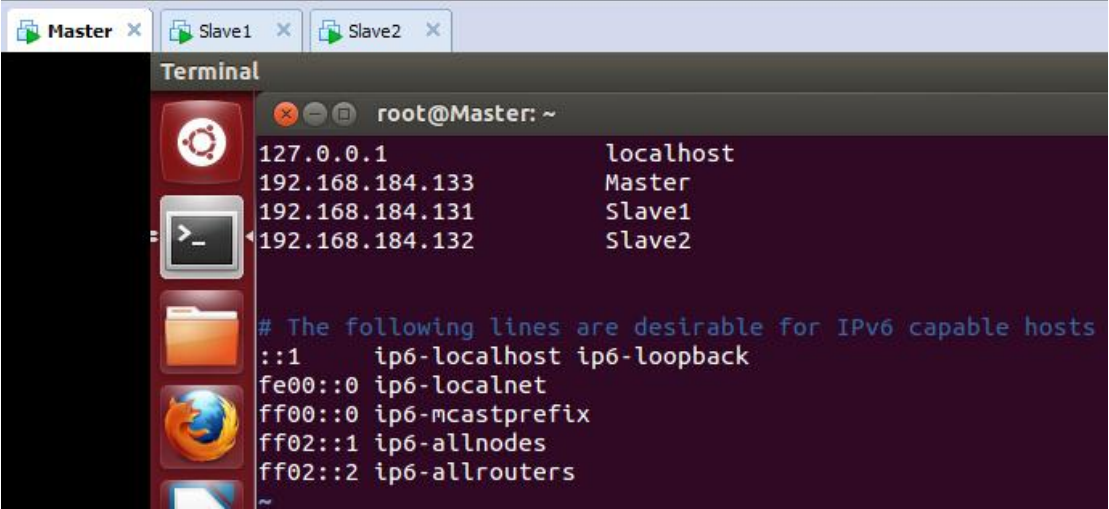
接着，在 Slave2 上的/etc/hosts 中配置主机名和IP地址的对应关系，配置完后如下：



保存退出。

此时我们 ping 一下 Master 和 Slave1 发现都可以 ping 通；

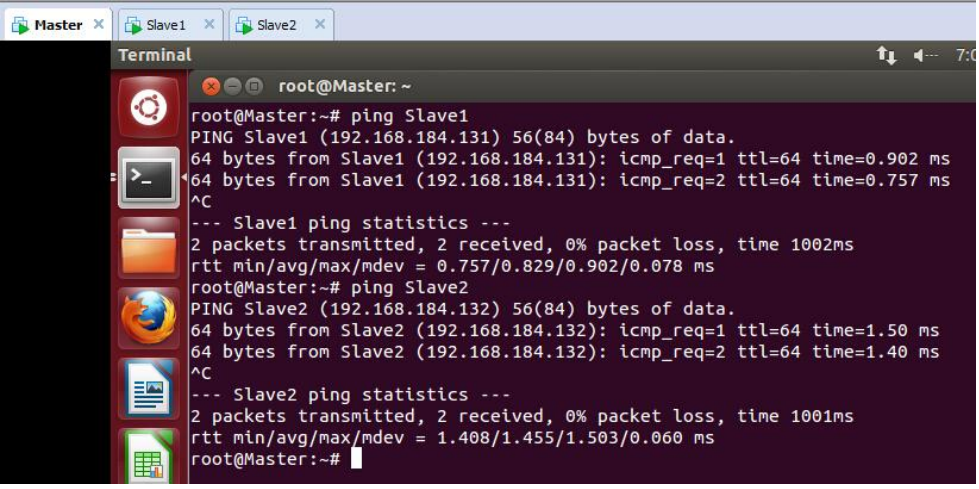
最后把在 Master 上的/etc/hosts 中配置主机名和 IP 地址的对应关系，配置完后如下：



```
Master x Slave1 x Slave2 x
Terminal
root@Master: ~
127.0.0.1      localhost
192.168.184.133 Master
192.168.184.131 Slave1
192.168.184.132 Slave2

# The following lines are desirable for IPv6 capable hosts
::1          ip6-localhost ip6-loopback
fe00::0      ip6-localnet
ff00::0      ip6-mcastprefix
ff02::1      ip6-allnodes
ff02::2      ip6-allrouters
```

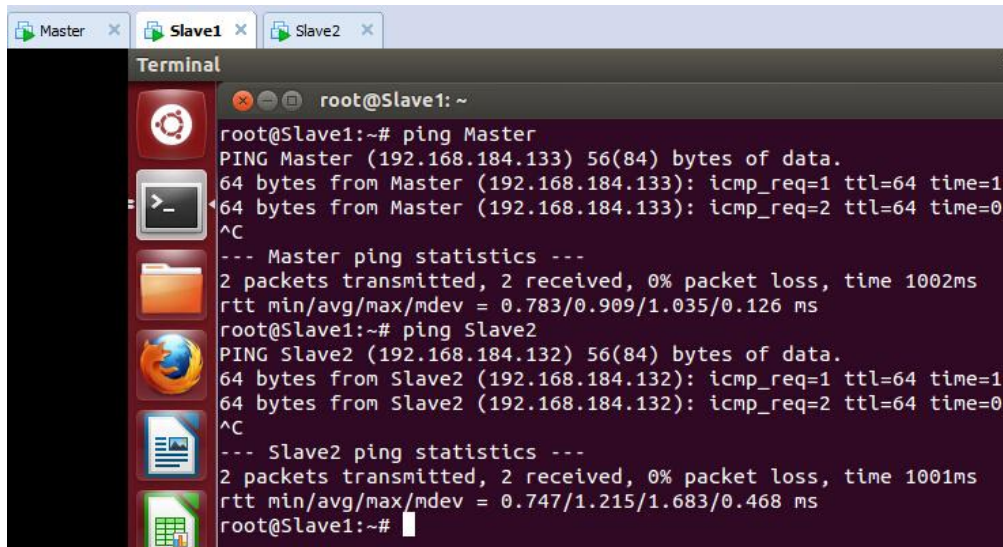
此时在 Master 上使用 ping 命令和 Slave1 和 Slave2 这两台机器进行沟通：



```
Master x Slave1 x Slave2 x
Terminal
root@Master: ~
root@Master:~# ping Slave1
PING Slave1 (192.168.184.131) 56(84) bytes of data.
64 bytes from Slave1 (192.168.184.131): icmp_req=1 ttl=64 time=0.902 ms
64 bytes from Slave1 (192.168.184.131): icmp_req=2 ttl=64 time=0.757 ms
^C
--- Slave1 ping statistics ---
2 packets transmitted, 2 received, 0% packet loss, time 1002ms
rtt min/avg/max/mdev = 0.757/0.829/0.902/0.078 ms
root@Master:~# ping Slave2
PING Slave2 (192.168.184.132) 56(84) bytes of data.
64 bytes from Slave2 (192.168.184.132): icmp_req=1 ttl=64 time=1.50 ms
64 bytes from Slave2 (192.168.184.132): icmp_req=2 ttl=64 time=1.40 ms
^C
--- Slave2 ping statistics ---
2 packets transmitted, 2 received, 0% packet loss, time 1001ms
rtt min/avg/max/mdev = 1.408/1.455/1.503/0.060 ms
root@Master:~#
```

发现此时已经 ping 通了两个 slave 节点的机器。

最后我们在测试一下 Slave1 这台机器和 Master、Slave2 的通信：

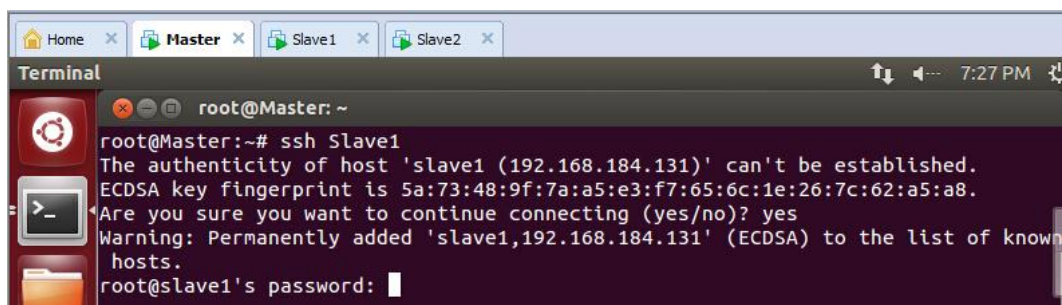


```
Master x Slave1 x Slave2 x
Terminal
root@Slave1: ~
root@Slave1:~# ping Master
PING Master (192.168.184.133) 56(84) bytes of data.
64 bytes from Master (192.168.184.133): icmp_req=1 ttl=64 time=1.002ms
64 bytes from Master (192.168.184.133): icmp_req=2 ttl=64 time=0.783ms
^C
--- Master ping statistics ---
2 packets transmitted, 2 received, 0% packet loss, time 1002ms
rtt min/avg/max/mdev = 0.783/0.909/1.035/0.126 ms
root@Slave1:~# ping Slave2
PING Slave2 (192.168.184.132) 56(84) bytes of data.
64 bytes from Slave2 (192.168.184.132): icmp_req=1 ttl=64 time=1.001ms
64 bytes from Slave2 (192.168.184.132): icmp_req=2 ttl=64 time=0.747ms
^C
--- Slave2 ping statistics ---
2 packets transmitted, 2 received, 0% packet loss, time 1001ms
rtt min/avg/max/mdev = 0.747/1.215/1.683/0.468 ms
root@Slave1:~#
```

到目前为止，Master、Slave1、Slave2 这三台机器之间实现了相互通信！

Step 2：SSH 无密码验证配置

首先我们看一下在没有配置的情况下 Master 通过 SSH 协议访问 Slave1 的情况：



```
Home x Master x Slave1 x Slave2 x
Terminal
root@Master: ~
root@Master:~# ssh Slave1
The authenticity of host 'slave1 (192.168.184.131)' can't be established.
ECDSA key fingerprint is 5a:73:48:9f:7a:a5:e3:f7:65:6c:1e:26:7c:62:a5:a8.
Are you sure you want to continue connecting (yes/no)? yes
Warning: Permanently added 'slave1,192.168.184.131' (ECDSA) to the list of known hosts.
root@slave1's password: 
```

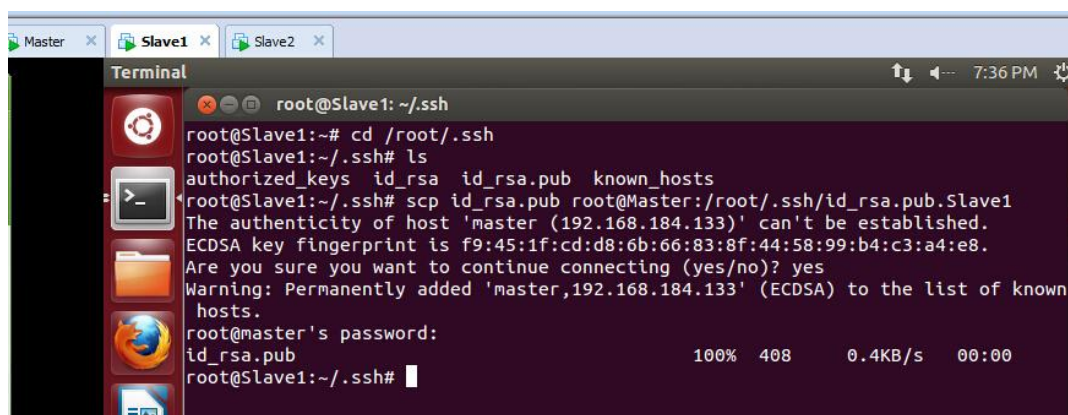
此时会发现我们是需要密码的。我们不登陆进去，直接退出。

怎么使得集群能够通过 SSH 免登陆密码呢？

按照前面的配置，我们已经分布在 Master、Slave1、Slave2 这三台机器上的/root/.ssh/

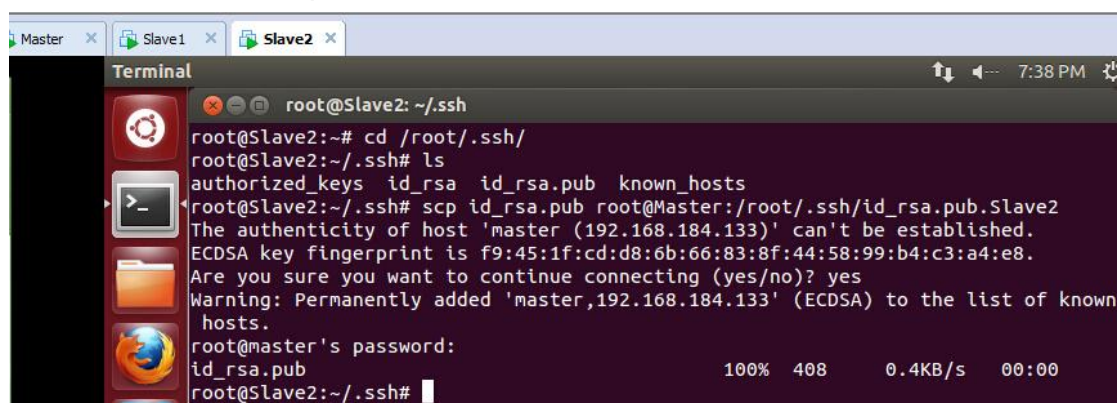
目录下生成一个私钥 id_rsa 和一个公钥 id_rsa.pub。

此时把 Slave1 的 id_rsa.pub 传给 Master，如下所示：

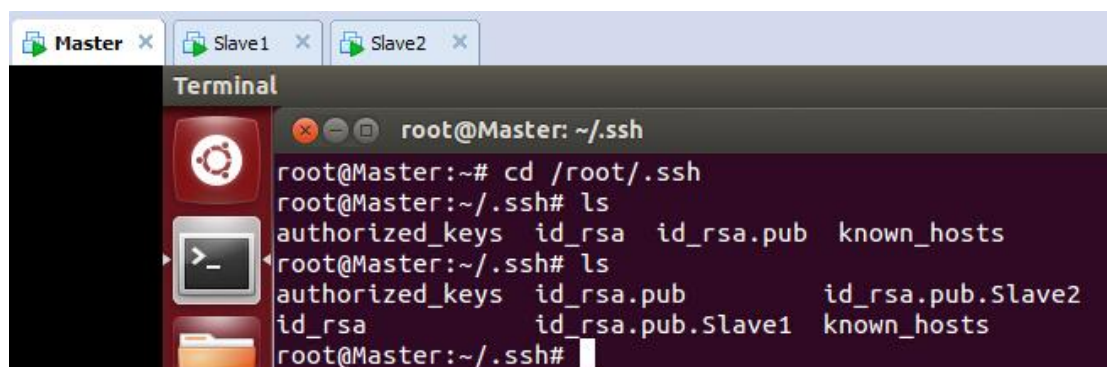


```
Master x Slave1 x Slave2 x
Terminal
root@Slave1: ~/.ssh
root@Slave1:~# cd /root/.ssh
root@Slave1:~/.ssh# ls
authorized_keys id_rsa id_rsa.pub known_hosts
root@Slave1:~/.ssh# scp id_rsa.pub root@Master:/root/.ssh/id_rsa.pub.Slave1
The authenticity of host 'master (192.168.184.133)' can't be established.
ECDSA key fingerprint is f9:45:1f:cd:d8:6b:66:83:8f:44:58:99:b4:c3:a4:e8.
Are you sure you want to continue connecting (yes/no)? yes
Warning: Permanently added 'master,192.168.184.133' (ECDSA) to the list of known
hosts.
root@master's password:
id_rsa.pub                                100% 408      0.4KB/s   00:00
root@Slave1:~/.ssh#
```

同时把 Slave2 的 id_rsa.pub 传给 Master , 如下所示 :



```
Master x Slave1 x Slave2 x
Terminal
root@Slave2: ~/.ssh
root@Slave2:~# cd /root/.ssh/
root@Slave2:~/.ssh# ls
authorized_keys id_rsa id_rsa.pub known_hosts
root@Slave2:~/.ssh# scp id_rsa.pub root@Master:/root/.ssh/id_rsa.pub.Slave2
The authenticity of host 'master (192.168.184.133)' can't be established.
ECDSA key fingerprint is f9:45:1f:cd:d8:6b:66:83:8f:44:58:99:b4:c3:a4:e8.
Are you sure you want to continue connecting (yes/no)? yes
Warning: Permanently added 'master,192.168.184.133' (ECDSA) to the list of known
hosts.
root@master's password:
id_rsa.pub                                100% 408      0.4KB/s   00:00
root@Slave2:~/.ssh#
```



```
Master x Slave1 x Slave2 x
Terminal
root@Master: ~/.ssh
root@Master:~# cd /root/.ssh
root@Master:~/.ssh# ls
authorized_keys id_rsa id_rsa.pub known_hosts
root@Master:~/.ssh# ls
authorized_keys id_rsa.pub id_rsa.pub.Slave2
id_rsa id_rsa.pub.Slave1 known_hosts
root@Master:~/.ssh#
```

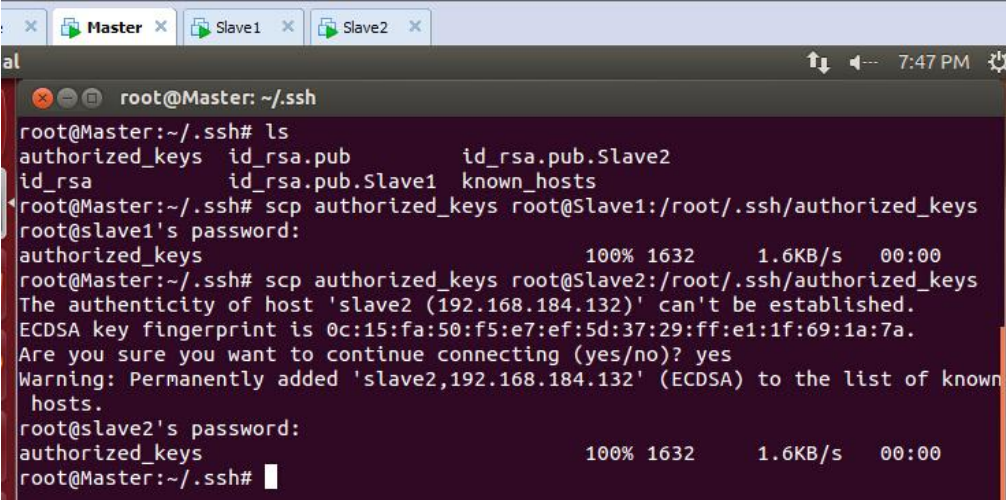
在 Master 上检查一下是否复制了过来 :

此时我们发现 Slave1 和 Slave2 节点的公钥已经传输过来

Master 节点上综合所有公钥 :

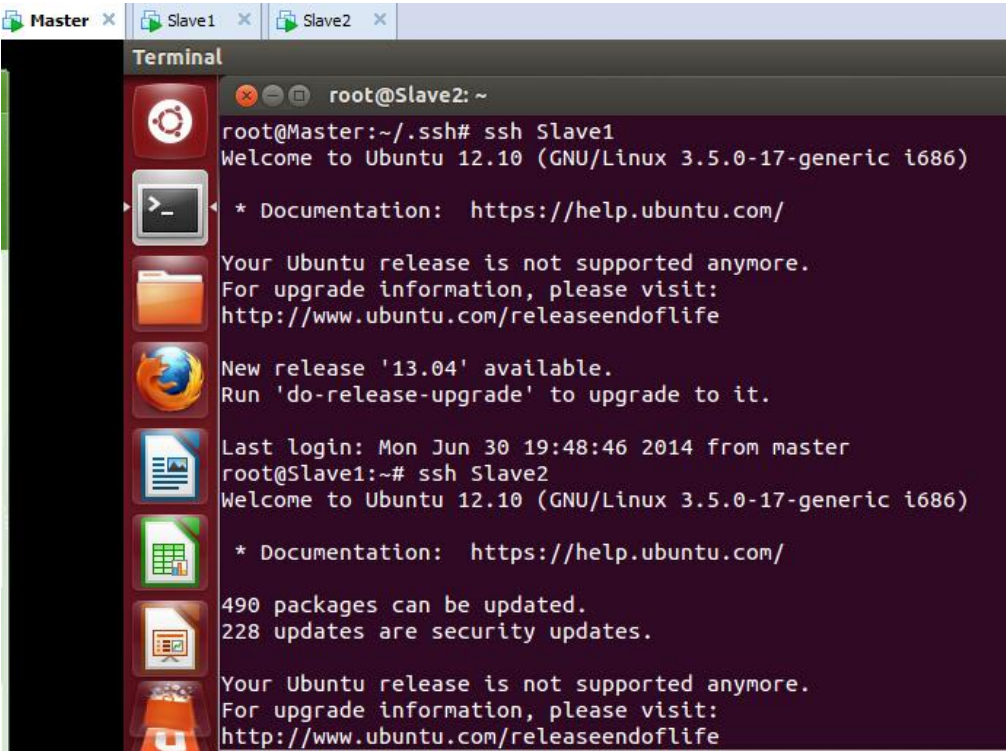
```
root@Master:~/.ssh# cat id_rsa.pub >> authorized_keys
root@Master:~/.ssh# cat id_rsa.pub.Slave1 >> authorized_keys
root@Master:~/.ssh# cat id_rsa.pub.Slave2 >> authorized_keys
```

将 Master 的公钥信息 authorized_keys 复制到 Slave1 和 Slave2 的.ssh 目录下：



```
root@Master: ~/.ssh
root@Master:~/.ssh# ls
authorized_keys  id_rsa.pub      id_rsa.pub.Slave2
id_rsa          id_rsa.pub.Slave1  known_hosts
root@Master:~/.ssh# scp authorized_keys root@Slave1:/root/.ssh/authorized_keys
root@slave1's password:
authorized_keys                                100% 1632    1.6KB/s   00:00
root@Master:~/.ssh# scp authorized_keys root@Slave2:/root/.ssh/authorized_keys
The authenticity of host 'slave2 (192.168.184.132)' can't be established.
ECDSA key fingerprint is 0c:15:fa:50:f5:e7:ef:5d:37:29:ff:e1:1f:69:1a:7a.
Are you sure you want to continue connecting (yes/no)? yes
Warning: Permanently added 'slave2,192.168.184.132' (ECDSA) to the list of known
hosts.
root@slave2's password:
authorized_keys                                100% 1632    1.6KB/s   00:00
root@Master:~/.ssh#
```

此时再次通过 SSH 登录 Slave1 和 Slave2：



```
root@Master: ~
root@Master:~/.ssh# ssh Slave1
Welcome to Ubuntu 12.10 (GNU/Linux 3.5.0-17-generic i686)

* Documentation:  https://help.ubuntu.com/

Your Ubuntu release is not supported anymore.
For upgrade information, please visit:
http://www.ubuntu.com/releaseendoflife

New release '13.04' available.
Run 'do-release-upgrade' to upgrade to it.

Last login: Mon Jun 30 19:48:46 2014 from master
root@Slave1:~# ssh Slave2
Welcome to Ubuntu 12.10 (GNU/Linux 3.5.0-17-generic i686)

* Documentation:  https://help.ubuntu.com/

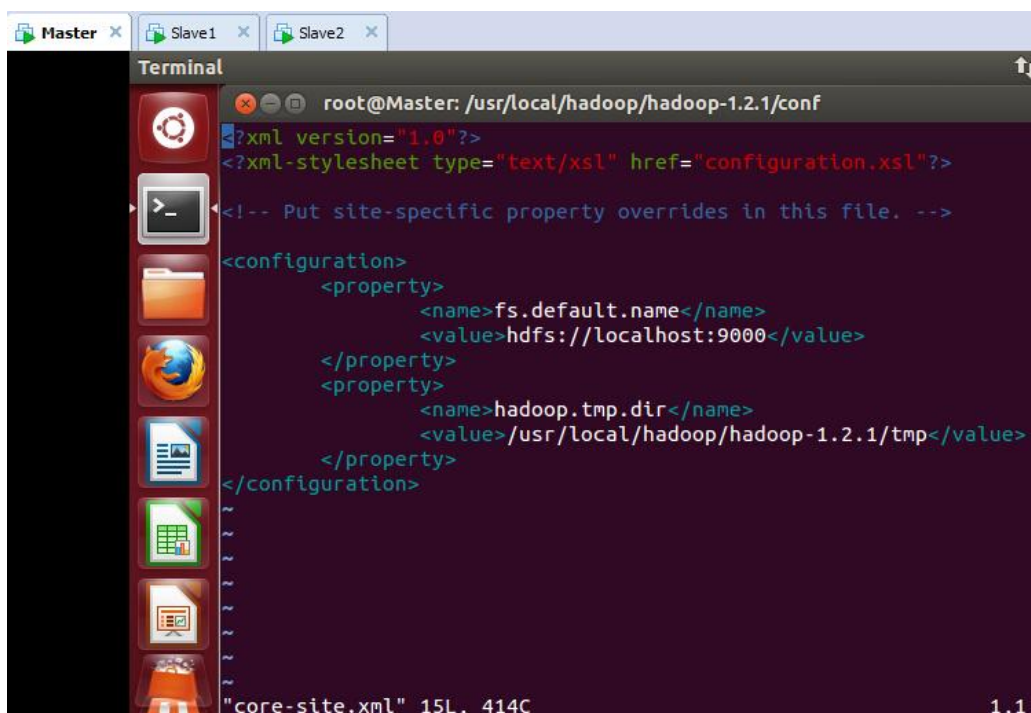
490 packages can be updated.
228 updates are security updates.

Your Ubuntu release is not supported anymore.
For upgrade information, please visit:
http://www.ubuntu.com/releaseendoflife
```

此时 Master 通过 SSH 登录 Slave1 和 Slave2 已经不需要密码，同样的 Slave1 或者 Slave2 通过 SSH 协议登录另外两台机器也不需要密码了。

Step 3：修改 Master、Slave1、Slave2 的配置文件

首先修改 Master 的 core-site.xml 文件，此时的文件内容是：



```

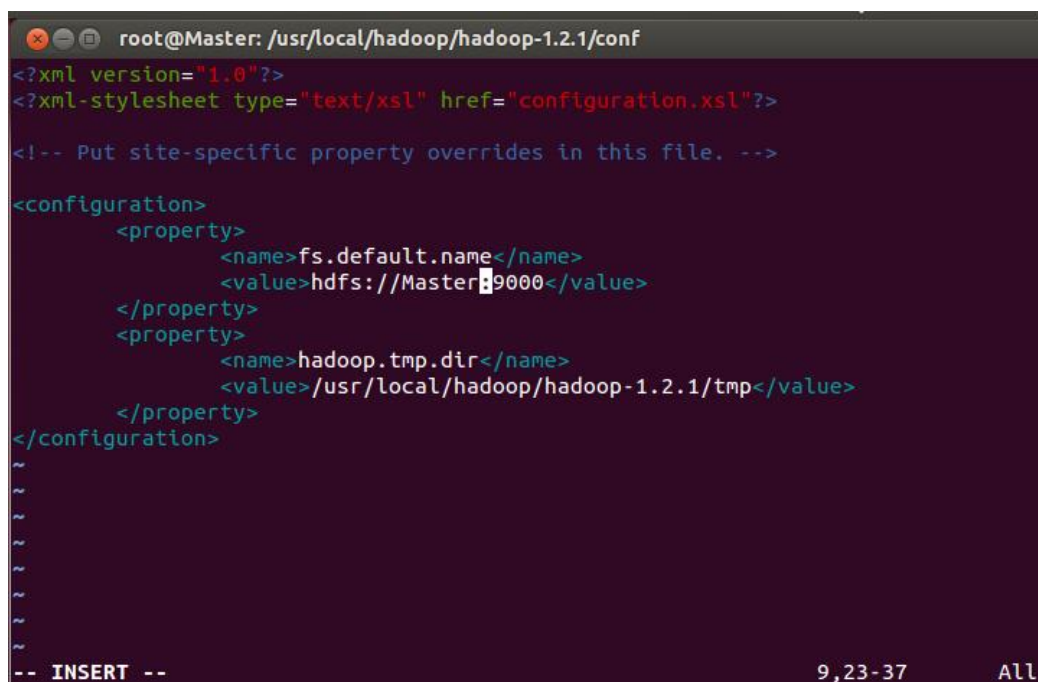
root@Master: /usr/local/hadoop/hadoop-1.2.1/conf
<?xml version="1.0"?>
<?xml-stylesheet type="text/xsl" href="configuration.xsl"?>

<!-- Put site-specific property overrides in this file. -->

<configuration>
  <property>
    <name>fs.default.name</name>
    <value>hdfs://localhost:9000</value>
  </property>
  <property>
    <name>hadoop.tmp.dir</name>
    <value>/usr/local/hadoop/hadoop-1.2.1/tmp</value>
  </property>
</configuration>
~
~
~
~
~
"core-site.xml" 15L, 414C
1,1

```

我们把“localhost”域名修改为“Master”：



```

root@Master: /usr/local/hadoop/hadoop-1.2.1/conf
<?xml version="1.0"?>
<?xml-stylesheet type="text/xsl" href="configuration.xsl"?>

<!-- Put site-specific property overrides in this file. -->

<configuration>
  <property>
    <name>fs.default.name</name>
    <value>hdfs://Master:9000</value>
  </property>
  <property>
    <name>hadoop.tmp.dir</name>
    <value>/usr/local/hadoop/hadoop-1.2.1/tmp</value>
  </property>
</configuration>
~
~
~
~
~
-- INSERT --
9,23-37
ALL

```

同样的操作分别打开 Slave1 和 Slave2 节点 core-site.xml，把“localhost”域名修改为

“Master”。

其次修改 Master、Slave1、Slave2 的 mapred-site.xml 文件。

进入 Master 节点的 mapred-site.xml 文件把 “localhost” 域名修改为 “Master”，保存退出。

同理 打开 Slave1 和 Slave2 节点 mapred-site.xml 把 “localhost” 域名修改为 “Master”，保存退出。

最后修改 Master、Slave1、Slave2 的 hdfs-site.xml 文件：

```
root@Master: /usr/local/hadoop/hadoop-1.2.1/conf
<?xml version="1.0"?>
<?xml-stylesheet type="text/xsl" href="configuration.xsl"?>

<!-- Put site-specific property overrides in this file. -->

<configuration>
  <property>
    <name>dfs.replication</name>
    <value>1</value>
  </property>
  <property>
    <name>dfs.name.dir</name>
    <value>/usr/local/hadoop/hadoop-1.2.1/hdfs/name</value>
  </property>
  <property>
    <name>dfs.data.dir</name>
    <value>/usr/local/hadoop/hadoop-1.2.1/hdfs/data</value>
  </property>
</configuration>

~
~
"hdfs-site.xml" 21L, 595C                                     1,1
```

我们把三台机器上的 “dfs.replication” 值由 1 改为 3，这样我们的数据就会有 3 份副本：

```

root@Master: /usr/local/hadoop/hadoop-1.2.1/conf
<?xml version="1.0"?>
<?xml-stylesheet type="text/xsl" href="configuration.xsl"?>

<!-- Put site-specific property overrides in this file. -->

<configuration>
  <property>
    <name>dfs.replication</name>
    <value>3</value>
  </property>
  <property>
    <name>dfs.name.dir</name>
    <value>/usr/local/hadoop/hadoop-1.2.1/hdfs/name</value>
  </property>
  <property>
    <name>dfs.data.dir</name>
    <value>/usr/local/hadoop/hadoop-1.2.1/hdfs/data</value>
  </property>
</configuration>

~
~
-- INSERT --
9,25 All

```

保存退出。

Step 4 : 修改两台机器中 hadoop 配置文件的 masters 和 slaves 文件

首先修改 Master 的 masters 文件 :

```
root@Master: /usr/local/hadoop/hadoop-1.2.1/conf# vim masters
```

进入文件 :

```

root@Master: /usr/local/hadoop/hadoop-1.2.1/conf
localhost

```

把 "localhost" 改为 "Master" :

```

root@Master: /usr/local/hadoop/hadoop-1.2.1/conf
Master

```

保存退出。

修改 Master 的 slaves 文件，

```
root@Master:/usr/local/hadoop/hadoop-1.2.1/conf# vim slaves
```

进入该文件：

```
root@Master:/usr/local/hadoop/hadoop-1.2.1/conf
localhost
```

具体修改为：

```
Terminal
root@Master:/usr/local/hadoop/hadoop-1.2.1/conf
Master
Slave1
Slave2
```

保存退出。

从上面的配置可以看出我们把 Master 即作为主节点，又作为数据处理节点，这是考虑我们数据的 3 份副本而我们的机器台数有限所致。

把 Master 配置的 masters 和 slaves 文件分别拷贝到 Slave1 和 Slave2 的 Hadoop 安装目录下的 conf 文件夹下：

```
root@Master:/usr/local/hadoop/hadoop-1.2.1/conf# scp masters root@Slave1:/usr/local/hadoop/hadoop-1.2.1/conf
masters                                100%  7    0.0KB/s  00:00
root@Master:/usr/local/hadoop/hadoop-1.2.1/conf# scp slaves root@Slave1:/usr/local/hadoop/hadoop-1.2.1/conf
slaves                                100% 21    0.0KB/s  00:00
root@Master:/usr/local/hadoop/hadoop-1.2.1/conf# scp masters root@Slave2:/usr/local/hadoop/hadoop-1.2.1/conf
masters                                100%  7    0.0KB/s  00:00
root@Master:/usr/local/hadoop/hadoop-1.2.1/conf# scp slaves root@Slave2:/usr/local/hadoop/hadoop-1.2.1/conf
slaves                                100% 21    0.0KB/s  00:00
root@Master:/usr/local/hadoop/hadoop-1.2.1/conf#
```

进入 Slave1 或者 Slave2 节点检查 masters 和 slaves 文件的内容：

```
root@Slave1: /usr/local/hadoop/hadoop-1.2.1/conf
Master
```

```
root@Slave1: /usr/local/hadoop/hadoop-1.2.1/conf
Master
Slave1
Slave2
```

发现拷贝完全正确。

至此 Hadoop 的集群环境终于配置完成！

4. 测试 Hadoop 分布式集群环境；

首先在通过 Master 节点格式化集群的文件系统：

```
root@Master: ~
root@Master:~# hadoop namenode -format
14/06/30 20:50:13 INFO namenode.NameNode: STARTUP_MSG:
/*****
STARTUP_MSG: Starting NameNode
STARTUP_MSG:  host = Master/192.168.184.133
STARTUP_MSG:  args = [-format]
STARTUP_MSG:  version = 1.2.1
STARTUP_MSG:  build = https://svn.apache.org/repos/asf/hadoop/common/branches/b
ranch-1.2 -r 1503152; compiled by 'mattf' on Mon Jul 22 15:23:09 PDT 2013
STARTUP_MSG:  java = 1.7.0_60
*****/
Re-format filesystem in /usr/local/hadoop/hadoop-1.2.1/hdfs/name ? (Y or N) █
```

输入 “Y” 完成格式化：

我们在尝试一下停止 Hadoop 集群：

```
root@Master:/# stop-all.sh
stopping jobtracker
Slave2: stopping tasktracker
Master: stopping tasktracker
Slave1: stopping tasktracker
stopping namenode
Master: no datanode to stop
Slave1: no datanode to stop
Slave2: no datanode to stop
Master: stopping secondarynamenode
root@Master:/#
```

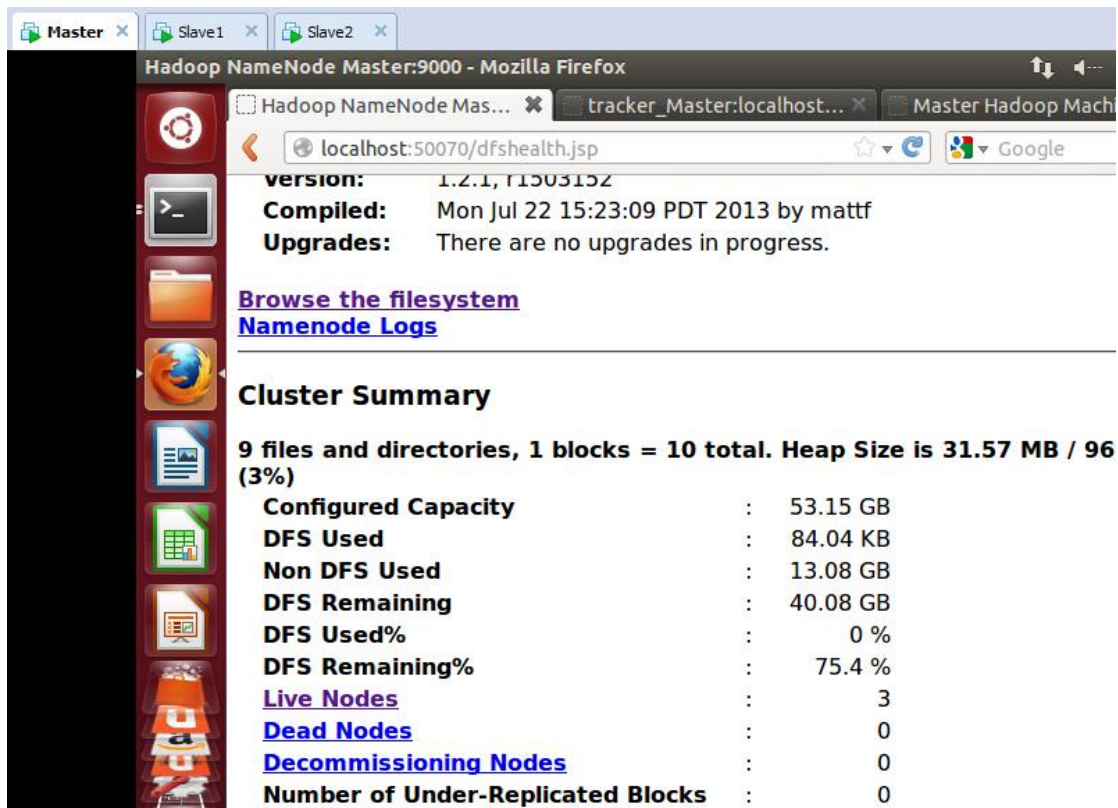
此时出现了 “no datanode to stop” 的错误，出现这种错误的原因如下：

每次使用 “hadoop namenode -format” 命令格式化文件系统的时候会出现一个新的 namenodeId，而我们在搭建 Hadoop 单机伪分布式版本的时候往我们自己创建的 tmp 目录下放了数据，现在需要把各台机器上的 “/usr/local/hadoop/hadoop-1.2.1/” 下面的 tmp 及其子目录的内容清空，于此同时把 “/tmp” 目录下的与 hadoop 相关的内容都清空，最后要把我们自定义的 hdfs 文件夹中的 data 和 name 文件夹中的内容清空：

```
root@Master: /usr/local/hadoop/hadoop-1.2.1
root@Master:/usr/local/hadoop/hadoop-1.2.1# rm -rf tmp/*
root@Master:/usr/local/hadoop/hadoop-1.2.1# rm -rf /tmp/hadoop*
root@Master:/usr/local/hadoop/hadoop-1.2.1#
```

把 Slave1 和 Slave2 中同样的内容均删除掉。

重新格式化并重新启动集群，此时进入 Master 的 Web 控制台：



此时可以看到 Live Nodes 只有三个，这正是我们预期的，因为我们 Master、Slave1、Slave2 都设置成为了 DataNode，当然 Master 本身同时也是 NameNode。

此时我们通过 JPS 命令查看一下三台机器中的进程信息：

```
root@Master: /usr/local/hadoop/hadoop-1.2.1/conf
root@Master: /usr/local/hadoop/hadoop-1.2.1/conf# jps
6055 JobTracker
6257 TaskTracker
12868 Jps
12235 DataNode
5969 SecondaryNameNode
12030 NameNode
root@Master: /usr/local/hadoop/hadoop-1.2.1/conf#
```



```
root@Slave1: /usr/local/hadoop/hadoop-1.2.1
root@Slave1:/usr/local/hadoop/hadoop-1.2.1# jps
6524 TaskTracker
6352 DataNode
6626 Jps
root@Slave1:/usr/local/hadoop/hadoop-1.2.1#
```

```
root@Slave2: /usr/local/hadoop/hadoop-1.2.1
root@Slave2:/usr/local/hadoop/hadoop-1.2.1# jps
6542 TaskTracker
6370 DataNode
6643 Jps
root@Slave2:/usr/local/hadoop/hadoop-1.2.1#
root@Slave2:/usr/local/hadoop/hadoop-1.2.1#
```

发现 Hadoop 集群的各种服务都正常启动。
至此，Hadoop 集群构建完毕。

■ Spark 亚太研究院

Spark 亚太研究院，提供 Spark、Hadoop、Android、Html5、云计算和移动互联网一站式解决方案。以帮助企业规划、部署、开发、培训和使用为核心，并规划和实施人才培训完整路径，提供源码研究和应用技术训练。

■ 近期活动及相关课程

决战云计算大数据时代 Spark 亚太研究院 100 期公益大奖堂

每周四晚上 20:00—21:00

课程介绍：http://edu.51cto.com/course/course_id-1659.html#showDesc

报名参与：http://ke.qq.com/cgi-bin/courseDetail?course_id=6167

【近期 Spark 公开课】

- ▶ 18 小时内掌握 Spark,把云计算大数据速度提升 100 倍以上

8 月 10-12 日 上海

8 月 15—17 日 北京

报名咨询：4006-998-758

QQ 学习交流群号：317540673

