

Junos®基础系列

第一天就上手：为 IOS 工程师准备的 JUNOS 入门

当用新的瞻博网络设备替换思科路由器的时刻到来的时候，这本书帮助你
将技能延伸到 **JUNOS**，使你快速自信
地过渡到更先进的操作系统

Chris Jones 著

钟敏 译

第一天就上手：为 IOS 工程师准备的 JUNOS 入门

Junos 基础系列 #8

在每个网络工程师的职业生涯中，都会遇到需要转到其它技术的时候，转型的过程会不可避免地会遇到复杂性。为了使从 IOS 到 Junos 的更加简单，你需要一本书来让你对如何操控新的瞻博网络路由器、交换机和安全设备更有信心。

《第一天就上手：为 IOS 工程师准备的 Junos 入门》满足了 IOS 工程师的需要，它并列对比了 IOS 和 Junos 的配置和技术。通过几个快捷的步骤，你可以了解从前使用 IOS 完成的任务现在如何使用 Junos 来完成。我们以直觉、技巧、和恰到好处的解释来伴随你的旅程。如果你是一名已经对 IOS 很熟悉的工程师，准备好认识一下“Junos 之道”，无论是简单地使用一下不同的语法，还是一个全新有效的组网方式。

“对于我们这些了解 IOS 而希望认识 Junos 的人，这本‘第一天’小册子正好满足了我们的需要。并列对比的 IOS 和 Junos 示例确实能帮助我们理解 Junos 配置。”

Jeff Fry, CCIE #22061

这是你接触 JUNOS 的第一天，你有任务需要完成，那么来学习如何：

- 理解 Junos 和 IOS 在思维上的差异；
- 在 Junos 设备上配置简单功能，完成常见任务；
- 将 Junos 配置与你所惯常的 IOS 配置进行比较；
- 为小型网络配置 VLAN，OSPF 及 BGP。

瞻博网络书籍专注于网络生产力和效率。完整的文库请访问 www.juniper.net/dayone。

Juniper Networks Books 出版

Junos®基础系列

第一天就上手：为 IOS 工程师准备的 Junos 入门

Chris Jones 著

钟敏 译

第一章 基础知识.....	7
第二章 基本配置.....	19
第三章 案例分析.....	59
跟住去边度.....	83

© 2012 Juniper Networks, Inc. 保留所有权利。

Juniper 网络公司, Juniper 网络公司标识, Junos, NetScreen 和 ScreenOS 是瞻博网络公司在美国和其他国家的注册商标。Junose 是瞻博网络公司的商标。所有其他商标, 服务标记, 注册商标, 商标或注册服务标记均为其各自所有者的财产。

瞻博网络公司不为本文中的任何不准确之处承担责任。瞻博网络公司保留对本文随时更改、修改、转换或修订的权利, 恕不另行通知。由瞻博网络制造或销售的产品或部件可能被瞻博网络拥有或被授权的一个或多个专利所保护: 美国专利号 5,473,599, 5,905,725, 5,909,440, 6,192,051, 6,333,650, 6,359,479, 6,406,312, 6,429,706, 6,459,579, 6,493,347, 6,538,518, 6,538,899, 6,552,918, 6,567,902, 6,578,186 及 6,590,785。

英文版由 Juniper Networks Books 出版
作者: Chris Jones
主要贡献者: Dana Yanch
技术审阅: Tom Dwyer, Nick Ryce 及 Stefan Fouant
主编: Patrick Ames
清样编辑和校对: Nancy Koerbel
J-Net 社区经理: Julie Wider

翻译/简体中文版清样编辑和校对: 钟敏

关于作者

Chris Jones 是一名网络工程师, 他持有 JNCIE-ENT #272 和 CCIE (R&S) # 25655 证书。Chris 在思科和瞻博网络的产品及解决方案领域有着超过 8 年的行业经验。

作者致谢

我要感谢我的妻子 Jennifer, 感谢她在我写这本书时给予我的所有支持, 以及在我学习 CCIE 和 JNCIE 的过程中体现的耐心。特别感谢我的好友 Dana Yanch, 他提供这了本书的 IOS 配置。没有他, 我无法做到。最后我想感谢编辑 Patrick Ames, 他使我有机会写这本 DayOne 小册子, 他的指导对于本书的顺利完成时是至关重要的。

译者谨将本书献给勤劳勇敢的瞻博网络中国团队。

ISBN: 978-1-936779-54-3 (英文印刷版)
在美国由 Vervante 公司印刷
ISBN: 978-1-936779-55-0 (英文电子版)

英文版修订历史: 2012 年 8 月第 1 版
2 3 4 5 6 7 8 9 10 #7100157-en
简体中文版修订历史: 2012 年 9 月第 1 版

在 www.juniper.net/dayone 可获得本书的多种格式(英文)。

关于贡献者和技术评审

像这样的书籍很少能做到完全客观和实用，但是下列人员的贡献有助于使这本书尽可能简单客观。

- Dana Yanch，CCIE # 25567/JNCIP-ENT，Insight Enterprises的网络架构师。Dana 是这本书的主要参与者，他作为独立贡献者提供了所有的Cisco IOS配置。
- Tom Dwyer 是 Nexum 公司的高级网络安全工程师，他拥有 18 年的网络经验，并拥有多项瞻博网络认证，包括 JNCIP-ENT/SEC 和 JNCI。
- Nick Ryce 是英国 Pulsant 的高级网络工程师。他是 JNCIE-ENT#232，在 IT 行业工作已有七年之久。Nick 对本书进行了技术审阅。
- Stefan Fount 是瞻博网络的技术讲师和 JNCP 考官。他是为数不多的三重 JNCIE 之一，他对本书进行了两个操作系统的技术审审阅。

比较 IOS 和 Junos

经过多次讨论，编辑、贡献者和笔者决定以前后逐点对比的方式来呈现两个操作系统的区别。读者应先通读他们应该已经很熟悉的 Cisco 配置，然后阅读相应的 Junos 配置以及其中的差异说明。

这本书开始的一章是关于在两个操作系统中常见的任务，然后比较了两者的配置，最后一章比较了一个简单但完整的网络配置任务。

这是一本帮助 IOS 工程师在一天内开始管理瞻博网络设备的书。

在开始阅读之前你需要了解的

- Junos 操作系统的基本知识对于完成本书中的任务是必须的，虽然本书针对比较 IOS 和 Junos 常见的配置，但它不包括对 JUNOS CLI 的介绍。
- 本书假定您已阅读了《Day One: Exploring the Junos CLI》¹和《Day One: Configuring Junos Basics》这两本 DayOne 小册子。两者都可以在 <http://www.juniper.com/dayone> 免费下载。
- 本书假定读者熟悉 IOS，熟悉配置基本的管理任务以及简单的 IGP 和 BGP。
- 最后，本书假定读者理解一般的网络协议，如 OSPF 和 BGP。

读完这本小册子，你将可以

- 理解 Junos 和 IOS 在思维上的差异；
- 在 Junos 设备上配置简单功能，完成常见任务；
- 将 Junos 配置与你所惯常的 IOS 配置进行比较；
- 为小型网络配置 VLAN，OSPF 及 BGP。

欢迎阅读 DayOne 丛书

DayOne 丛书帮助你为你提供第一天所需的信息，帮助你对新的课题快速上手。本系列通过直观的解释，一步步的指引和易于仿效的实际例子涵盖了基础知识，同时还为深入了解提供了大量的参考文献。如需了解 DayOne 书库的更多信息，请参阅 <http://www.juniper.net/dayone>。

¹ 译者注：该小册子的繁体中文版为 [《第一次使用就上手：Junos CLI 大探索》](#)。

第一章

基础知识

设置主机名	9
创建用户	10
启用 SSHv2	12
为端口分配 IP 地址	13
保存配置	15
升级软件	17
总结	18

本书将帮助 IOS 工程师将常见的网络任务与 Junos 的对应实现进行比较。这些任务看上去似乎太简单了，但在学习任何新的语言时，都是从琐碎的日常事物开始的。

想知道更多？ 如果你没有留意前言，这里再强调一下：作者假设你知道Junos的基础知识。你可以通过《Day One: Exploring the Junos CLI》²和《Day One: Configuring Junos Basics》这两本书来掌握这些知识，两者都可以在<http://www.juniper.com/dayone>免费下载，或通过iTunes的iBookstore和亚马逊的Kindle书店获得。

本章所介绍的基本任务，是工程师会被要求完成的基本 Juniper 路由器配置：

- 设置主机名
- 创建用户
- 启用 SSHv2
- 为端口分配 IP 地址
- 保存配置
- 升级软件

² 译者注：该小册子的繁体中文版为 [《第一次使用就上手：Junos CLI 大探索》](#)。

设置主机名

主机名是路由器上最常见的配置，它对于在常规配置和日志记录中标识设备是必不可少的。我们来比较一下在两种 OS 中如何完成这项工作。

IOS 配置

在 IOS 设备上设置主机名

```
Router#configure terminal
Router(config)#hostname R1
R1(config)# end
R1#
```

验证主机名已经被设置在 IOS 设备

路由器的提示符已经反映了新的主机名，不过你仍然可以查看 running-config 中所配置的主机名：

```
R1#show running-config | include hostname
hostname R1
R1#
```

Junos 配置

在 Junos 设备上设置主机名

```
cjones@router> configure
cjones@router# set system host-name R1
cjones@router# commit and-quit
cjones@R1>
```

验证主机名已经被设置在 Junos 设备

同样的，路由器的提示符已经反映了新的主机名。你也可以查看配置中所配置的主机名：

```
cjones@R1> show configuration system | match host-name
host-name R1;
```

小结

你可以看到，主机名配置在 IOS 和 Junos 是非常相似的，都只有一个命令。到目前为止，一切良好。

你当然会注意到两个操作系统有不同的外观和感觉。看完这本书，你就知道为什么，并且对这种改变感觉舒适。下面我们尝试另一项任务，注意两者的不同。

创建用户

创建用户是配置中重要的一部分。让我们比较一下如何允许多个工程师对设备的访问，而他们在查看和更改配置上有不同等级的权利。

IOS 配置

在 IOS 中创建用户并指定明文的密码

```
R1#configure terminal
R1(config)#username plaintextuser password p@$sw0Rd
R1(config)#end
R1#
```

在 IOS 中创建用户并指定加密的密码

```
R1#configure terminal
R1(config)#username encrypteduser secret p@$sw0Rd
R1(config)#end
R1#
```

在 IOS 中验证用户已经被创建

```
R1#show running-config | include username
username plaintextuser password 0 p@$sw0Rd
username encrypteduser secret 5 $1$kkb9$i/TLOSPZr.F8g2LD/MFVi0
R1#
```

Junos 配置

在 Junos 中配置一个用户

```
cjones@R1> configure
cjones@R1# set system login user bob class super-user full-name "Bob" authentication
plain-text-password
New password:
Retype new password:
cjones@router# commit and-quit
cjones@R1>
```

在 Junos 中验证用户已经被创建

```
cjones@R1> show configuration system login
user bob {
    full-name Bob;
    uid 2001;
    class super-user;
    authentication {
        encrypted-password "$1$wkvA6Ppe$NHN5HgacMAE4OYlvMxF7j/"; ## SECRET-DATA
    }
}
cjones@R1>
```

小结

在 Junos 中的用户配置比在 IOS 中稍微复杂，但也明显地更为灵活。可以为用户分配预先定义好的用户职能是一个明显的优势。

请注意在 Junos 默认对密码进行散列。

让我们在此稍作停顿，讨论一下 Junos 配置的显示。这个例子显示了使用花括

号的标准 Junos 配置输出，这可能起初会令人生畏(不是 C 语言编程哦)，然而一旦你了解它们是如何布局，你会发现相对于 IOS 无迹可寻的配置段落，Junos 更为直观。而当你的配置日趋庞大时，你会对这样的输出格式爱不释手。

Junos 小提示 反之，在某些时候能够看到一个配置所对应的 `set` 命令列表是很有帮助的。要做到这一点，只需在 `show configuration` 后使用 `| display set`：

```
cjones@R1> show configuration | display set3
set system login user bob full-name Bob
set system login user bob uid 2001
set system login user bob class super-user
set system login user bob authentication encrypted-password "$1$wkvA6Ppe$NHN5HgacMAE4OYlvMxF7j/"
```

在配置模式下同样也可以做到：

```
cjones@R1# show | display set
```

```
set system login user bob full-name Bob
set system login user bob uid 2001
set system login user bob class super-user
set system login user bob authentication encrypted-password "$1$wkvA6Ppe$NHN5HgacMAE4OYlvMxF7j/"
```

³ 译者注：你有没有留意到当前用户 `cjones` 并没有出现在这条命令的输出中？要真的仅仅得到文中所述的输出，请用 `show configuration system login user bob | display set`。

开启 SSHv2

SSHv2 是一种用于加密通信的协议，而相比之下 Telnet 是没有加密的。让我们看看如何在两种操作系统中完成这个任务。

IOS 配置

在 IOS 中配置 SSHv2

```
R1#configure terminal
R1(config)#ip domain-name juniper.net
R1(config)#crypto key generate rsa modulus 1024
R1(config)#ip ssh version 2
R1(config)#end
R1#
```

在 IOS 中验证 SSHv2

```
R1#show ip ssh
SSH Enabled - version 2.0
Authentication timeout: 120 secs; Authentication retries: 3
R1#
```

Junos 配置

在 Junos 中配置 SSHv2

```
cjones@R1> configure
cjones@R1# set system services ssh
cjones@R1# commit and-quit
cjones@R1>
```

在 Junos 中验证 SSHv2

```
cjones@R1> show configuration system services
ssh;
```

小结

在 Junos 中配置 SHv2 更为直观，由于 Junos 的底层是 FreeBSD，RSA 密钥在操作系统的安装过程中已经被生成了。而 IOS 需要在生成 RSA 密钥之前先设置域名。

SSH 的默认版本在 IOS 中是 1.99，在 Junos 中是 2。

注意 如果配置了防火墙过滤器的话，需要确保其允许 TCP 端口 22 的 SSH 流量通过。

为端口分配 IP 地址

对于任何的网络配置来说，分配 IP 地址都是一项基本任务，因为设备需要 IP 地址才能互通。如何完成这项基本任务在两个操作系统上有很明显的不同。

IOS 配置

在 IOS 中为端口分配 IP 地址

```
R1#configure terminal
R1 (config)#interface f0/0
R1 (config-if)#ip address 192.0.2.1 255.255.255.0
R1 (config-if)#no shutdown
R1 (config-if)#end
R1#
```

在 IOS 中检验端口的 IP 地址

```
R1#show ip interface brief FastEthernet 0/0
```

Interface	IP-Address	OK?	Method	Status	Protocol
FastEthernet0/0	192.0.2.1	YES	manual	up	up

Junos 配置

在 Junos 中为端口分配 IP 地址

在这个 Junos 的例子中显示的语法使用单条命令指定了若干个参数:

- 端口名(ge-0/0/0)
- 逻辑单元号(本例中为 0)
- 协议族(本例中为 inet，用于 IPv4 地址的协议族)
- 端口地址(192.0.2.1/24)

注意 在Junos中一个接口的逻辑单元号在本质上相当于IOS的子接口。不过为了保持一致性，在Junos中单元号是必须的。逻辑配置(如IP地址等)位于单元层次，而物理配置(如速度，双工等)位于物理接口层次。单元号通常为 0，除非一些特殊配置，如VLAN标记。⁴

```
cjones@R1> configure
cjones@R1# set interfaces ge-0/0/0 unit 0 family inet address 192.0.2.1/24
cjones@R1# commit and-quit
cjones@R1>
```

在 Junos 中检验端口的 IP 地址

```
cjones@R1> show configuration interfaces fe-0/0/0
unit 0 {
    family inet {
        address 192.0.2.1/24;
    }
}

cjones@R1> show interfaces terse fe-0/0/0
```

⁴有两种等价的方式可以用于在 CLI 中指定接口单元号: `interfaces <interface-name> unit <unit-number>` 或者 `interfaces <interface-name>.<unit-number>`，在命令输出中通常使用第二种形式，所以下一页你看到 `fe-0/0/0.0` 的时候请不要惊讶。

Interface	Admin	Link	Proto	Local	Remote
fe-0/0/0	up	up			
fe-0/0/0.0	up	up	inet	192.0.2.1/24	

小结

在 Junos 中为一个端口分配 IP 地址只需要一条命令，要注意的是，与 IOS 要求掩码一点分十进制输入不同，Junos 允许你使用 CIDR 表示法。

注意 在 Junos 中如果不指定子网掩码的话，系统作为/32 处理。

在 IOS 中，除非使用 **secondary** 关键字，否则为端口再次配置一个 IP 地址会导致原有的地址被覆盖。在 Junos 中，新旧地址会并存。

在 IOS 中，端口是默认关闭的，需要在配置中指定 **no shutdown** 来启用。在 Junos 中，端口是默认启用的，如果需要关闭的话，使用 **set interfaces <interface> disable** 并提交。

保存配置

保存配置是路由器或者交换机管理的一个重要组成部分，如果不保存的话，设备重启时配置就丢失了。我们看看两个操作系统是如何以不同的方式来实现的。

IOS 配置

在 IOS 中保存配置

```
R1#copy running-config startup-config
Destination filename [startup-config]?
Building configuration...
[OK]
R1#
```

在 IOS 中验证配置已经被保存

验证配置需要检查 `nvr`am: 文件系统, 确认 `startup-config` 文件存在且其长度非零。

```
R1#dir nvram:
Directory of nvram:/
 52  -rw-          839          <no date>  startup-config
 53  ----         1924          <no date>  private-config
  1  ----          15          <no date>  persistent-data
  2  -rw-           0          <no date>  ifIndex-table
57336 bytes total (52473 bytes free)
R1#
```

Junos 配置

在 Junos 中，活动配置与设备启动时加载的配置是相同的。

在 Junos 中保存候选配置是通过 `commit` 命令来完成的，这个命令会试图激活候选配置，并在应用更改之前提示你潜在的问题。

你可以使用 `commit check` 命令来单纯进行配置校验而不真正激活。

最后，你可以使用 `commit confirmed` 命令来让路由器在指定时间内回滚到上一次的正确配置，这在你试图提交对防火墙过滤器或者路由策略的重大变动时尤其实用。当确认配置更改正确后，你可以在确认时间窗口内再次使用 `commit` 来取消回滚，时间窗口的默认值为 10 分钟。

在 Junos 中保存配置

```
cjones@R1> configure
cjones@R1# commit and-quit
cjones@R1>
```

在 Junos 中验证配置已经被保存

验证配置已经被保存的最佳途径是将其与之前的版本进行比较，这里我们比较一下 `rollback 0` (当前配置) 和 `rollback 1` (之前提交的)

```
cjones@R1> show system rollback compare 1
[edit interfaces]
- fe-0/0/0 {
-     unit 0 {
-         family inet {
-             address 192.0.2.1/24;
-         }
-     }
- }
```

- }

小结

在 IOS 中,键入的命令是立即生效的。要保存配置,你必须将 **running configuration** 复制到 **startup configuration**。相比之下,Junos 使用的是提交模式,它允许你键入配置命令但不会对设备产生即时的影响。

Junos 还允许你使用 **confirmed** 关键字来实现自动回滚,而 IOS 需要对路由器进行配置才能回滚到 **startup-config**。

验证 Junos 软件已经被升级

```
cjones@R1> show version
Hostname: R1
Model: srx100h
JUNOS Software Release [10.4R7.5]
...
```

小结

在 IOS 和 Junos 中，升级软件都是很直截了当的，只需要一个命令即可完成。不过在 IOS 中需要另外执行 `reload` 命令。

总结

你习惯了两种操作系统在外观上的差别了吗？记住，在 Junos 中，你可以在 `show configuration` 后面使用 `| display set` 来显示你所熟悉的配置输出。遗憾的是，IOS 并没有相应的机制来实现类似 Junos 的层次化显示。

好了，这些都是基本概念，接下来的章节你可以在此基础上完成一些更实质性的工作。你启动了瞻博设备，建立了用户帐号，还升级了软件，现在是进行一些基本配置的时候了。

第二章

基本配置

单域 OSPF	20
EBGP	32
IBGP	40
VLAN 配置	47
简单 NAT	53

正如“第一天就上手”的书名，我们在动手的过程中学习。这一章直奔主题，告诉你如何构建在真实网络中使用的配置。每个例子都详细介绍了具体的技术，并且在结尾的小结部分总结比较了 IOS 和 Junos 配置的差异

让我们利用例子中的配置来学习从 IOS 向 Junos 的迁移，比较配置顺序，找到相应的命令，并逐步习惯 Junos 的输出。

单域 OSPF

让我们来构建一个有三个路由器的小型网络，并加载 OSPF 配置，使得路由器之间可以交换路由信息。

图 2.1 是这个小型网络的示意图，简单的拓扑中有三台路由器配置为单 OSPF 域。

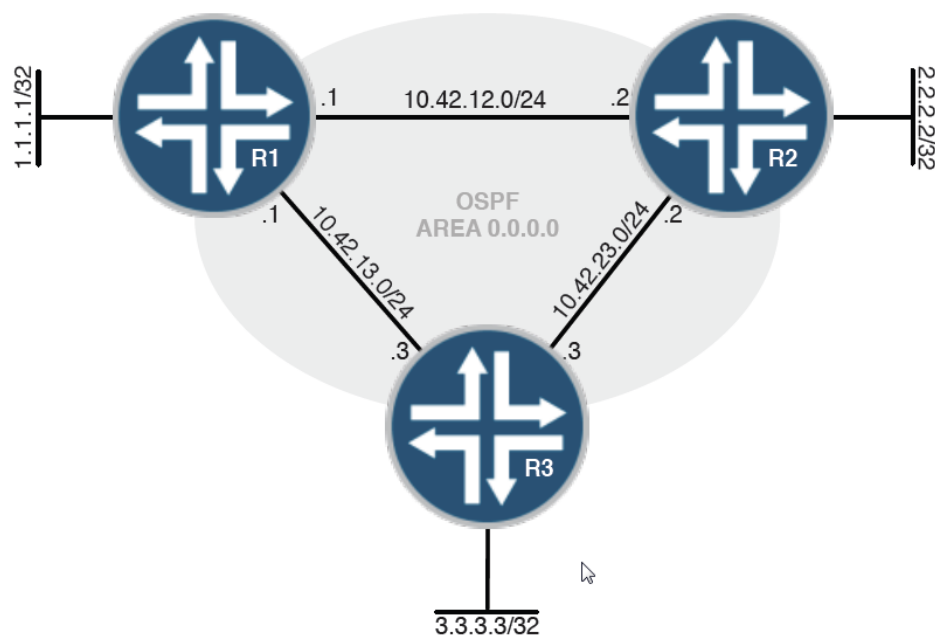


图 2.1 单域 OSPF 拓扑

在本节中我们的任务是：

- 在 R1、R2 和 R3 之间配置 OSPF；
- 将所有路由器的 loopback 端口通告到 OSPF，并确认 loopback 被配置为被动接口；
- 手动配置 OSPF router-id。

IOS 配置

让我们先按照下面的步骤使用 IOS 来配置这个网络。

配置 IOS 路由器以完成初始连接

为了让路由器之间能够通信，在互联端口上必须要配置 IP 地址。下面是在 IOS 路由器上设置 IP 地址的步骤。

1. 在 R1 的端口上设置 IP 地址：

```
R1# configure terminal
R1(config)# interface loopback 0
R1(config-if)# ip address 1.1.1.1 255.255.255.255
```

```

R1(config-if)# no shutdown
R1(config-if)# interface FastEthernet 0/0
R1(config-if)# ip address 10.42.13.1 255.255.255.0
R1(config-if)# no shutdown
R1(config-if)# interface FastEthernet 0/1
R1(config-if)# ip address 10.42.12.1 255.255.255.0
R1(config-if)# no shutdown
R1(config-if)# end
R1#

```

2. 在 R2 的端口上设置 IP 地址:

```

R2# configure terminal
R2(config)# interface loopback 0
R2(config-if)# ip address 2.2.2.2 255.255.255.255
R2(config-if)# no shutdown
R2(config-if)# interface FastEthernet 0/0
R2(config-if)# ip address 10.42.12.2 255.255.255.0
R2(config-if)# no shutdown
R2(config-if)# interface FastEthernet 0/1
R2(config-if)# ip address 10.42.23.2 255.255.255.0
R2(config-if)# no shutdown
R2(config-if)# end
R2#

```

3. 在 R3 的端口上设置 IP 地址:

```

R3# configure terminal
R3(config)# interface loopback 0
R3(config-if)# ip address 3.3.3.3 255.255.255.255
R3(config-if)# no shutdown
R3(config-if)# interface FastEthernet 0/0
R3(config-if)# ip address 10.42.13.3 255.255.255.0
R3(config-if)# no shutdown
R3(config-if)# interface FastEthernet 0/1
R3(config-if)# ip address 10.42.23.3 255.255.255.0
R3(config-if)# no shutdown
R3(config-if)# end
R3#

```

验证 IOS 路由器的初始连接

为了验证 IP 地址的配置是正确的,我们可以用简单的 ping 命令来测试远端连接的设备是否可达。

1. 从 R1 ping R2:

```

R1#ping 10.42.12.2
Type escape sequence to abort.
Sending 5, 100-byte ICMP Echos to 10.42.12.2, timeout is 2 seconds:
!!!!
Success rate is 100 percent (5/5), round-trip min/avg/max = 1/2/4 ms
R1#

```

2. 从 R1 ping R3:

```

R1#ping 10.42.13.3
Type escape sequence to abort.
Sending 5, 100-byte ICMP Echos to 10.42.13.3, timeout is 2 seconds:
!!!!
Success rate is 100 percent (5/5), round-trip min/avg/max = 1/2/4 ms
R1#

```

3. 从 R2 ping R3:

```

R2#ping 10.42.23.3

```

```
Type escape sequence to abort.
Sending 5, 100-byte ICMP Echos to 10.42.0.3, timeout is 2 seconds:
!!!!
Success rate is 100 percent (5/5), round-trip min/avg/max = 1/2/4 ms
R2#
```

配置 IOS 路由器的 OSPF

为了达成网络的完全可达，路由器必须知道每一个网络。通常这是通过某种内部网关协议(Interior Gateway Protocol -IGP)来实现的，OSPF(Open Shortest Path First)就是 IGP 的一种。

OSPF 以链路状态广播(Link-State Advertisements - LSAs)的形式发送更新，在域中的所有路由器使用这些信息来构建网络的全貌。

1. 在 R1 上配置 OSPF 进程并将端口分配到 area 0:

```
R1# configure terminal
R1(config)# router ospf 1
R1(config-router)# network 10.42.12.1 0.0.0.0 area 0
R1(config-router)# network 10.42.13.1 0.0.0.0 area 0
R1(config-router)# end
R1#
```

2. 在 R2 上配置 OSPF 进程并将端口分配到 area 0:

```
R2# configure terminal
R2(config)# router ospf 1
R2(config-router)# network 10.42.12.2 0.0.0.0 area 0
R2(config-router)# network 10.42.23.2 0.0.0.0 area 0
R2(config-router)# end
R2#
```

3. 在 R3 上配置 OSPF 进程并将端口分配到 area 0:

```
R3# configure terminal
R3(config)# router ospf 1
R3(config-router)# network 10.42.13.3 0.0.0.0 area 0
R3(config-router)# network 10.42.23.3 0.0.0.0 area 0
R3(config-router)# end
R3#
```

验证 OSPF 配置

通过检查 OSPF 是否形成正确的邻接关系，我们可以验证 OSPF 是否被正确地配置。活跃的 OSPF 邻居应该处于 FULL 状态。

1. 在 R1 上验证 OSPF 邻接关系:

```
R1#show ip ospf neighbor
```

Neighbor ID	Pri	State	Dead Time	Address	Interface
2.2.2.2	1	FULL/BDR	00:00:32	10.42.12.2	FastEthernet0/1
3.3.3.3	1	FULL/DR	00:00:34	10.42.13.3	FastEthernet0/0

```
R1#
```

2. 在 R2 上验证 OSPF 邻接关系:

```
R2#show ip ospf neighbor
```

Neighbor ID	Pri	State	Dead Time	Address	Interface
1.1.1.1	1	FULL/DROTHER	00:00:30	10.42.12.1	FastEthernet0/0
3.3.3.3	1	FULL/BDR	00:00:33	10.42.23.3	FastEthernet0/1

```
R2#
```

3. 在 R3 上验证 OSPF 邻接关系:

```
R3#show ip ospf neighbor
```

Neighbor ID	Pri	State	Dead Time	Address	Interface
-------------	-----	-------	-----------	---------	-----------

```

1.1.1.1      1    FULL/DROTHER  00:00:37    10.42.13.1    FastEthernet0/0
2.2.2.2      1    FULL/BDR    00:00:35    10.42.23.2    FastEthernet0/1
R3#

```

在 OSPF 上公告 loopback 端口并确保其处于被动

将 loopback 端口公告到 OSPF 中是可选的，在本书中这样做的目的是为了提供一个简单的方式来验证 OSPF 域工作正常。在生产网络中，loopback 被公告到 OSPF 中，这样设备的 router-id 成为可达地址。

使用 `passive-interface` 命令的目的是告诉软件将连接作为 Type-1 LSA 发布，但是不要试图在这个端口上建立邻接关系。

1. 在 R1 上配置 OSPF，包含 loopback 端口；由于永远不会通过 loopback 建立邻接关系，配置中包含了 `passive-interface` 命令：

```

R1# configure terminal
R1(config)# router ospf 1
R1(config-router)# network 1.1.1.1 0.0.0.0 area 0
R1(config-router)# passive-interface lo0
R1(config-router)# end
R1#

```

2. 接着，在 R2 的 loopback 上配置被动 OSPF：

```

R2# configure terminal
R2(config)# router ospf 1
R2(config-router)# network 2.2.2.2 0.0.0.0 area 0
R2(config-router)# passive-interface lo0
R2(config-router)# end
R2#

```

3. 最后，在 R3 的 loopback 端口上配置 OSPF 并将其设置为被动：

```

R3# configure terminal
R3(config)# router ospf 1
R3(config-router)# network 3.3.3.3 0.0.0.0 area 0
R3(config-router)# passive-interface lo0
R3(config-router)# end
R3#

```

验证 OSPF 中的 loopback 端口

验证 loopback 端口很简单，只要检查一下 loopback 端口是不是在运行 OSPF。然后通过检查其它路由器上的 RIB，确定 loopback 端口已经被其它设备学习到。

另外也可以简单地任何一台路由器上检查一下 LSDB 以获取类似的信息。

1. 确定 R1 上的 loopback (1.1.1.1/32)配置了 OSPF，并且已经被 R2 和 R3 学到：

```

R1#show ip ospf interface lo0
Loopback0 is up, line protocol is up
  Internet Address 1.1.1.1/32, Area 0
  Process ID 1, Router ID 1.1.1.1, Network Type LOOPBACK, Cost: 1
  Loopback interface is treated as a stub Host
R1#

```

```

R2#show ip route 1.1.1.1
Routing entry for 1.1.1.1/32
  Known via "ospf 1", distance 110, metric 11, type intra area
  Last update from 10.42.12.1 on FastEthernet0/0, 00:02:41 ago
  Routing Descriptor Blocks:
    * 10.42.12.1, from 1.1.1.1, 00:02:41 ago, via FastEthernet0/0
      Route metric is 11, traffic share count is 1
R2#

```

```
R3#show ip route 1.1.1.1
Routing entry for 1.1.1.1/32
  Known via "ospf 1", distance 110, metric 11, type intra area
  Last update from 10.42.13.1 on FastEthernet0/0, 00:03:12 ago
  Routing Descriptor Blocks:
    * 10.42.13.1, from 1.1.1.1, 00:03:12 ago, via FastEthernet0/0
      Route metric is 11, traffic share count is 1
R3#
```

2. 确定 R2 上的 loopback (2.2.2.2/32)配置了 OSPF, 并且已经被 R1 和 R3 学到:

```
R2#show ip ospf interface lo0
Loopback0 is up, line protocol is up
  Internet Address 2.2.2.2/32, Area 0
  Process ID 1, Router ID 2.2.2.2, Network Type LOOPBACK, Cost: 1
  Loopback interface is treated as a stub Host
R2#
```

```
R1#show ip route 2.2.2.2
Routing entry for 2.2.2.2/32
  Known via "ospf 1", distance 110, metric 11, type intra area
  Last update from 10.42.12.2 on FastEthernet0/1, 00:01:58 ago
  Routing Descriptor Blocks:
    * 10.42.12.2, from 2.2.2.2, 00:01:58 ago, via FastEthernet0/1
      Route metric is 11, traffic share count is 1
R1#
```

```
R3#show ip route 2.2.2.2
Routing entry for 2.2.2.2/32
  Known via "ospf 1", distance 110, metric 11, type intra area
  Last update from 10.42.23.2 on FastEthernet0/1, 00:02:16 ago
  Routing Descriptor Blocks:
    * 10.42.23.2, from 2.2.2.2, 00:02:16 ago, via FastEthernet0/1
      Route metric is 11, traffic share count is 1
R3#
```

3. 确定 R3 上的 loopback (3.3.3.3/32)配置了 OSPF, 并且已经被 R1 和 R2 学到:

```
R3#show ip ospf int lo0
Loopback0 is up, line protocol is up
  Internet Address 3.3.3.3/32, Area 0
  Process ID 1, Router ID 3.3.3.3, Network Type LOOPBACK, Cost: 1
  Loopback interface is treated as a stub Host
R3#
```

```
R1#show ip route 3.3.3.3
Routing entry for 3.3.3.3/32
  Known via "ospf 1", distance 110, metric 11, type intra area
  Last update from 10.42.13.3 on FastEthernet0/0, 00:04:36 ago
  Routing Descriptor Blocks:
    * 10.42.13.3, from 3.3.3.3, 00:04:36 ago, via FastEthernet0/0
      Route metric is 11, traffic share count is 1
R1#
```

```
R2#show ip route 3.3.3.3
Routing entry for 3.3.3.3/32
  Known via "ospf 1", distance 110, metric 11, type intra area
  Last update from 10.42.23.3 on FastEthernet0/1, 00:04:50 ago
  Routing Descriptor Blocks:
    * 10.42.23.3, from 3.3.3.3, 00:04:50 ago, via FastEthernet0/1
      Route metric is 11, traffic share count is 1
R2#
```


手动配置 OSPF router-id

有时手工配置 router-id 会有一些好处，例如当我们需要这个 router-id 不是分配给路由器端口的任何一个地址的时候。有时候也需要明确指定 OSPF 链路状态数据库(link-state database - LSDB)中的表目，而不是由路由器自行选择。

注意 OSPF router-id 是一个 32bit 的数字，以点分十进制的形式表示(类似于 IPv4 地址)。然而，像 OSPF area ID 一样，router-id 并不一定是个 IP 地址，虽然通常来说它会是设备上的一个 IP 地址。

1. 在 R1 上使用 **show** 命令查看当前的 router-id，然后配置 R1 使用 router-id 11.11.11.11:

```
R1#show ip protocols | include ID
  Router ID 1.1.1.1
R1#configure terminal
R1(config)#router ospf 1
R1(config-router)#router-id 11.11.11.11
Reload or use "clear ip ospf process" command, for this to take effect
R1(config-router)#end
R1#clear ip ospf process
Reset ALL OSPF processes? [no]: yes
R1#
```

2. 在 R2 上使用 **show** 命令查看当前的 router-id，然后配置 R2 使用 router-id 22.22.22.22:

```
R2#show ip protocols | include ID
  Router ID 2.2.2.2
R2#configure terminal
R2(config)#router ospf 1
R2(config-router)#router-id 22.22.22.22
Reload or use "clear ip ospf process" command, for this to take effect
R2(config-router)#end
R2#clear ip ospf process
Reset ALL OSPF processes? [no]: yes
R2#
```

3. 在 R3 上使用 **show** 命令查看当前的 router-id，然后配置 R3 使用 router-id 33.33.33.33:

```
R3#show ip protocols | include ID
  Router ID 3.3.3.3
R3#configure terminal
R3(config)#router ospf 1
R3(config-router)#router-id 33.33.33.33
Reload or use "clear ip ospf process" command, for this to take effect
R3(config-router)#end
R3#clear ip ospf process
Reset ALL OSPF processes? [no]: yes
R3#
```

验证手工配置的 OSPF router-id

通过检查 **show ip protocol** 命令的输出，你可以验证 router-id 是否已经被正确地设置。另外也可以查看邻接表中的 ID 是否已经相应地改变。

1. 在 R1 上使用 **show** 命令验证 router-id:

```
R1#show ip protocols | include ID
  Router ID 11.11.11.11
R1#
R1#show ip ospf neighbor
```

Neighbor ID	Pri	State	Dead Time	Address	Interface
-------------	-----	-------	-----------	---------	-----------

```

33.33.33.33      1    FULL/BDR      00:00:33      10.42.13.3      FastEthernet0/0
22.22.22.22      1    FULL/BDR      00:00:36      10.42.12.2      FastEthernet0/1
R1#

```

2. 在 R2 上使用 show 命令验证 router-id:

```

R2#show ip protocols | include ID
  Router ID 22.22.22.22
R2#show ip ospf neighbor
Neighbor ID      Pri   State           Dead Time   Address      Interface
33.33.33.33      1    FULL/BDR        00:00:31   10.42.23.3   FastEthernet0/1
11.11.11.11      1    FULL/DR         00:00:39   10.42.12.1   FastEthernet0/0
R2#

```

3. 在 R3 上使用 show 命令验证 router-id:

```

R3#show ip protocols | include ID
  Router ID 33.33.33.33
R3#show ip ospf neighbor
Neighbor ID      Pri   State           Dead Time   Address      Interface
22.22.22.22      1    FULL/DR         00:00:30   10.42.23.2   FastEthernet0/1
11.11.11.11      1    FULL/DR         00:00:36   10.42.13.1   FastEthernet0/0
R3#

```

Junos 配置

现在我们来在 Junos 中配置同样的拓扑。请留意配置的逻辑结构以及接口和协议配置的位置。如此简单且标准化的结构，其优点是明显的。

配置 Junos 路由器以完成初始连接

这里你要运用在第一章中学到的知识，在每个路由器上为端口指定地址，包括 loopback 端口。

1. 在 R1 上设置端口 IP 地址:

```

cjones@R1> configure
Entering configuration mode
[edit]
cjones@R1# set interfaces fe-0/0/0.0 family inet address 10.42.12.1/24
[edit]
cjones@R1# set interfaces fe-0/0/1.0 family inet address 10.42.13.1/24
[edit]
cjones@R1# set interfaces lo0.0 family inet address 1.1.1.1/32
[edit]
cjones@R1# commit and-quit
commit complete
Exiting configuration mode

```

2. 在 R2 上设置端口 IP 地址:

```

cjones@R2> configure
Entering configuration mode
[edit]
cjones@R2# set interfaces fe-0/0/0.0 family inet address 10.42.23.2/24
[edit]
cjones@R2# set interfaces fe-0/0/1.0 family inet address 10.42.12.2/24
[edit]
cjones@R2# set interfaces lo0.0 family inet address 2.2.2.2/32
[edit]
cjones@R2# commit and-quit
commit complete
Exiting configuration mode

```

3. 在 R3 上设置端口 IP 地址:

```
cjones@R3> configure
Entering configuration mode
[edit]
cjones@R3# set interfaces fe-0/0/0.0 family inet address 10.42.13.3/24
[edit]
cjones@R3# set interfaces fe-0/0/1.0 family inet address 10.42.23.3/24
[edit]
cjones@R3# set interfaces lo0.0 family inet address 3.3.3.3/32
[edit]
cjones@R3# commit and-quit
commit complete
Exiting configuration mode
```

验证 Junos 路由器的初始连接

1. 从 R1 ping R2:

```
cjones@R1> ping 10.42.12.2 rapid
PING 10.42.12.2 (10.42.12.2): 56 data bytes
!!!!
--- 10.42.12.2 ping statistics ---
5 packets transmitted, 5 packets received, 0% packet loss
round-trip min/avg/max/stddev = 2.238/2.791/4.261/0.752 ms
```

2. 从 R1 ping R3:

```
cjones@R1> ping 10.42.13.3 rapid
PING 10.42.13.3 (10.42.13.3): 56 data bytes
!!!!
--- 10.42.13.3 ping statistics ---
5 packets transmitted, 5 packets received, 0% packet loss
round-trip min/avg/max/stddev = 2.332/5.504/17.629/6.063 ms
```

3. 从 R2 ping R3:

```
cjones@R2> ping 10.42.23.3 rapid
PING 10.42.23.3 (10.42.23.3): 56 data bytes
!!!!
--- 10.42.23.3 ping statistics ---
5 packets transmitted, 5 packets received, 0% packet loss
round-trip min/avg/max/stddev = 2.042/7.932/30.717/11.394 ms
```

配置 Junos 路由器的 OSPF

在 Junos 中配置 OSPF 相对来说没有那么痛苦，唯一需要做的就是将每个端口添加到 OSPF 层级中。你会注意到，在 Junos 中，哪个端口运行 OSPF 是在 protocol 配置节中定义的，相对于 IOS 中过时的 network 命令，Junos 显然大大降低了复杂程度。

欲知更多? 要获得更多关于配置 OSPF 的信息，请参阅《Day One Book: Advanced OSPF in the Enterprise》，你可以在<http://www.juniper.net/dayone>免费获得。⁶

1. 在 R1 上配置 OSPF 进程并将端口分配到 area 0:

```
cjones@R1> configure
Entering configuration mode
[edit]
cjones@R1# set protocols ospf area 0 interface fe-0/0/0.0
[edit]
```

⁶ 译者注：《Junos Enterprise Routing: A Practical Guide to Junos Routing and Certification》也是一本不错的参考书(此书无法在 Juniper 的网站免费下载)。

```
cjones@R1# set protocols ospf area 0 interface fe-0/0/1.0
[edit]
cjones@R1# commit and-quit
commit complete
Exiting configuration mode
```

2. 在 R2 上配置 OSPF 进程并将端口分配到 area 0:

```
cjones@R2> configure
Entering configuration mode
[edit]
cjones@R2# set protocols ospf area 0 interface fe-0/0/0.0
[edit]
cjones@R2# set protocols ospf area 0 interface fe-0/0/1.0
[edit]
cjones@R2# commit and-quit
commit complete
Exiting configuration mode
```

3. 在 R3 上配置 OSPF 进程并将端口分配到 area 0:

```
cjones@R3> configure
Entering configuration mode
[edit]
cjones@R3# set protocols ospf area 0 interface fe-0/0/0.0
[edit]
cjones@R3# set protocols ospf area 0 interface fe-0/0/1.0
[edit]
cjones@R3# commit and-quit
commit complete
Exiting configuration mode
```

验证 OSPF 配置

要验证 OSPF 已经被正确配置，最简单的方法是验证预期的邻接关系已经建立。命令的输出与 IOS 等价命令的输出提供了类似的信息。

1. 在 R1 上验证 OSPF 邻接关系:

```
cjones@R1> show ospf neighbor
```

Address	Interface	State	ID	Pri	Dead
10.42.12.2	fe-0/0/0.0	Full	2.2.2.2	128	31
10.42.13.3	fe-0/0/1.0	Full	3.3.3.3	128	34

2. 在 R2 上验证 OSPF 邻接关系:

```
cjones@R2> show ospf neighbor
```

Address	Interface	State	ID	Pri	Dead
10.42.23.3	fe-0/0/0.0	Full	3.3.3.3	128	38
10.42.12.1	fe-0/0/1.0	Full	1.1.1.1	128	39

3. 在 R3 上验证 OSPF 邻接关系:

```
cjones@R3> show ospf neighbor
```

Address	Interface	State	ID	Pri	Dead
10.42.13.1	fe-0/0/0.0	Full	1.1.1.1	128	39
10.42.23.2	fe-0/0/1.0	Full	2.2.2.2	128	34

在 OSPF 上公告 loopback 端口并确保其处于被动

1. 在 R1 上配置 OSPF，包含 loopback 端口；由于永远不会通过 loopback 建立邻接关系，配置中包含了 `passive` 选项:

```
cjones@R1> configure
Entering configuration mode
[edit]
cjones@R1# set protocols ospf area 0 interface lo0.0 passive
```

```
[edit]
cjones@R1# commit and-quit
commit complete
Exiting configuration mode
```

2. 接下来在 R2 上为 loopback 配置 OSPF 并设为被动:

```
cjones@R2> configure
Entering configuration mode
[edit]
cjones@R2# set protocols ospf area 0 interface lo0.0 passive
[edit]
cjones@R1# commit and-quit
commit complete
Exiting configuration mode
```

3. 最后, 在 R3 上为 loopback 配置 OSPF 并设为被动:

```
cjones@R3> configure
Entering configuration mode
[edit]
cjones@R3# set protocols ospf area 0 interface lo0.0 passive
[edit]
cjones@R3# commit and-quit
commit complete
Exiting configuration mode
```

验证 OSPF 中的 loopback 端□

要在 Junos 中验证 loopback 是否已经被公告到 OSPF 中, 可以检查 loopback 端口的 OSPF 状态, 也可以域中其它路由器的路由表。

这是你在本书中首次见到 Junos 的路由表, 在 Junos 中没有分类网络(classful network)的概念, 因此你不会看到 IOS 中类似“x.x.x.x/xx is subnetted”的输出。

1. 在 R1 上确认 loopback (1.1.1.1/32)已经配置了 OSPF, R2 和 R3 已经学习到其地址:

```
cjones@R1> show ospf interface brief | match lo0.0
lo0.0          DRother 0.0.0.0          0.0.0.0          0.0.0.0          0

cjones@R2> show route protocol ospf terse 1.1.1.1
inet 0: 9 destinations, 9 routes (9 active, 0 holddown, 0 hidden)
+ = Active Route, - = Last Active, * = Both
A Destination      P Prf  Metric 1    Metric 2    Next hop      AS path
* 1.1.1.1/32       O  10           1              >10.42.12.1

cjones@R3> show route protocol ospf terse 1.1.1.1
inet 0: 9 destinations, 9 routes (9 active, 0 holddown, 0 hidden)
+ = Active Route, - = Last Active, * = Both
A Destination      P Prf  Metric 1    Metric 2    Next hop      AS path
* 1.1.1.1/32       O  10           1              >10.42.13.1
```

2. 在 R2 上确认 loopback(2.2.2.2/32)已经配置了 OSPF, R1 和 R3 已经学习到其地址:

```
cjones@R2> show ospf interface brief | match lo0.0
lo0.0          DRother 0.0.0.0          0.0.0.0          0.0.0.0          0

cjones@R1> show route protocol ospf terse 2.2.2.2
inet 0: 9 destinations, 9 routes (9 active, 0 holddown, 0 hidden)
+ = Active Route, - = Last Active, * = Both
A Destination      P Prf  Metric 1    Metric 2    Next hop      AS path
* 2.2.2.2/32       O  10           1              >10.42.12.2
```

```
cjones@R3> show route protocol ospf terse 2.2.2.2
inet 0: 9 destinations, 9 routes (9 active, 0 holddown, 0 hidden)
+ = Active Route, - = Last Active, * = Both
A Destination      P Prf  Metric 1    Metric 2 Next hop      AS path
* 2.2.2.2/32       O 10      1          >10.42.23.2
```

3. 在 R3 上确认 loopback(3.3.3.3/32)已经配置了 OSPF, R1 和 R2 已经学习到其地址:

```
cjones@R3> show ospf interface brief | match lo0.0
lo0.0                DRother 0.0.0.0          0.0.0.0          0.0.0.0          0
```

```
cjones@R1> show route protocol ospf terse 3.3.3.3
inet 0: 9 destinations, 9 routes (9 active, 0 holddown, 0 hidden)
+ = Active Route, - = Last Active, * = Both
A Destination      P Prf  Metric 1    Metric 2 Next hop      AS path
* 3.3.3.3/32       O 10      1          >10.42.13.3
```

```
cjones@R2> show route protocol ospf terse 3.3.3.3
inet 0: 9 destinations, 9 routes (9 active, 0 holddown, 0 hidden)
+ = Active Route, - = Last Active, * = Both
A Destination      P Prf  Metric 1    Metric 2 Next hop      AS path
* 3.3.3.3/32       O 10      1          >10.42.23.3
```

手动配置 OSPF router-id

Junos软件在rpd进程开始执行时(通常是设备启动时)将其侦测到的第一个非火星地址⁷(martian address)作为router-id, 这样的话通常路由器都会选择loopback端口的地址, 这样的行为在本质上与IOS是相同的。

1. 在 R1 上使用 show 命令查看当前的 router-id, 然后将其配置为 11.11.11.11:

```
cjones@R1> show ospf overview | match "Router ID"
Router ID: 1.1.1.1
cjones@R1> configure
Entering configuration mode
[edit]
cjones@R1# set routing-options router-id 11.11.11.11
[edit]
cjones@R1# commit and-quit
commit complete
Exiting configuration mode
```

2. 在 R2 上使用 show 命令查看当前的 router-id, 然后将其配置为 22.22.22.22:

```
cjones@R2> show ospf overview | match "Router ID"
Router ID: 2.2.2.2
cjones@R2> configure
Entering configuration mode
[edit]
cjones@R2# set routing-options router-id 22.22.22.22
[edit]
cjones@R1# commit and-quit
commit complete
Exiting configuration mode
```

3. 在 R3 上使用 show 命令查看当前的 router-id, 然后将其配置为 33.33.33.33:

```
cjones@R3> show ospf overview | match "Router ID"
Router ID: 3.3.3.3
cjones@R3> configure
```

⁷ 译者注: 简而言之 martian address 就是不可路由的地址, 例如 127.0.0.1。详情请参见 http://www.juniper.net/techpubs/en_US/junos12.2/topics/topic-map/martian-addresses.html。

```
Entering configuration mode
[edit]
cjones@R3# set routing-options router-id 33.33.33.33
[edit]
cjones@R1# commit and-quit
commit complete
Exiting configuration mode
```

验证手工配置的 OSPF router-id

1. 在 R1 上使用 show 命令验证 router-id:

```
cjones@R1> show ospf overview | match "Router ID"
Router ID: 11.11.11.11
```

2. 在 R2 上使用 show 命令验证 router-id:

```
cjones@R2> show ospf overview | match "Router ID"
Router ID: 22.22.22.22
```

3. 在 R3 上使用 show 命令验证 router-id:

```
cjones@R3> show ospf overview | match "Router ID"
Router ID: 33.33.33.33
```

小结

本节中的任务是一个简单的 OSPF 网络配置,但是我们已经可以看到,相对于 IOS 使用几乎过时的 `network` 命令来暗示哪个端口运行 OSPF,Junos 允许网络管理员在 `prorocol` 层级来指定哪个端口运行 OSPF。而且再强调一下,这些看似细小的功能扩展会随着网络规模的扩大而大放异彩。

EBGP

在这个例子里，我们配置两个自治域(Autonomous Systems - AS)之间简单的 EBGP 对等体会话。你不仅可以学习到 Junos 的方法，而且还能了解其策略配置的统一框架。

我们使用两台直连的路由器，EBGP 对等体将使用物理端口来建立，下面是拓扑图示：

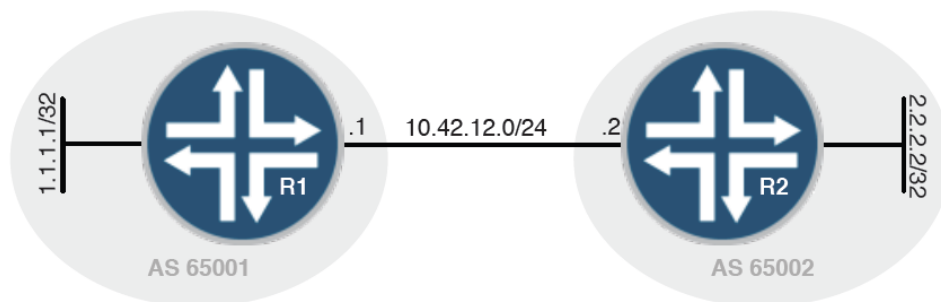


图2.2 EBGP 拓扑

本节中我们的任务是：

- 在 AS 65001 的 R1 和 AS 65002 的 R2 之间配置 EBGP 对等体；
- 使用物理端口建立对等关系；
- 将 loopback 端口公告到对方。

IOS 配置

首先我们使用 IOS 来配置网络。

配置 IOS 路由器以完成初始连接

1. 在 R1 的端口上设置 IP 地址：

```
R1# configure terminal
R1(config)# interface loopback 0
R1(config-if)# ip address 1.1.1.1 255.255.255.255
R1(config-if)# no shutdown
R1(config-if)# interface FastEthernet 0/0
R1(config-if)# ip address 10.42.12.1 255.255.255.0
R1(config-if)# no shutdown
R1(config-if)# end
R1#
```

2. 在 R2 的端口上设置 IP 地址：

```
R2# configure terminal
R2(config)# interface loopback 0
R2(config-if)# ip address 2.2.2.2 255.255.255.255
R2(config-if)# no shutdown
R2(config-if)# interface FastEthernet 0/0
R2(config-if)# ip address 10.42.12.2 255.255.255.0
R2(config-if)# no shutdown
R2(config-if)# end
R2#
```

验证 IOS 路由器的初始连接

1. 从 R1 ping R2:


```
R1#ping 10.42.12.2
Type escape sequence to abort.
Sending 5, 100-byte ICMP Echos to 10.42.12.2, timeout is 2 seconds:
!!!!
Success rate is 100 percent (5/5), round-trip min/avg/max = 1/2/4 ms
R1#
```

配置 EBGp 使用物理端口建立对等体

1. 在 R1 上配置 EBGp 使其与 AS 65002 中的 R2 对等:

```
R1#configure terminal
R1(config)#router bgp 65001
R1(config-router)# no synchronization
R1(config-router)# neighbor 10.42.12.2 remote-as 65002
R1(config-router)# no auto-summary
R1(config-router)#end
R1#
```

2. 在 R2 上配置 EBGp 使其与 AS 65001 中的 R1 对等:

```
R2#configure terminal
R2(config)#router bgp 65002
R2(config-router)# no synchronization
R2(config-router)# neighbor 10.42.12.1 remote-as 65001
R2(config-router)# no auto-summary
R2(config-router)#end
R2#
```

验证 EBGp 对等体

1. 在 R1 上检验 EBGp 对等体状态为 Established:

```
R1#show ip bgp neighbors
BGP neighbor is 10.42.12.2, remote AS 65002, external link
  BGP version 4, remote router ID 2.2.2.2
  BGP state = Established, up for 00:00:30
  Last read 00:00:30, last write 00:00:30, hold time is 180, keepalive interval is 60 seconds
...
```

2. 在 R2 上检验 EBGp 对等体状态为 Established:

```
R2#show ip bgp neighbors
BGP neighbor is 10.42.12.1, remote AS 65001, external link
  BGP version 4, remote router ID 1.1.1.1
  BGP state = Established, up for 00:01:57
  Last read 00:00:57, last write 00:00:57, hold time is 180, keepalive interval is 60 seconds
...
```

配置 EBGp 公告 loopback 地址

1. 在 R1 上配置 BGP 进程, 使其公告 loopback 地址:

```
R1#configure terminal
R1(config)#router bgp 65001
R1(config-router)#network 1.1.1.1 mask 255.255.255.255
R1(config-router)#end
R1#
```

2. 在 R2 上配置 BGP 进程, 使其公告 loopback 地址:

```
R2#configure terminal
R2(config)#router bgp 65002
R2(config-router)#network 2.2.2.2 mask 255.255.255.255
R2(config-router)#end
R2#
```

验证 EBGP 公告了 loopback 地址

1. 在 R1 上执行 `show ip bgp summary` 命令，确定其是否收到任何 BGP 前缀:

```
R1#show ip bgp sum | include Nei|65002
Neighbor      V      AS MsgRcvd MsgSent   TblVer  InQ OutQ Up/Down  State/PfxRcd
10.42.12.2     4 65002      9       9        3    0    0 00:05:16      1
```

2. 在 R1 上显示其接收到的路由:

```
R1#show ip bgp regexp ^65002
BGP table version is 3, local router ID is 1.1.1.1
Status codes: s suppressed, d damped, h history, * valid, > best, i - internal,
               r RIB-failure, S Stale
Origin codes: i - IGP, e - EGP, ? - incomplete
   Network        Next Hop           Metric LocPrf Weight Path
*> 2.2.2.2/32     10.42.12.2             0                   0 65002 i
```

3. 在 R1 上显示 RIB:

```
R1#show ip route bgp | exclude subnetted
B          2.2.2.2 [20/0] via 10.42.12.2, 00:05:30
```

4. 在 R2 上显示 BGP 邻居汇总:

```
R2#show ip bgp summary | include Nei|65001
Neighbor      V      AS MsgRcvd MsgSent   TblVer  InQ OutQ Up/Down  State/PfxRcd
10.42.12.1     4 65001      6       6        3    0    0 00:02:24      1
```

5. 在 R2 上显示其接收到的路由:

```
R2#show ip bgp regexp ^65001
BGP table version is 3, local router ID is 2.2.2.2
Status codes: s suppressed, d damped, h history, * valid, > best, i - internal,
               r RIB-failure, S Stale
Origin codes: i - IGP, e - EGP, ? - incomplete
   Network        Next Hop           Metric LocPrf Weight Path
*> 1.1.1.1/32     10.42.12.1             0                   0 65001 i
```

6. 在 R2 上显示 RIB:

```
R2#show ip route bgp | exclude subnetted
B          1.1.1.1 [20/0] via 10.42.12.1, 00:07:17
```

Junos 配置

现在我们在 Junos 路由器上配置同样的网络，看看有什么不同。

配置 Junos 路由器以完成初始连接

1. 在 R1 上为端口设置 IP 地址:

```
[edit]
cjones@R1# set interfaces fe-0/0/0.0 family inet address 10.42.12.1/24
[edit]
cjones@R1# set interfaces lo0.0 family inet address 1.1.1.1/32
[edit]
cjones@R1# commit and-quit
commit.complete
Exiting.configuration.mode
```

2. 检查 R1 上的端口配置:

```
cjones@R1> show configuration interfaces
fe-0/0/0 {
    unit 0 {
        family inet {
```

```

        address 10.42.12.1/24;
    }
}
lo0 {
    unit 0 {
        family inet {
            address 1.1.1.1/32;
        }
    }
}

```

3. 在 R2 上为端口配置 IP 地址:

```

[edit]
cjones@R2# set interfaces fe-0/0/0.0 family inet address 10.42.12.2/24
[edit]
cjones@R2# set interfaces lo0.0 family inet address 2.2.2.2/32
[edit]
cjones@R2# commit and-quit
commit complete
Exiting configuration mode

```

4. 检查 R2 上的端口配置:

```

cjones@R2> show configuration interfaces
fe-0/0/0 {
    unit 0 {
        family inet {
            address 10.42.12.2/24;
        }
    }
}
lo0 {
    unit 0 {
        family inet {
            address 2.2.2.2/32;
        }
    }
}

```

5. 现在从 R1 ping R2 来验证一下初始连接:

```

cjones@R1> ping 10.42.12.2 rapid
PING 10.42.12.2 (10.42.12.2): 56 data bytes
!!!!
--- 10.42.12.2 ping statistics ---
5 packets transmitted, 5 packets received, 0% packet loss
round-trip min/avg/max/stddev = 2.238/2.791/4.261/0.752 ms

```

配置 EBGp 使用物理端口建立对等体

与 IOS 相比，在 Junos 中 BGP 配置的最大改变是 BGP 邻居通常用一条命令就可以配置完毕(路由器自身的 AS 号是另外配置的)。

1. 在 R1 上配置 EBGp 使其与 AS 65002 中的 R2 对等:

```

[edit]
cjones@R1# set routing-options autonomous-system 65001
[edit]
cjones@R1# set protocols bgp group EBGp neighbor 10.42.12.2 peer-as 65002
[edit]
cjones@R1# commit and-quit
commit complete
Exiting configuration mode

```

2. 检查 R1 上的 BGP 配置:

```
cjones@R1> show configuration routing-options
autonomous-system 65001;
cjones@R1> show configuration protocols bgp
group EBGp {
    neighbor 10.42.12.2 {
        peer-as 65002;
    }
}
```

3. 在 R2 上配置 EBGp 使其与 AS 65001 中的 R1 对等:

```
[edit]
cjones@R2 set routing-options autonomous-system 65002
[edit]
cjones@R2 set protocols bgp group EBGp neighbor 10.42.12.1 peer-as 65001
[edit]
cjones@R2 commit and-quit
commit complete
Exiting configuration mode
```

4. 检查 R2 上的 BGP 配置:

```
cjones@R2> show configuration routing-options
autonomous-system 65002;
cjones@R2> show configuration protocols bgp
group EBGp {
    neighbor 10.42.12.1 {
        peer-as 65001;
    }
}
```

验证使用物理端口建立的 EBGp 对等体

1. 在 R1 上检验 EBGp 对等体状态为 Established:

```
cjones@R1> show bgp neighbor 10.42.12.2 | match Established
Type: External      State: Established      Flags: <ImportEval Sync>
```

2. 在 R2 上检验 EBGp 对等体状态为 Established:

```
cjones@R2> show bgp neighbor 10.42.12.1 | match Established
Type: External      State: Established      Flags: <ImportEval Sync>
```

配置 EBGp 公告 loopback 地址

一旦确认了 BGP 对等关系已经建立，我们就可以继续下一步的配置。为了验证 BGP 工作正常，我们将一些地址前缀向 BGP 对等体进行公告。

正如多数的 Junos 路由操作，这是通过路由策略来实现的。

1. 首先我们在 R1 上创建一条匹配其 loopback 端口地址的策略，以便随后将其应用到 BGP 进程:

```
[edit]
cjones@R1# edit policy-options policy-statement EBGp_ADVERTISE_LOOPBACK
[edit policy-options policy-statement EBGp_ADVERTISE_LOOPBACK]
cjones@R1# set term ADVERTISE_LO0 from protocol direct
[edit policy-options policy-statement EBGp_ADVERTISE_LOOPBACK]
cjones@R1# set term ADVERTISE_LO0 from route-filter 1.1.1.1/32 exact
[edit policy-options policy-statement EBGp_ADVERTISE_LOOPBACK]
cjones@R1# set term ADVERTISE_LO0 then accept
[edit policy-options policy-statement EBGp_ADVERTISE_LOOPBACK]
cjones@R1# top
[edit]
cjones@R1# commit and-quit
```

```
commit complete
Exiting configuration mode
```

2. 检查 R1 的策略配置:

```
cjones@R1> show configuration policy-options
policy-statement EBGp_ADVERTISE_LOOPBACK {
  term ADVERTISE_LO0 {
    from {
      protocol direct;
      route-filter 1.1.1.1/32 exact;
    }
    then accept;
  }
}
```

3. 在 R1 上将该策略配置为 BGP 对等体的输出策略:

```
cjones@R1> configure
Entering configuration mode
[edit]
cjones@R1# set protocols bgp group EBGp neighbor 10.42.12.2 export EBGp_ADVERTISE_LOOPBACK
[edit]
cjones@R1# commit and-quit
commit complete
Exiting configuration mode
```

4. 检查 R1 的 BGP 输出策略:

```
cjones@R1> show configuration protocols bgp
group EBGp {
  neighbor 10.42.12.2 {
    export EBGp_ADVERTISE_LOOPBACK;
    peer-as 65002;
  }
}
```

5. 接着我们在 R2 上创建一条匹配其 loopback 端口地址的策略, 以便随后将其应用到 BGP 进程:

```
[edit]
cjones@R2# edit policy-options policy-statement EBGp_ADVERTISE_LOOPBACK
[edit policy-options policy-statement EBGp_ADVERTISE_LOOPBACK]
cjones@R2# set term ADVERTISE_LO0 from protocol direct
[edit policy-options policy-statement EBGp_ADVERTISE_LOOPBACK]
cjones@R2# set term ADVERTISE_LO0 from route-filter 2.2.2.2/32 exact
[edit policy-options policy-statement EBGp_ADVERTISE_LOOPBACK]
cjones@R2# set term ADVERTISE_LO0 then accept
[edit policy-options policy-statement EBGp_ADVERTISE_LOOPBACK]
cjones@R2# top
[edit]
cjones@R2# commit and-quit
commit complete
Exiting configuration mode
```

6. 检查 R2 的策略配置:

```
cjones@R2> show configuration policy-options
policy-statement EBGp_ADVERTISE_LOOPBACK {
  term ADVERTISE_LO0 {
    from {
      protocol direct;
      route-filter 2.2.2.2/32 exact;
    }
    then accept;
  }
}
```

```
}
```

7. 在 R2 上将该策略配置为 BGP 对等体的输出策略:

```
cjones@R2> configure
Entering configuration mode
[edit]
cjones@R2# set protocols bgp group EBGp neighbor 10.42.12.1 export EBGp_ADVERTISE_LOOPBACK
[edit]
cjones@R2# commit and-quit
commit complete
Exiting configuration mode
```

8. 检查 R2 的 BGP 输出策略:

```
cjones@R2> show configuration protocols bgp
group EBGp {
    neighbor 10.42.12.1 {
        export EBGp_ADVERTISE_LOOPBACK;
        peer-as 65001;
    }
}
```

验证 EBGp 公告了 loopback 地址

通过检查 BGP 汇总信息和 RIB(显示了在应用任何输入策略之前收到的 BGP 前缀列表)可以验证我们的 BGP 输出策略是否工作正常。另外,也可以对路由表进行一次快速的检查来确认预期的前缀已经被载入。

1. 在 R1 上显示 BGP 汇总, 确定其是否收到任何 BGP 前缀::

```
cjones@R1> show bgp summary
Groups: 1 Peers: 1 Down peers: 0
Table Tot Paths Act Paths Suppressed History Damp State Pending
inet 0 1 1 0 0 0 0
Peer AS InPkt OutPkt OutQ Flaps Last Up/Dwn State|#Active/Received/Accepted/Damped
10.42.12.2 65002 12 13 0 0 4:04 1/1/1/0 0/0/0/0
```

2. 在 R1 上显示其接收到的路由::

```
cjones@R1> show route receive-protocol bgp 10.42.12.2
inet 0: 6 destinations, 6 routes (6 active, 0 holddown, 0 hidden)
Prefix Nexthop MED Lclpref AS path
* 2.2.2.2/32 10.42.12.2 65002 I
```

3. 在 R1 上显示 RIB:

```
cjones@R1> show route protocol bgp
inet 0: 6 destinations, 6 routes (6 active, 0 holddown, 0 hidden)
+ = Active Route, - = Last Active, * = Both
2.2.2.2/32 *[BGP/170] 00:05:22, localpref 100
AS path: 65002 I
> to 10.42.12.2 via fe-0/0/0.0
```

4. 在 R2 上显示 BGP 汇总, 确定其是否收到任何 BGP 前缀::

```
cjones@R2> show bgp summary
Groups: 1 Peers: 1 Down peers: 0
Table Tot Paths Act Paths Suppressed History Damp State Pending
inet 0 1 1 0 0 0 0
Peer AS InPkt OutPkt OutQ Flaps Last Up/Dwn State|#Active/Received/Accepted/Damped
10.42.12.1 65001 15 15 0 0 7:21 1/1/1/0 0/0/0/0
```

5. 在 R2 上显示其接收到的路由::

```
cjones@R2> show route receive-protocol bgp 10.42.12.1
inet 0: 6 destinations, 6 routes (6 active, 0 holddown, 0 hidden)
```

Prefix	Nexthop	MED	Lclpref	AS path
* 1.1.1.1/32	10.42.12.1			65001 I

6. 在 R2 上显示 RIB:

```
cjones@R2> show route protocol bgp
inet 0: 6 destinations, 6 routes (6 active, 0 holddown, 0 hidden)
+ = Active Route, - = Last Active, * = Both
1.1.1.1/32      *[BGP/170] 00:16:32, localpref 100
                AS path: 65001 I
                > to 10.42.12.1 via fe-0/0/0.0
```

小结

设置自治域号, 设置对等体 – 仅仅两条语句就能够配置一个简单的 BGP 对等体, 很明显 Junos 又领先了。虽然在 IOS 中可以直接在 BGP 进程下配置路由公告, 但是毫无疑问 Junos 的策略配置才是真正的亮点。统一的策略配置框架是 Junos 可以在你的网络中大放异彩的一个明显例子。当网络规模扩大或者需要新的服务时, 这些优势也将更为突出。

另一个值得注意的是同步的概念, IOS 规定, 从 IGP 学到的路由在通过 BGP 发布之前, 必须在路由表中是活动的。同步通常都被禁用。在 JUNOS 没有这样的规定。

IBGP

本节中我们配置 IBGP。IBGP 对等关系是在 loopback 地址之间建立的，这就需要我们添加到远端 loopback 地址的静态路由使之可达。IOS 工程师需要留意 Junos 将 BGP 对等体会话组合在一起的能力。

在此我们使用如图 2.3 所示在同一个 AS 中直连的两台路由器。

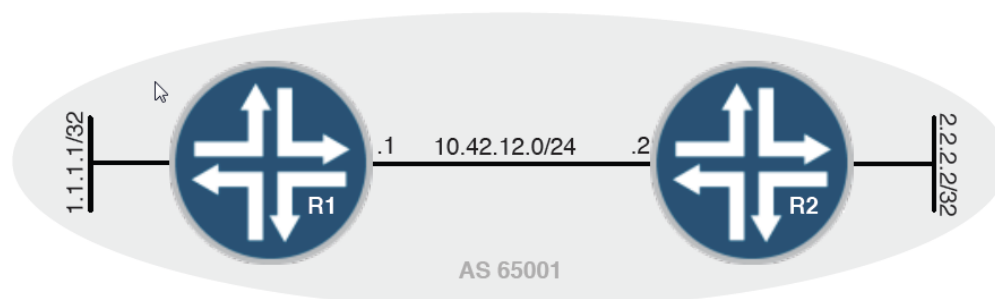


图2.3 IBGP 拓扑

在本节中我们的任务是：

- 在 R1 和 R2 上配置静态路由以实现 loopback 地址的互通；
- 在 R1 和 R2 上配置 AS 65001 内的对等体关系；
- 使用 loopback 地址形成 IBGP 对等。

IOS 配置

首先我们使用 IOS 来配置这个网络，以下的步骤对于你而言应该是很熟悉的了。

配置 IOS 路由器以完成初始连接

1. 在 R1 的端口上设置 IP 地址：

```
R1# configure terminal
R1(config)# interface loopback 0
R1(config-if)# ip address 1.1.1.1 255.255.255.255
R1(config-if)# no shutdown
R1(config-if)# interface FastEthernet 0/0
R1(config-if)# ip address 10.42.12.1 255.255.255.0
R1(config-if)# no shutdown
R1(config-if)# end
R1#
```

2. 在 R2 的端口上设置 IP 地址：

```
R2# configure terminal
R2(config)# interface loopback 0
R2(config-if)# ip address 2.2.2.2 255.255.255.255
R2(config-if)# no shutdown
R2(config-if)# interface FastEthernet 0/0
R2(config-if)# ip address 10.42.12.2 255.255.255.0
R2(config-if)# no shutdown
R2(config-if)# end
R2#
```

验证 IOS 路由器的初始连接

1. 从 R1 ping R2:

```
R1#ping 10.42.12.2
```



```
Type escape sequence to abort.
Sending 5, 100-byte ICMP Echos to 10.42.12.2, timeout is 2 seconds:
!!!!
Success rate is 100 percent (5/5), round-trip min/avg/max = 1/2/4 ms
R1#
```

为全互通配置静态路由

1. 在 R1 上配置到 R2 的 loopback 地址的静态路由，指定下一跳为 R2 连接 R1 的物理端口地址：

```
R1#configure terminal
R1(config)#ip route 2.2.2.2 255.255.255.255 10.42.12.2
R1(config)#end
R1#
```

2. 在 R2 上配置到 R1 的 loopback 地址的静态路由，指定下一跳为 R1 连接 R2 的物理端口地址：

```
R2#configure terminal
R2(config)#ip route 1.1.1.1 255.255.255.255 10.42.12.1
R2(config)#end
R2#
```

验证静态路由生效，全连接建立

1. 在 R1 上显示 RIB:

```
R1#show ip route static | exclude subnett
S      2.2.2.2 [1/0] via 10.42.12.2
R1#
```

2. 在 R2 上显示 RIB:

```
R2#show ip route static | exclude subnett
S      1.1.1.1 [1/0] via 10.42.12.1
R2#
```

3. 从 R1 ping R2 的 loopback:

```
R1#ping 2.2.2.2
Type escape sequence to abort.
Sending 5, 100-byte ICMP Echos to 2.2.2.2, timeout is 2 seconds:
!!!!
Success rate is 100 percent (5/5), round-trip min/avg/max = 20/22/24 ms
R1#
```

4. 从 R2 ping R1 的 loopback:

```
R2#ping 1.1.1.1
Type escape sequence to abort.
Sending 5, 100-byte ICMP Echos to 1.1.1.1, timeout is 2 seconds:
!!!!
Success rate is 100 percent (5/5), round-trip min/avg/max = 8/16/20 ms
R2#
```

配置 loopback 之间的 IBGP 对等体

1. 在 R1 上配置 IBGP:

```
R1#configure terminal
R1(config)#router bgp 65001
R1(config-router)#no auto-summary
R1(config-router)#no synchronization
R1(config-router)#neighbor 2.2.2.2 remote-as 65001
R1(config-router)#neighbor 2.2.2.2 update-source lo0
R1(config-router)#end
```

R1#

2. 在 R2 上配置 IBGP:

```
R2#configure terminal
R2(config)#router bgp 65001
R2(config-router)#no auto-summary
R2(config-router)#no synchronization
R2(config-router)#neighbor 1.1.1.1 remote-as 65001
R2(config-router)#neighbor 1.1.1.1 update-source lo0
R2(config-router)#end
R2#
```

验证 loopback 之间的对等关系已经建立

1. 在 R1 上检验 IBGP 对等体状态为 Established:

```
R1#show ip bgp neighbors
BGP neighbor is 2.2.2.2, remote AS 65001, internal link
  BGP version 4, remote router ID 2.2.2.2
  BGP state = Established, up for 00:01:01
  Last read 00:00:01, last write 00:00:01, hold time is 180, keepalive interval is 60 seconds
...
```

2. 在 R2 上检验 IBGP 对等体状态为 Established:

```
R2#show ip bgp neighbors
BGP neighbor is 1.1.1.1, remote AS 65001, internal link
  BGP version 4, remote router ID 1.1.1.1
  BGP state = Established, up for 00:02:42
  Last read 00:00:42, last write 00:00:42, hold time is 180, keepalive interval is 60 seconds
...
```

Junos 配置

现在让我们的 JUNOS 路由器上进行相同的配置。你会发现这个配置与之前的 EBGP 配置几乎相同，主要区别在于现在的类型是内部的。另外请注意我们使用了 local-address 配置选项来指定 IBGP 消息源于 loopback 地址。

IOS 工程师可能会注意到在 Junos 中我们没有为 IBGP 邻居配置 AS 号。这是因为 Junos 显式定义了该 BGP 对等体的类型是内部的。而在 IOS 中，类型是通过将对等体的 AS 配置为与本地路由器相同来隐式定义的。

配置 Junos 路由器以完成初始连接

1. 首先，在 R1 的端口上设置 IP 地址:

```
cjones@R1> configure
Entering configuration mode
[edit]
cjones@R1# set interfaces fe-0/0/0.0 family inet address 10.42.12.1/24
[edit]
cjones@R1# set interfaces lo0.0 family inet address 1.1.1.1/32
[edit]
cjones@R1# commit and-quit
commit complete
Exiting configuration mode
```

2. 检查一下 R1 上的端口配置:

```
cjones@R1> show configuration interfaces
fe-0/0/0 {
    unit 0 {
        family inet {
```

```

        address 10.42.12.1/24;
    }
}
lo0 {
    unit 0 {
        family inet {
            address 1.1.1.1/32;
        }
    }
}
}

```

3. 接着，在 R2 的端口上设置 IP 地址：

```

[edit]
cjones@R2# set interfaces fe-0/0/0.0 family inet address 10.42.12.2/24
[edit]
cjones@R2# set interfaces lo0.0 family inet address 2.2.2.2/32
[edit]
cjones@R2# commit and-quit
commit complete
Exiting configuration mode

```

4. 检查一下 R2 上的端口配置：

```

cjones@R2> show configuration interfaces
fe-0/0/0 {
    unit 0 {
        family inet {
            address 10.42.12.2/24;
        }
    }
}
lo0 {
    unit 0 {
        family inet {
            address 2.2.2.2/32;
        }
    }
}

```

5. 最后验证一下初始连接，从 R1 ping R2：

```

cjones@R1> ping 10.42.12.2 rapid
PING 10.42.12.2 (10.42.12.2): 56 data bytes
!!!!
--- 10.42.12.2 ping statistics ---
5 packets transmitted, 5 packets received, 0% packet loss
round-trip min/avg/max/stddev = 2.238/2.791/4.261/0.752 ms

```

为全互通配置静态路由

在 Junos 中配置静态路由的方式与在 IOS 中是有差异的。

1. 在 R1 上配置到 R2 的 loopback 地址的静态路由，指定下一跳为 R2 连接 R1 的物理端口地址：

```

cjones@R1> configure
Entering configuration mode
[edit]
cjones@R1# set routing-options static route 2.2.2.2/32 next-hop 10.42.12.2
[edit]
cjones@R1# commit and-quit
commit complete
Exiting configuration mode

```

2. 检查 R1 上的静态路由配置:

```
cjones@R1> show configuration routing-options
static {
    route 2.2.2.2/32 next-hop 10.42.12.2;
}
```

3. 在 R2 上配置到 R1 的 loopback 地址的静态路由, 指定下一跳为 R1 连接 R2 的物理端口地址:

```
cjones@R2> configure
Entering configuration mode
[edit]
cjones@R2# set routing-options static route 1.1.1.1/32 next-hop 10.42.12.1
[edit]
cjones@R2# commit and-quit
commit complete
Exiting configuration mode
```

4. 检查 R2 上的静态路由配置:

```
cjones@R2> show configuration routing-options
static {
    route 1.1.1.1/32 next-hop 10.42.12.1;
}
```

验证静态路由生效, 全连接建立

1. 在 R1 上显示 RIB:

```
cjones@R1> show route
inet 0: 4 destinations, 4 routes (4 active, 0 holddown, 0 hidden)
+ = Active Route, - = Last Active, * = Both
1.1.1.1/32          *[Direct/0] 00:09:23
                    > via lo0.0
2.2.2.2/32          *[Static/5] 00:06:02
                    > to 10.42.12.2 via fe-0/0/0.0
10.42.12.0/24       *[Direct/0] 00:09:23
                    > via fe-0/0/0.0
10.42.12.1/32       *[Local/0] 00:09:23
                    Local via fe-0/0/0.0
```

2. 在 R2 上显示 RIB:

```
cjones@R2> show route
inet 0: 4 destinations, 4 routes (4 active, 0 holddown, 0 hidden)
+ = Active Route, - = Last Active, * = Both
1.1.1.1/32          *[Static/5] 00:07:27
                    > to 10.42.12.1 via fe-0/0/0.0
2.2.2.2/32          *[Direct/0] 00:12:11
                    > via lo0.0
10.42.12.0/24       *[Direct/0] 00:08:16
                    > via fe-0/0/0.0
10.42.12.2/32       *[Local/0] 00:10:04
                    Local via fe-0/0/0.0
```

3. 从 R1 ping R2 的 loopback:

```
cjones@R1> ping 2.2.2.2 rapid
PING 2.2.2.2 (2.2.2.2): 56 data bytes
!!!!
--- 2.2.2.2 ping statistics ---
5 packets transmitted, 5 packets received, 0% packet loss
round-trip min/avg/max/stddev = 0.244/0.370/0.575/0.117 ms
```

4. 从 R2 ping R1 的 loopback:

```
cjones@R2> ping 1.1.1.1 rapid
PING 1.1.1.1 (1.1.1.1): 56 data bytes
!!!!
--- 1.1.1.1 ping statistics ---
5 packets transmitted, 5 packets received, 0% packet loss
round-trip min/avg/max/stddev = 0.374/0.449/0.510/0.045 ms
```

配置 loopback 之间的 IBGP 对等体

这里我们可以再一次看到 Junos 中 BGP 的配置是很简单的，只需要一条命令来配置自身的 AS 号，一条命令来配置对等体即可。

1. 在 R1 上配置 IBGP:

```
cjones@R1> configure
Entering configuration mode
[edit]
cjones@R1# set routing-options autonomous-system 65001
[edit]
cjones@R1# set protocols bgp group IBGP type internal neighbor 2.2.2.2 local-address 1.1.1.1
[edit]
cjones@R1# commit and-quit
commit complete
Exiting configuration mode
```

2. 检查 R1 上的 IBGP 配置:

```
cjones@R1> show configuration routing-options autonomous-system
65001;
cjones@R1> show configuration protocols bgp
group IBGP {
    type internal;
    neighbor 2.2.2.2 {
        local-address 1.1.1.1;
        peer-as 65001;
    }
}
```

3. 在 R2 上配置 IBGP:

```
cjones@R2> configure
Entering configuration mode
[edit]
cjones@R2# set routing-options autonomous-system 65001
[edit]
cjones@R2# set protocols bgp group IBGP type internal neighbor 1.1.1.1 local-address 2.2.2.2
[edit]
cjones@R2# commit and-quit
commit complete
Exiting configuration mode
```

验证 loopback 之间的对等关系已经建立

验证 IBGP 已经正常工作的步骤与验证 EBGP 是相同的，检查对等体状态是否为 Established 即可。

1. 在 R1 上检验 IBGP 对等体状态为 Established:

```
cjones@R1> show bgp neighbor 2.2.2.2 | match Established
Type: Internal    State: Established    Flags: <ImportEval Sync>
```

2. 在 R2 上检验 IBGP 对等体状态为 Established:

```
cjones@R2> show bgp neighbor 1.1.1.1 | match Established
Type: Internal    State: Established    Flags: <ImportEval Sync>
```

小结

在 Junos 中配置 IBGP 与配置 EBGP 几乎是相同的，主要的区别是使用了 `internal` 类型，且 `peer-as` 与本地 AS 吻合。但其实 `peer-as` 命令是不需要的，当类型为 `internal` 时，对等体 AS 与本地 AS 就是相同的。

Junos 的优势之一是能够将 BGP 对等体会话组合在一起，这样不管出于什么目的，你可以将 IBGP 和 EBGP 对等会话或者运营商连接进行逻辑分离。

还要注意 Junos 不需要一个与 IOS 中 `auto summary` 等价的命令，这是因为 Junos 没有分级或无类别网络的概念，只使用 CIDR。

配置 VLAN

我们来完成一些基本的交换机配置，比较一下两种操作系统的差异。对于 IOS 工程师，要留意的是 Junos 的 VLAN 是如何命名的，VLAN id 是如何在 VLAN 层级下配置的。

这里使用了单台三层交换机，所以不需要图示。

本节中我们的任务是：

- 配置一个 VLAN；
- 将新的 VLAN 分配给接入(untagged)端口；
- 创建一个中继(802.1Q tagged)端口；
- 创建一个三层端口。

IOS 配置

首先我们使用 IOS 来完成配置，以下的步骤对于你而言应该是很熟悉的了。

在 IOS 交换机上配置 VLAN

1. 很简单，创建一个 VLAN，并赋予它一个名字：

```
SW1#configure terminal
SW1(config)#vlan 10
SW1(config-vlan)#name IOS_VLAN
SW1(config-vlan)#exit
SW1(config)#end
SW1#
```

验证 VLAN 已被创建并命名

1. 显示 VLAN 列表：

```
SW1#show vlan brief
```

VLAN Name	Status	Ports
1 default	active	Fa0/1, Fa0/2, Fa0/3, Fa0/4 Fa0/5, Fa0/6, Fa0/7, Fa0/8 Fa0/9, Fa0/10, Fa0/11, Fa0/12 Fa0/13, Fa0/14, Fa0/15, Fa0/16 Fa0/17, Fa0/18, Fa0/19, Fa0/20 Fa0/21, Fa0/22, Gi0/1, Gi0/2
10 IOS_VLAN	active	

```
SW1#
```

将 VLAN 分配给接入端口

1. 将端口配置为接入模式，并制定其 VLAN 为 10：

```
SW1#configure terminal
SW1(config)#interface fa0/2
SW1(config-if)#switchport mode access
SW1(config-if)#switchport access vlan 10
SW1(config-if)#end
SW1#
```

验证 VLAN 已经被添加到非标签 fa0/2 端口

1. 再次显示 VLAN 列表：

```
SW1#show vlan brief
```

VLAN	Name	Status	Ports
1	default	active	Fa0/1, Fa0/3, Fa0/4 Fa0/5, Fa0/6, Fa0/7, Fa0/8 Fa0/9, Fa0/10, Fa0/11, Fa0/12 Fa0/13, Fa0/14, Fa0/15, Fa0/16 Fa0/17, Fa0/18, Fa0/19, Fa0/20 Fa0/21, Fa0/22, Gi0/1, Gi0/2
10	IOS_VLAN	active	Fa0/2

SW1#

创建中继端口承载所有 VLAN

1. 在 IOS 中，一个中继端口默认承载所有配置了的 VLAN，因此我们将一个端口配置为 802.1Q 标签端口如下：

```
SW1#configure terminal
SW1(config)#interface fa0/24
SW1(config-if)#switchport mode trunk
SW1(config-if)#switchport trunk encapsulation dot1q
SW1(config-if)#end
SW1#
```

验证中继端口

1. 显示中继端口以验证：

```
SW1#show interfaces trunk
Port      Mode      Encapsulation  Status        Native vlan
Fa0/24    on        802.1q         trunking      1
Port      Vlans allowed on trunk
Fa0/24    1-4094
Port      Vlans allowed and active in management domain
Fa0/24    1,10
Port      Vlans in spanning tree forwarding state and not pruned
Fa0/24    1,10
```

创建三层端口 (IOS 中称为 SVI)

1. 为 IOS_VLAN 配置三层端口：

```
SW1#configure terminal
SW1(config)#interface vlan 10
SW1(config-if)#ip address 10.10.10.1 255.255.255.0
SW1(config-if)#end
SW1#
```

验证三层端口

1. 在交换机上显示 IP 端口的列表即可完成验证工作：

```
SW1#show ip interfaces brief | include Vlan
Vlan1          unassigned      YES NVRAM  administratively down down
Vlan10         10 10 10 1      YES manual  up         up
```

Junos 配置

我们来看看在 Junos⁸ 中如何完成这些任务。IOS 工程师要留意 VLAN 是如何命名和配置的。

⁸ 译者注：本节内容适用于 SRX/EX 平台，M/T/MX 平台的配置有所差异，请参阅 http://www.juniper.net/techpubs/en_US/junos12.2/information-products/pathway-pages/config-guide-network-interfaces/ethernet-802-1q-vlans.html#configuration。

在 Junos 交换机上配置 VLAN

1. 创建 VLAN 并赋予其 VLAN id:

```
cjones@SW1> configure
Entering configuration mode
[edit]
cjones@SW1# set vlans JUNOS_VLAN vlan-id 10
[edit]
cjones@SW1# commit and-quit
commit complete
Exiting configuration mode
```

2. 检查 VLAN 配置:

```
cjones@SW1> show configuration vlans
JUNOS_VLAN {
    vlan-id 10;
}
```

验证 VLAN 以被创建并命名

显示 VLAN 列表:

```
cjones@SW1> show vlans JUNOS_VLAN
Name      Tag      Interfaces
JUNOS_VLAN 10
None
```

将接入端口放置到 VLAN 中

在 Junos 中有两种方式将 VLAN 与端口联系起来:

- 在 vlan 配置节中
- 在 interface 配置节中

这里我们将接入端口放置在 vlan 配置节下。要注意无论采用何种方式配置, interface 下都必须配置 ethernet-switching 协议族。

1. 配置端口的 ethernet-switching 协议族, 将其模式设置为 access; 创建 VLAN, 设置其 id 为 10, 并将端口放入 VLAN 中:

```
cjones@SW1> configure
Entering configuration mode
[edit]
cjones@SW1# set interfaces fe-0/0/0 unit 0 family ethernet-switching port-mode access
[edit]
cjones@SW1# set vlans JUNOS_VLAN vlan-id 10 interface fe-0/0/0.0
[edit]
cjones@SW1# commit and-quit
commit complete
Exiting configuration mode
```

2. 检查端口和 VLAN 配置:

```
cjones@SW1> show configuration interfaces fe-0/0/0.0
family ethernet-switching {
    port-mode access;
}
cjones@SW1> show configuration vlans
JUNOS_VLAN {
    vlan-id 10;
    interface {
        fe-0/0/0.0;
    }
}
```

}

验证 fe-0/0/0.0 已成为 VLAN 的无标签接入端口

1. 显示 VLAN 列表:

```
cjones@SW1> show vlans JUNOS_VLAN
Name      Tag      Interfaces
JUNOS_VLAN 10      fe-0/0/0.0
```

2. 显示 VLAN 列表的扩展输出以验证 fe-0/0/0.0 端口是无标签的:

```
cjones@SW1> show vlans extensive JUNOS_VLAN
VLAN: JUNOS_VLAN, Created at: Sun Jun 24 12:54:01 2012
802.1Q Tag: 10, Internal index: 2, Admin State: Enabled, Origin: Static
Protocol: Port Mode, Mac aging time: 300 seconds
Number of interfaces: Tagged 0 (Active = 0), Untagged 1 (Active = 0)
fe-0/0/0.0, untagged, access
```

3. 显示 ethernet-switching 端口详情:

```
cjones@SW1> show ethernet-switching interfaces detail.
Interface: fe-0/0/0.0 Index: 69.
. State: UP.
. Vlans: JUNOS_VLAN(untagged)
```

创建中继端口承载所有 VLAN

这次我们将vlan放置在端口配置下，而不是端口放置在vlan配置下。这样的做法对于IOS工程师来说可能更熟悉。但是无论你选用哪种方法，最重要的是保持一致性。⁹

1. 配置端口为 ethernet-switching 协议族 trunk 模式，将全部 VLAN 加入到这个端口下:

```
cjones@SW1> configure
Entering configuration mode
[edit]
cjones@SW1# set interfaces fe-0/0/0.0 family ethernet-switching port-mode trunk vlan members all10
[edit]
cjones@SW1# commit and-quit
commit complete
Exiting configuration mode
```

2. 检查中继端口配置:

```
cjones@SW1> show configuration interfaces fe-0/0/0.0
family ethernet-switching {
    port-mode trunk;
    vlan {
        members all;
    }
}
```

验证中继端口

1. 显示 VLAN 列表:

```
cjones@SW1> show vlans JUNOS_VLAN
Name      Tag      Interfaces
```

⁹ 译者注：为了易于 CLI 管理，我们建议对 VLAN 成员集中配置。对于接入端口，在 VLAN 下配置其所有成员；对于中继端口，在接口下配置其所属的所有 VLAN。

¹⁰ 译者注：all 是一个特殊关键字，代表了所有在交换机上配置了的 VLAN，你也可以在此指定 VLAN 的名字或者 id，可想而知 all 是不能用于定义 VLAN 名字的。

```
JUNOS_VLAN      10
                  fe-0/0/0.0
```

2. 显示 VLAN 列表的扩展输出以验证 fe-0/0/0.0 端口是带标签的:

```
cjones@SW1> show vlans extensive JUNOS_VLAN
VLAN: JUNOS_VLAN, Created at: Sun Jun 24 12:54:01 2012
802.1Q Tag: 10, Internal index: 2, Admin State: Enabled, Origin: Static
Protocol: Port Mode, Mac aging time: 300 seconds
Number of interfaces: Tagged 1 (Active = 0), Untagged 0 (Active = 0)
                  fe-0/0/0.0, tagged, trunk
```

3. 显示 ethernet-switching 端口详情:

```
cjones@SW1> show ethernet-switching interfaces detail
Interface: fe-0/0/6.0, Index: 84, State: up, Port mode: Trunk
Ether type for the interface: 0x8100
VLAN membership:
    JUNOS_VLAN, 802.1Q Tag: 10, tagged, msti-id: 0, blocked by STP
```

创建三层端口(在 Junos 中称为 RVI)

对于 IOS 工程师而言,配置路由虚拟端口(Routed Virtual Interface - RVI)可能看起来有些陌生,不过它实际上提供了一个与配置 Junos 端口相一致的处理手段。Junos 有一个 VLAN 端口,可以指定多个单元号;相比之下 IOS 需要为每个 VLAN 创建其各自的端口。

1. 配置 RVI(VLAN 端口),注意这里的单元号与其对应的 VLAN id 没有必然的联系,但是为了维护的方便,最好使它们保持一致:

```
cjones@SW1> configure
Entering configuration mode
[edit]
cjones@R1# set interfaces vlan unit 10 family inet address 10.10.10.1/24
```

2. 在 VLAN 下配置其对应的三层端口:

```
[edit]
cjones@SW1# set vlans JUNOS_VLAN 13-interface vlan.10
[edit]
cjones@SW1# commit and-quit
commit complete
Exiting configuration mode
```

3. 检查配置

```
cjones@SW1> show configuration interfaces vlan
unit 10 {
    family inet {
        address 10.10.10.1/24;
    }
}

cjones@R1> show configuration vlans
JUNOS_VLAN {
    vlan-id 10;
    13-interface vlan.10;
}
```

验证三层端口

在交换机上显示端口 IP 地址的列表来验证三层端口:

```
cjones@SW1> show interfaces terse | match vlan
vlan                up      up
vlan 10             up      up      inet      10.10.10.1/24
```

小结

对于两种操作系统而言，VLAN 和三层端口的基本配置都很简单。最大的区别在于 IOS 配置先创建 VLAN id 然后对其进行命名，而 Junos 则是先命名 VLAN 然后再配置其 VLAN id。

简单 NAT

在本节中，我们来配置在企业网络中最常见的两种 NAT：将源地址翻译为外部接口地址和将目的地址翻译为内部服务器地址。两种操作系统都可以完成这些工作，对 IOS 工程师而言只是需要留意一下过程。传言说 Junos 比 IOS 复杂，至于你们信不信，反正我是不信。

本节的拓扑中有一台路由器和一台内部的 Web 服务器，如图 2.4 所示。

本节中我们的任务是：

- 将 10.42.0.0/24 进行源地址翻译为外部接口地址 200.200.200.1；
- 对 200.200.200.1 端口 80 进行目的地址翻译为内部 Web 服务器地址 10.42.0.100 端口 80。

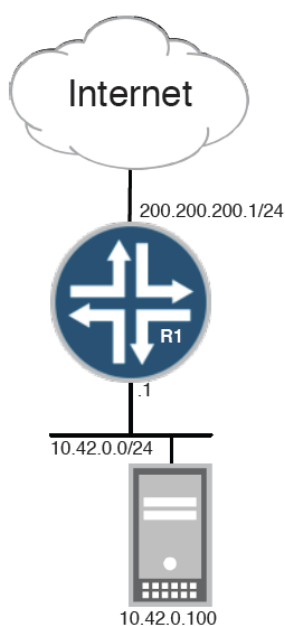


图2.4 NAT 拓扑

IOS 配置

在 IOS 中为 10.42.0.0/24 配置源地址翻译

```
R1#configure terminal
R1(config)#interface f0/0
R1(config-if)#ip address 200.200.200.1 255.255.255.0
R1(config-if)#ip nat outside
R1(config-if)#interface f0/1
R1(config-if)#ip address 10.42.0.1 255.255.255.0
R1(config-if)#ip nat inside
R1(config-if)#exit
R1(config)#access-list 1 permit 10.42.0.0 0.0.0.255
R1(config)#ip nat inside source list 1 interface FastEthernet 0/0 overload
R1(config)#end
R1#
```

验证 10.42.0.0/24 的源地址翻译

```
R1# show ip nat translations
Pro Inside global      Inside local      Outside local      Outside global
tcp 200.200.200.1:53638 10.42.0.2:53638 74.125.225.136:80 74.125.225.136:80
```

在 IOS 中为 Web 服务器配置目的地址翻译

```
R1# configure terminal
R1(config)# ip nat inside source static tcp 10.42.0.100 80 200.200.200.1 80 extendable
R1(config)# end
R1#
```

验证到 Web 服务器的目的地址翻译

```
R1#.show ip nat translations
Pro Inside global      Inside local      Outside local      Outside global
tcp 200.200.200.1:80    10.42.0.100:80    ---                ---
```

Junos 配置

现在让我们在Junos上完成同样的配置¹¹。要注意，设备必须处于流模式才能配置NAT，对于SRX设备而言这个是默认模式。在流模式下，我们采用防火墙安全策略控制流量。如何配置安全策略不在本书的讨论范围之内。

欲知更多？ 有关更多配置安全策略的信息，可以参看Rob Cameron和Brad Woodberg等合著的《JUNOS Security》。¹²

在 Junos 中为 10.42.0.0/24 配置源地址翻译

在 Junos 中的源地址翻译也是直截了当的，我们使用标准的策略 match/then 语法，同时指明在 NAT 过程中的安全域。

```
cjones@R1> configure
Entering configuration mode
[edit]
cjones@R1# set interfaces fe-0/0/0 unit 0 family inet address 200.200.200.1/24
[edit]
cjones@R1# set interfaces fe-0/0/1 unit 0 family inet address 10.42.0.1/24
[edit]
cjones@R1# edit security nat source rule-set LAN_RULE
[edit security nat source rule-set LAN_RULE]
cjones@R1# set from zone LAN
[edit security nat source rule-set LAN_RULE]
cjones@R1# set to zone INET
[edit security nat source rule-set LAN_RULE]
cjones@R1# set rule NAT_10.42_24 match source-address 10.42.0.0/24
[edit security nat source rule-set LAN_RULE]
cjones@R1# set rule NAT_10.42_24 match destination-address 0.0.0.0/0
[edit security nat source rule-set LAN_RULE]
cjones@R1# set rule NAT_10.42_24 then source-nat interface
[edit security nat source rule-set LAN_RULE]
cjones@R1# top
[edit]
cjones@R1#.commit and-quit
commit.complete
Exiting configuration mode
```

以上的 set 命令生成了如下配置：

```
interfaces {
```

¹¹译者注：本节内容适用于 SRX 平台，在 M/T/MX 平台上的配置有所差异，请参阅 http://www.juniper.net/techpubs/en_US/junos12.2/information-products/pathway-pages/service-s-interfaces/router-services-network-address-translation.html#configuration

¹²译者注：此书也无法从 Juniper 网站免费获得，因此你也可以参见两本免费的 Day One 小册子：《[Deploying SRX Series Services Gateways](#)》和《[Configuring SRX Series with J-Web](#)》

```

fe-0/0/0 {
    unit 0 {
        family inet {
            address 200.200.200.1/24;
        }
    }
}
fe-0/0/1 {
    unit 0 {
        family inet {
            address 10.42.0.1/24;
        }
    }
}
}
security {
    nat {
        source {
            rule-set LAN_RULE {
                from zone LAN;
                to zone INET;
            }
            rule NAT_ALL {
                match {
                    source-address 10.42.0.0/24;
                    destination-address 0.0.0.0/0;
                }
                then {
                    source-nat {
                        interface;
                    }
                }
            }
        }
    }
}

```

验证 10.42.0.0/24 的源地址翻译

1. 显示源地址翻译汇总:

```

cjones@R1> show security nat source summary
Total port number usage for port translation pool: 0
Maximum port number for port translation pool: 16777216
Total pools: 0
Total rules: 1

```

Rule name	Rule set	From	To	Action
NAT_ALL	LAN_RULE	LAN	INET	interface

2. 显示 10.42.0.100 的安全流会话:

```

cjones@R1> show security flow session nat source-prefix 10.42.0.100
Session ID: 14582, Policy name: PERMIT_ALL/4, Timeout: 1754, Valid
  In: 10.42.0.100/49401 -> 74.125.239.8/443;tcp, If: fe-0/0/1, Pkts: 10, Bytes: 394
  Out: 74.125.239.8/443 -> 200.200.200.1/15654;tcp, If: fe-0/0/0.0, Pkts: 8, Bytes: 3970
Session ID: 23401, Policy name: PERMIT_ALL/4, Timeout: 1786, Valid
  In: 10.42.0.100/49163 -> 207.17.137.239/80;tcp, If: fe-0/0/1, Pkts: 18, Bytes: 276
  Out: 207.17.137.239/80 -> 200.200.200.1/5973;tcp, If: fe-0/0/0.0, Pkts: 14, Bytes: 353
Total sessions: 2

```

在 Junos 中为 Web 服务器配置目的地址翻译

目的地址翻译在 Junos 中是很简单且直接的，很容易就想象得出在配置中会有什么，因为 Junos 安全策略是以 match/then 的格式来定义的。

对于 Junos 中的目的地址翻译来说，创建目的地址池是必需的。这个地址池指定了外部地址被翻译后所使用的内部地址。

一个目的地址翻译策略指定了匹配条件，以及在匹配成功之后所执行的动作，在本例中所执行的动作是 `destination-nat pool <poolname>`。

```
cjones@R1> configure
Entering configuration mode
[edit]
cjones@R1# set security nat destination pool WEBSERVER address 10.42.0.100/32
[edit]
cjones@R1# set security nat destination pool WEBSERVER address port 80
[edit]
cjones@R1# edit security nat destination rule-set DESTINATION_NAT
[edit security nat destination rule-set DESTINATION_NAT]
cjones@R1# set from zone INET
[edit security nat destination rule-set DESTINATION_NAT]
cjones@R1# set rule DST_NAT_WEBSERVER match source-address 0.0.0.0/0
[edit security nat destination rule-set DESTINATION_NAT]
cjones@R1# set rule DST_NAT_WEBSERVER match destination-address 200.200.200.1/32
[edit security nat destination rule-set DESTINATION_NAT]
cjones@R1# set rule DST_NAT_WEBSERVER match destination-port 80
[edit security nat destination rule-set DESTINATION_NAT]
cjones@R1# set rule DST_NAT_WEBSERVER then destination-nat pool WEBSERVER
[edit security nat destination rule-set DESTINATION_NAT]
cjones@R1# top
[edit]
cjones@R1# commit and-quit
commit complete
Exiting configuration mode
```

以上的 `set` 命令生成了如下配置：

```
security {
  nat {
    destination {
      pool WEBSERVER {
        address 10.42.0.100/32 port 80;
      }
      rule-set DESTINATION_NAT {
        from zone INET;
        rule DST_NAT_WEBSERVER {
          match {
            source-address 0.0.0.0/0;
            destination-address 200.200.200.1/32;
            destination-port 80;
          }
          then {
            destination-nat pool WEBSERVER;
          }
        }
      }
    }
  }
}
```

验证到 Web 服务器的目的地址翻译

1. 显示目的地址翻译汇总：

```
cjones@R1> show security nat destination summary
Total pools: 1
Pool name      Address          Routing          Port  Total
              Range          Instance        Address
```



```

WEBSERVER      10.42.0.100 - 10.42.0.100      default      80      1
Total rules: 1
Rule name      Rule set      From      Action
DST_NAT_WEBSERVER  DESTINATION_NAT  INET      WEBSERVER

```

2. 显示安全会话中去往 10.42.0.100 的 NAT 流:

```

cjones@R1> show security flow session nat destination-prefix 200.200.200.1
Session ID: 10057, Policy name: PERMIT_HTTP_TO_WEBSERVER/4, Timeout: 1772, Valid
  In: 207.17.137.239/22072 --> 200.200.200.1/80;tcp, If: fe-0/0/0.0, Pkts: 904, Bytes: 265591
  Out: 10.42.0.100/80 --> 207.17.137.239/22072;tcp, If: fe-0/0/0.1, Pkts: 879, Bytes: 38638

  Deploying SRX Series Services Gateways
  Configuring SRX Series with J-Web

```

小结

Junos 的 NAT 配置显然更直截了当，而 IOS 在翻译前后为 IP 地址使用了一些费解的名字。不过由于有简单易懂的 NAT 翻译表，IOS 的 NAT 验证更容易完成。

第三章

案例分析

IOS 配置	61
Junos 配置	68
总结	82

在本章中你将学习到书中多个配置示例是如何组合在一起的，我们会利用前面章节的配置来构建配置一个缩小版的典型企业网络。

图 3.1 所示意的拓扑中有三个路由器，配置为一个单域的 OSPF 网络，并通过 EBGP 连接到两个 ISP，每个 ISP 都发送缺省路由。

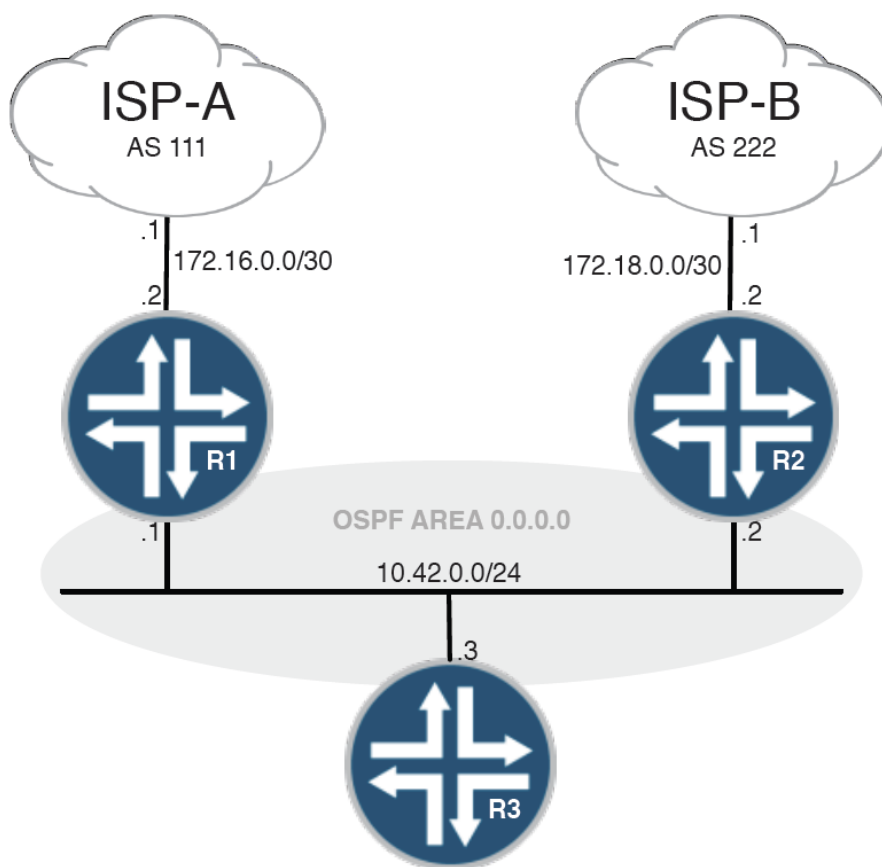


图 3.1 案例分析拓扑

我们在这一章的任务包括：

- 在 R1、R2 和 R3 之间配置 OSPF；
- Advertise a default route into OSPF on R1 and R2
- 在 R1 和 ISP-A 之间配置 EBGP；
- 在 R2 和 ISP-B 之间配置 EBGP；
- 在 R1 和 R2 之间使用 loopback 端口配置 IBGP，并包括 next-hop-self 策略；
- 配置入向和出向流量优先通过 ISP-B；
- 将聚合前缀 10.42.0.0/16 公告到 ISP-A 和 ISP-B。

IOS 配置

我们先使用 IOS 来进行配置。

配置基本连接

1. 在 R1 上设置 IP 地址:

```
R1# configure terminal
R1(config)# interface loopback 0
R1(config-if)# ip address 1.1.1.1 255.255.255.255
R1(config-if)# no shutdown
R1(config-if)# interface FastEthernet 0/0
R1(config-if)# ip address 172.16.0.2 255.255.255.252
R1(config-if)# no shutdown
R1(config-if)# interface FastEthernet 0/1
R1(config-if)# ip address 10.42.0.1 255.255.255.0
R1(config-if)# no shutdown
R1(config-if)# end
R1#
```

2. 在 R2 上设置 IP 地址:

```
R2# configure terminal
R2(config)# interface loopback 0
R2(config-if)# ip address 2.2.2.2 255.255.255.255
R2(config-if)# no shutdown
R2(config-if)# interface FastEthernet 0/0
R2(config-if)# ip address 172.18.0.2 255.255.255.252
R2(config-if)# no shutdown
R2(config-if)# interface FastEthernet 0/1
R2(config-if)# ip address 10.42.0.2 255.255.255.0
R2(config-if)# no shutdown
R2(config-if)# end
R2#
```

3. 在 R3 上设置 IP 地址:

```
R3# configure terminal
R3(config)# interface loopback 0
R3(config-if)# ip address 3.3.3.3 255.255.255.255
R3(config-if)# no shutdown
R3(config-if)# interface FastEthernet 0/0
R3(config-if)# ip address 10.42.0.3 255.255.255.0
R3(config-if)# no shutdown
R3(config-if)# end
R3#
```

验证基本连接

1. 从 R1 ping R2:

```
R1#ping 10.42.0.2
Type escape sequence to abort.
Sending 5, 100-byte ICMP Echos to 10.42.0.2, timeout is 2 seconds:
!!!!
Success rate is 100 percent (5/5), round-trip min/avg/max = 1/2/4 ms
R1#
```

2. 从 R1 ping R3:

```
R1#ping 10.42.0.3.
Type escape sequence to abort.
Sending 5, 100-byte ICMP Echos to 10.42.0.3, timeout is 2 seconds:
!!!!
Success rate is 100 percent (5/5), round-trip min/avg/max = 1/2/4 ms
```

R1#

3. 从 R2 ping R3:

```
R2#ping 10.42.0.3.
Type escape sequence to abort.
Sending 5, 100-byte ICMP Echos to 10.42.0.3, timeout is 2.seconds:
!!!!
Success rate is 100 percent (5/5), round-trip min/avg/max = 1/2/4 ms
R2#
```

4. 从 R1 ping ISP-A:

```
R1#ping 172.16.0.1.
Type escape sequence to abort.
Sending 5, 100-byte ICMP Echos to 172.16.0.1, timeout is 2.seconds:
!!!!
Success rate is 100 percent (5/5), round-trip min/avg/max = 1/2/4 ms
R1#
```

5. 从 R2 ping ISP-B:

```
R2#ping 172.18.0.1.
Type escape sequence to abort.
Sending 5, 100-byte ICMP Echos to 172.18.0.1, timeout is 2.seconds:
!!!!
Success rate is 100 percent (5/5), round-trip min/avg/max = 1/2/4 ms
R2#
```

配置 OSPF

1. 在 R1 上配置 OSPF 进程, 将端口放置到 area 0, 配置 loopback 端口为 passive, 另外手工设定 OSPF router-id:

```
R1#.configure.terminal
R1(config)#.router.ospf.1
R1(config-router)#.router-id.1.1.1.1
R1(config-router)#.network.1.1.1.1.0.0.0.0.area.0
R1(config-router)#.network.10.42.0.1.0.0.0.0.area.0
R1(config-router)#.passive-interface.lo0
R1(config-router)#.end
R1#
```

2. 在 R2 上配置 OSPF 进程, 将端口放置到 area 0, 配置 loopback 端口为 passive, 另外手工设定 OSPF router-id:

```
R2#.configure.terminal
R2(config)#.router.ospf.1
R2(config-router)#.router-id.2.2.2.2
R2(config-router)#.network.2.2.2.2.0.0.0.0.area.0
R2(config-router)#.network.10.42.0.2.0.0.0.0.area.0
R2(config-router)#.passive-interface.lo0
R2(config-router)#.end
R2#
```

3. 在 R3 上配置 OSPF 进程, 将端口放置到 area 0, 配置 loopback 端口为 passive, 另外手工设定 OSPF router-id:

```
R3#.configure.terminal
R3(config)#.router.ospf.1
R3(config-router)#.router-id.3.3.3.3
R3(config-router)#.network.3.3.3.3.0.0.0.0.area.0
R3(config-router)#.network.10.42.0.3.0.0.0.0.area.0
R3(config-router)#.passive-interface.lo0
R3(config-router)#.end
R3#
```

验证 OSPF 配置

1. 在 R1 上验证 OSPF 邻接关系:

```
R1#show ip ospf neigh
Neighbor ID      Pri   State             Dead Time   Address      Interface
2.2.2.2          1     FULL/BDR          00:00:32    10.42.0.2    FastEthernet0/1
3.3.3.3          1     FULL/DR           00:00:34    10.42.0.3    FastEthernet0/1
R1#
```

2. 在 R2 上验证 OSPF 邻接关系:

```
R2#show ip ospf neigh
Neighbor ID      Pri   State             Dead Time   Address      Interface
1.1.1.1          1     FULL/DROTHER      00:00:30    10.42.0.1    FastEthernet0/1
3.3.3.3          1     FULL/BDR          00:00:33    10.42.0.3    FastEthernet0/1
R2#
```

3. 在 R3 上验证 OSPF 邻接关系:

```
R3#show ip ospf neigh
Neighbor ID      Pri   State             Dead Time   Address      Interface
1.1.1.1          1     FULL/DROTHER      00:00:37    10.42.0.1    FastEthernet0/0
2.2.2.2          1     FULL/BDR          00:00:35    10.42.0.2    FastEthernet0/0
R3#
```

在 R1 和 R2 上配置将缺省路由注入 OSPF

1. 配置 R1 将缺省路由注入 OSPF 进程:

```
R1#configure terminal
R1(config)#router ospf 1
R1(config-router)#default-information originate
R1(config-router)#end
R1#
```

2. 配置 R2 将缺省路由注入 OSPF 进程:

```
R2#configure terminal
R2(config)#router ospf 1
R2(config-router)#default-information originate
R2(config-router)#end
R2#
```

验证 OSPF 缺省路由

1. 在 R3 上检查 OSPF 数据库中的 type-5 LSA:

```
R3#show ip ospf database external
      OSPF Router with ID (3.3.3.3) (Process ID 1)
        Type-5 AS External Link States
Routing Bit Set on this LSA
LS age: 164
Options: (No TOS-capability, DC)
LS Type: AS External Link
Link State ID: 0.0.0.0 (External Network Number )
Advertising Router: 1.1.1.1
LS Seq Number: 80000001
Checksum: 0x1D91
Length: 36
Network Mask: /0
  Metric Type: 2 (Larger than any link state path)
  TOS: 0
  Metric: 1
  Forward Address: 0.0.0.0
  External Route Tag: 1
Routing Bit Set on this LSA
```

```

LS age: 82
Options: (No TOS-capability, DC)
LS Type: AS External Link
Link State ID: 0.0.0.0 (External Network Number )
Advertising Router: 2.2.2.2
LS Seq Number: 80000001
Checksum: 0xFEAB
Length: 36
Network Mask: /0
    Metric Type: 2 (Larger than any link state path)
    TOS: 0
    Metric: 1
    Forward Address: 0.0.0.0
    External Route Tag: 1

```

2. 检查 RIB 中的 OSPF 外部路由:

```

R3#show ip route ospf
    1.0.0.0/32 is subnetted, 1 subnets
O       1.1.1.1 [110/11] via 10.42.0.1, 00:59:55, FastEthernet0/0
    2.0.0.0/32 is subnetted, 1 subnets
O       2.2.2.2 [110/11] via 10.42.0.2, 00:59:55, FastEthernet0/0
O*E2 0.0.0.0/0 [110/1] via 10.42.0.2, 00:03:00, FastEthernet0/0
       [110/1] via 10.42.0.1, 00:04:22, FastEthernet0/0

```

配置 R1 和 R2 上到 ISP-A 和 ISP-B 的 EBGp

1. 配置 R1(AS 65001)与 ISP-A (AS 111)建立对等:

```

R1# configure terminal
R1(config)# router bgp 65001
R1(config-router)# no synchronization
R1(config-router)# neighbor 172.16.0.1 remote-as 111
R1(config-router)# no auto-summary
R1(config-router)# end
R1#

```

2. 配置 R2(AS 65001)与 ISP-B (AS 222)建立对等:

```

R2# configure terminal
R2(config)# router bgp 65001
R2(config-router)# no synchronization
R2(config-router)# neighbor 172.18.0.1 remote-as 222
R2(config-router)# no auto-summary
R2(config-router)# end
R2#

```

验证 EBGp 配置

1. 验证 R1 与 ISP-A 的 EBGp 对等关系:

```

R1#show ip bgp neighbors 172.16.0.1 | include =
    BGP state = Established, up for 00:08:37
R1#show ip bgp summary | include 172.16.0.1|Neighbor
Neighbor      V    AS MsgRcvd MsgSent  TblVer  InQ  OutQ Up/Down  State/PfxRcd
172.16.0.1    4   111     11     10       2    0    0 00:07:50      1
R1#

```

2. 检查 R1 上的 Adj-RIB-In 表(又称为 BGP 表), 验证 R1 收到了由 ISP-A 公告的缺省路由:

```

R1#show ip bgp
BGP table version is 2, local router ID is 1.1.1.1
Status codes: s suppressed, d damped, h history, * valid, > best, i - internal,
               r RIB-failure, S Stale
Origin codes: i - IGP, e - EGP, ? - incomplete

```


Network	Next Hop	Metric	LocPrf	Weight	Path
*> 0.0.0.0	172.16.0.1	0		0	111 i

3. 在 R1 上验证由 BGP 学习到的缺省路由被安装到了 RIB(又称为路由表):

```
R1#show ip route bgp
B* 0.0.0.0/0 [20/0] via 172.16.0.1, 00:23:52
```

4. 验证 R2 与 ISP-B 的 EBGP 对等关系:

```
R2#show ip bgp neighbors 172.18.0.1 | include =
BGP state = Established, up for 00:02:46
R2#show ip bgp summary | include 172.18.0.1|Neighbor
Neighbor      V    AS MsgRcvd MsgSent   TblVer   InQ OutQ Up/Down State/PfxRcd
172.18.0.1    4    222      7      6        2    0    0 00:03:37      1
```

5. 检查 R2 上的 Adj-RIB-In 表, 验证 R2 收到了由 ISP-B 公告的缺省路由:

```
R2#show ip bgp
BGP table version is 2, local router ID is 2.2.2.2
Status codes: s suppressed, d damped, h history, * valid, > best, i - internal,
               r RIB-failure, S Stale
Origin codes: i - IGP, e - EGP, ? - incomplete
Network        Next Hop        Metric LocPrf Weight Path
*> 0.0.0.0      172.18.0.1      0                0 222 i
```

6. 在 R2 上验证由 BGP 学习到的缺省路由被安装到了 RIB:

```
R2#show ip route bgp
B* 0.0.0.0/0 [20/0] via 172.18.0.1, 00:21:23
```

在 R1 和 R2 上配置 IBGP

1. 在 R1 上配置到 R2 的 IBGP, 使用 loopback 端口建立对等关系:

```
R1# configure terminal
R1(config)# router bgp 65001
R1(config-router)# neighbor 2.2.2.2 remote-as 65001
R1(config-router)# neighbor 2.2.2.2 update-source Loopback0
R1(config-router)# end
R1#
```

2. 在 R1 上配置静态路由以确保到 R2 loopback 地址的可达性:

由于 OSPF 已经在运行, 所以这一步不是必须的, 在此只是起演示作用。为了让 OSPF 路由被优先使用, 我们将静态路由的 preference 值(在 IOS 中称为管理距离 administrative distance)调高

```
R1# configure terminal
R1(config)#ip route 2.2.2.2 255.255.255.255 10.42.0.2 254
R1(config)#end
R1#
```

3. 在 R1 上配置 IBGP 对等体, 更新所有送往 R2 前缀的下一跳:

```
R1#configure terminal
R1(config)#router bgp 65001
R1(config-router)#neighbor 2.2.2.2 next-hop-self
R1(config-router)#end
R1#
```

4. 在 R2 上配置到 R1 的 IBGP, 使用 loopback 端口建立对等关系:

```
R2#configure terminal
R2(config)#router bgp 65001
R2(config-router)#neighbor 1.1.1.1 remote-as 65001
R2(config-router)#neighbor 1.1.1.1 update-source Loopback0
R2(config)#end
```

R2#

5. 在 R2 上配置静态路由以确保在 OSPF 失效时 R1 loopback 地址仍然可达:

```
R2#configure terminal
R2(config)#ip route 1.1.1.1 255.255.255.255 10.42.0.1 254
R2(config)#end
R2#
```

6. 在 R2 上配置 IBGP 对等体, 更新所有送往 R1 前缀的下一跳:

```
R2#configure terminal
R2(config)#router bgp 65001
R2(config-router)#neighbor 1.1.1.1 next-hop-self
R2(config-router)#end
R2#
```

验证 R1 与 R2 之间的 IBGP

1. 验证 R1 与 R2 之间的 IBGP 邻接关系:

```
R1#show ip bgp neighbors 2.2.2.2 | include =
    BGP state = Established, up for 00:09:14
```

2. 验证 R1 收到了来自 R2 的 BGP 前缀:

```
R1#show ip bgp summary | include 2.2.2.2|Neighbor
Neighbor      V    AS MsgRcvd MsgSent   TblVer  InQ OutQ Up/Down  State/PfxRcd
2.2.2.2        4 65001      12      11         2    0    0 00:06:30      1
```

将聚合前缀 10.42.0.0/16 公告到 AS 111 和 AS 222

1. 在 R1 上将 aggregate-address 命令添加到 BGP 进程下:

```
R1#configure terminal
R1(config)#router bgp 65001
R1(config-router)# aggregate-address 10.42.0.0 255.255.0.0 summary-only
R1(config-router)# network 10.42.0.0 mask 255.255.255.0
R1(config-router)# end
R1#
```

2. 在 R2 上将 aggregate-address 命令添加到 BGP 进程下:

```
R2#configure terminal
R2(config)#router bgp 65001
R2(config-router)# aggregate-address 10.42.0.0 255.255.0.0 summary-only
R2(config-router)# network 10.42.0.0 mask 255.255.255.0
R2(config-router)# end
R2#
```

验证聚合已经被送往 AS 111 和 AS 222

1. 检查路由列表已经从 R1 公告到 ISP-A:

```
R1#show ip bgp neighbors 172.16.0.1 advertised-routes
BGP table version is 5, local router ID is 1.1.1.1
Status codes: s suppressed, d damped, h history, * valid, > best, i - internal,
               r RIB-failure, S Stale
Origin codes: i - IGP, e - EGP, ? - incomplete
   Network        Next Hop           Metric LocPrf Weight Path
*> 10.42.0.0/16    0.0.0.0                   32768 i
Total number of prefixes 1
```

2. 检查路由列表已经从 R2 公告到 ISP-B:

```
R2#show ip bgp neighbors 172.18.0.1 advertised-routes
BGP table version is 6, local router ID is 2.2.2.2
Status codes: s suppressed, d damped, h history, * valid, > best, i - internal,
```

```

        r RIB-failure, S Stale
Origin codes: i - IGP, e - EGP, ? - incomplete
   Network        Next Hop           Metric LocPrf Weight Path
*> 10.42.0.0/16    0.0.0.0                          32768 i
Total number of prefixes 1

```

使入向流量优先通过 ISP-B 进入网络

1. 在 R1 上创建 route-map，将本地 AS 号预置在 AS-path 中三次：

```

R1#configure terminal
R1(config)#route-map PREFER_ISP2_INBOUND permit 10
R1(config-route-map)# set as-path prepend 65001 65001 65001
R1(config-route-map)#exit

```

2. 在 R1 上将 route-map 应用到 BGP 配置：

```

R1(config)#router bgp 65001
R1(config-router)# neighbor 172.16.0.1 route-map PREFER_ISP2_INBOUND out
R1(config-router)#end
R1#

```

验证 AS 路径预置配置已经生效

注意 不幸的是在 IOS 中没有很好的办法来验证 AS-path 已经被正确地修改了，你必须在 ISP 路由器上检查 BGP 表。

```

ISPA#show ip bgp
BGP table version is 8, local router ID is 172.16.0.1
Status codes: s suppressed, d damped, h history, * valid, > best, i - internal,
               r RIB-failure, S Stale
Origin codes: i - IGP, e - EGP, ? - incomplete
   Network        Next Hop           Metric LocPrf Weight Path
*> 10.42.0.0/16    172.16.0.2              0             0 65001 65001 65001 65001 i

```

使出向流量优先通过 ISP-B 离开网络

在 R2 上配置对所有从 ISP-B 学到的路由增加 local preference：

```

R2#configure terminal
R2(config)#route-map PREFER_ISPB_OUTBOUND permit 10
R2(config-route-map)#set local-preference 110
R2(config-route-map)#router bgp 65001
R2(config-router)#neighbor 172.18.0.1 route-map PREFER_ISPB_OUTBOUND in
R2(config-router)#end
R2#

```

验证 local preference 已经被修改

```

R2#show ip bgp | include Network|172.18.0.1
   Network        Next Hop           Metric LocPrf Weight Path
*  0.0.0.0         172.18.0.1              0      110      0 222 i

```

Junos 配置

现在让我们在 Junos 中完成同样的配置，IOS 工程师可以留意 Junos 是怎样依靠策略来操控 IGP 和 BGP 路由的。

当你阅读完这一节之后，研究一下你的最终配置，你会发现它组织得多么有逻辑，而如何更进一步配置都是可以预见的，因为 Junos 对于配置任何对象都遵循了非常标准的格式。

配置初始连接

1. 在 R1 上设置端口 IP 地址:

```
cjones@R1> configure
[edit]
cjones@R1# set interfaces ge-0/0/0 unit 0 family inet address 172.16.0.2/30
[edit]
cjones@R1# set interfaces ge-0/0/1 unit 0 family inet address 10.42.0.1/24
[edit]
cjones@R1# set interfaces lo0 unit 0 family inet address 1.1.1.1/32
[edit]
cjones@R1# commit and-quit
commit complete
Exiting configuration mode
cjones@R1>
```

2. 检查 R1 上的 IP 地址:

```
cjones@R1> show configuration interfaces
ge-0/0/0 {
    unit 0 {
        family inet {
            address 172.6.0.2/30;
        }
    }
}
ge-0/0/1 {
    unit 0 {
        family inet {
            address 10.42.0.1/24;
        }
    }
}
lo0 {
    unit 0 {
        family inet {
            address 1.1.1.1/24;
        }
    }
}
```

3. 在 R2 上设置端口 IP 地址:

```
cjones@R2> configure
[edit]
cjones@R2# set interfaces ge-0/0/0 unit 0 family inet address 172.18.0.2/30
[edit]
cjones@R2# set interfaces ge-0/0/1 unit 0 family inet address 10.42.0.2/24
[edit]
cjones@R2# set interfaces lo0 unit 0 family inet address 2.2.2.2/32
[edit]
cjones@R2# commit and-quit
commit complete
Exiting configuration mode
```

```
cjones@R2>
```

4. 检查 R2 上的 IP 地址:

```
cjones@R2> show configuration interfaces
```

```
ge-0/0/0 {
  unit 0 {
    family inet {
      address 172.18.0.2/30;
    }
  }
}
ge-0/0/1 {
  unit 0 {
    family inet {
      address 10.42.0.2/24;
    }
  }
}
lo0 {
  unit 0 {
    family inet {
      address 2.2.2.2/24;
    }
  }
}
```

5. 在 R3 上设置端口 IP 地址:

```
cjones@R3> configure
```

```
[edit]
```

```
cjones@R3# set interfaces ge-0/0/1 unit 0 family inet address 10.42.0.3/24
```

```
[edit]
```

```
cjones@R3# set interfaces lo0 unit 0 family inet address 3.3.3.3/32
```

```
[edit]
```

```
cjones@R2# commit and-quit
```

```
commit complete
```

```
Exiting configuration mode
```

```
cjones@R3>
```

6. 检查 R3 上的 IP 地址:

```
cjones@R3> show configuration interfaces
```

```
ge-0/0/1 {
  unit 0 {
    family inet {
      address 10.42.0.3/24;
    }
  }
}
lo0 {
  unit 0 {
    family inet {
      address 3.3.3.3/32;
    }
  }
}
```

验证 Junos 路由器的初始连接

1. 从 R1 ping R2:

```
cjones@R1> ping 10.42.0.2 rapid
```

```
PING 10.42.0.2 (10.42.0.2): 56 data bytes
```

```
!!!!
```

```
--- 10.42.0.2 ping statistics ---
```

5 packets transmitted, 5 packets received, 0% packet loss
round-trip min/avg/max/stddev = 2.238/2.791/4.261/0.752 ms

2. 从 R1 ping R3:

```
cjones@R1> ping 10.42.0.3 rapid
PING 10.42.0.3 (10.42.0.3): 56 data bytes
!!!!
--- 10.42.0.3 ping statistics ---
5 packets transmitted, 5 packets received, 0% packet loss
round-trip min/avg/max/stddev = 2.332/5.504/17.629/6.063 ms
```

3. 从 R2 ping R3:

```
cjones@R2> ping 10.42.0.3 rapid
PING 10.42.0.3 (10.42.0.3): 56 data bytes
!!!!
--- 10.42.0.3 ping statistics ---
5 packets transmitted, 5 packets received, 0% packet loss
round-trip min/avg/max/stddev = 2.042/7.932/30.717/11.394 ms
```

4. 从 R1 ping ISP-A:

```
cjones@R1> ping 172.16.0.1 rapid
PING 172.16.0.1 (172.16.0.1): 56 data bytes
!!!!
--- 172.16.0.1 ping statistics ---
5 packets transmitted, 5 packets received, 0% packet loss
round-trip min/avg/max/stddev = 2.625/7.827/31.212/10.268 ms
```

5. 从 R2 ping ISP-B:

```
cjones@R2> ping 172.18.0.1 rapid
PING 172.18.0.1 (172.18.0.1): 56 data bytes
!!!!
--- 172.18.0.1 ping statistics ---
5 packets transmitted, 5 packets received, 0% packet loss
round-trip min/avg/max/stddev = 2.766/6.983/29.472/11.121 ms
```

配置 OSPF

1. 在 R1 上配置 OSPF 进程, 将端口放置到 area 0, 配置 loopback 端口为 passive, 另外手工设定 OSPF router-id:

```
cjones@R1> configure
[edit]
cjones@R1# set protocols ospf area 0 interface ge-0/0/1.0
[edit]
cjones@R1# set protocols ospf area 0 interface lo0.0 passive
[edit]
cjones@R1# set routing-options router-id 1.1.1.1
[edit]
cjones@R1# commit and-quit
commit complete
Exiting configuration mode
cjones@R1>
```

2. 检查 R1 上的 OSPF 配置:

```
cjones@R1> show configuration routing-options
router-id 1.1.1.1;
cjones@R1> show configuration protocols
ospf {
    area 0.0.0.0 {
        interface ge-0/0/1.0;
        interface lo0.0 {
            passive;
```

```

    }
}

```

3. 在 R2 上配置 OSPF 进程, 将端口放置到 area 0, 配置 loopback 端口为 passive, 另外手工设定 OSPF router-id:

```

cjones@R2> configure
[edit]
cjones@R2# set protocols ospf area 0 interface ge-0/0/1.0
[edit]
cjones@R2# set protocols ospf area 0 interface lo0.0 passive
[edit]
cjones@R2# set routing-options router-id 2.2.2.2
[edit]
cjones@R2# commit and-quit
commit complete
Exiting configuration mode
cjones@R2>

```

4. 检查 R2 上的 OSPF 配置:

```

cjones@R2> show configuration routing-options
router-id 2.2.2.2;
cjones@R2> show configuration protocols
ospf {
    area 0.0.0.0 {
        interface ge-0/0/1.0;
        interface lo0.0 {
            passive;
        }
    }
}

```

5. 在 R3 上配置 OSPF 进程, 将端口放置到 area 0, 配置 loopback 端口为 passive, 另外手工设定 OSPF router-id:

```

cjones@R3> configure
[edit]
cjones@R3# set protocols ospf area 0 interface ge-0/0/1.0
[edit]
cjones@R3# set protocols ospf area 0 interface lo0.0 passive
[edit]
cjones@R3# set routing-options router-id 3.3.3.3
[edit]
cjones@R3# commit and-quit
commit complete
Exiting configuration mode
cjones@R3>

```

6. 检查 R3 上的 OSPF 配置:

```

cjones@R3> show configuration routing-options
router-id 3.3.3.3;
cjones@R3> show configuration protocols
ospf {
    area 0.0.0.0 {
        interface ge-0/0/1.0;
        interface lo0.0 {
            passive;
        }
    }
}

```

验证 OSPF 配置

1. 在 R1 上验证 OSPF 邻接关系:

```
cjones@R1> show ospf neighbor
Address      Interface      State      ID            Pri    Dead
10.42.0.3    ge-0/0/1.0     Full       3.3.3.3       128    30
10.42.0.2    ge-0/0/1.0     Full       2.2.2.2       128    37
```

2. 在 R2 上验证 OSPF 邻接关系:

```
cjones@R2> show ospf neighbor
Address      Interface      State      ID            Pri    Dead
10.42.0.1    ge-0/0/1.0     Full       1.1.1.1       128    32
10.42.0.3    ge-0/0/1.0     Full       3.3.3.3       128    30
```

3. 在 R3 上验证 OSPF 邻接关系:

```
cjones@R3> show ospf neighbor
Address      Interface      State      ID            Pri    Dead
10.42.0.1    ge-0/0/1.0     Full       1.1.1.1       128    32
10.42.0.2    ge-0/0/1.0     Full       2.2.2.2       128    37
```

在 R1 和 R2 上配置将缺省路由注入 OSPF

1. 在 R1 上配置静态缺省路由:

```
cjones@R1> configure
Entering configuration mode
[edit]
cjones@R1# set routing-options static route 0/0 discard
```

2. 在 R1 上配置输出策略:

```
[edit]
cjones@R1# edit policy-options policy-statement DEFAULT_TO_OSPF
[edit policy-options policy-statement DEFAULT_TO_OSPF]
cjones@R1# set from protocol static
[edit policy-options policy-statement DEFAULT_TO_OSPF]
cjones@R1# set from route-filter 0/0 exact
[edit policy-options policy-statement DEFAULT_TO_OSPF]
cjones@R1# set then accept
[edit policy-options policy-statement DEFAULT_TO_OSPF]
cjones@R1# top
[edit]
cjones@R1# show policy-options
policy-statement DEFAULT_TO_OSPF {
    from {
        protocol static;
        route-filter 0.0.0.0/0 exact;
    }
    then accept;
}
```

3. 在 R1 上将输出策略应用到 OSPF, 达到将缺省路由注入 OSPF 的目的:

```
[edit]
cjones@R1# set protocols ospf export DEFAULT_TO_OSPF
[edit]
cjones@R1# show protocols ospf
export DEFAULT_TO_OSPF;
area 0.0.0.0 {
    interface ge-0/0/1.0;
    interface lo0.0 {
        passive;
    }
}
```

4. 检查 R1 上的配置并提交使其生效:


```

[edit]
cjones@R1# show | compare
[edit routing-options]
+   static {
+       route 0.0.0.0/0 discard;
+   }
[edit protocols ospf]
+   export DEFAULT_TO_OSPF;
[edit]
+   policy-options {
+       policy-statement DEFAULT_TO_OSPF {
+           from {
+               protocol static;
+               route-filter 0.0.0.0/0 exact;
+           }
+           then accept;
+       }
+   }
[edit]
cjones@R1# commit and-quit
commit complete
Exiting configuration mode

```

5. 下面我们将同样的配置添加到 R2:

这次我们不重新键入所有配置，而是来试试新的小技巧。我们拷贝上面 **show|compare** 命令的输出，然后在 R2 上使用 **load patch terminal** 命令，粘贴，再按 **ctrl+d** 完成输入，最后提交生效。如下所示:

```

cjones@R2> configure
Entering configuration mode
[edit]
cjones@R2# load patch terminal
[Type ^D at a new line to end input]

[edit routing-options]
+   static {
+       route 0.0.0.0/0 discard;
+   }
[edit protocols ospf]
+   export DEFAULT_TO_OSPF;
[edit]
+   policy-options {
+       policy-statement DEFAULT_TO_OSPF {
+           from {
+               protocol static;
+               route-filter 0.0.0.0/0 exact;
+           }
+           then accept;
+       }
+   }

load complete
[edit]
cjones@R2# commit and-quit
commit complete
Exiting configuration mode

```

验证 OSPF 缺省路由

1. 在 R3 上检查 OSPF 数据库中的 type-5 LSA:

```

cjones@R3> show ospf database external
      OSPF AS SCOPE link state database
Type      ID              Adv Rtr          Seq      Age  Opt  Cksum  Len

```

Extern	0.0.0.0	1.1.1.1	0x80000001	599	0x22	0xe2cb	36
Extern	0.0.0.0	2.2.2.2	0x80000001	220	0x22	0xc4e5	36

2. 在 R3 上检查 RIB 中的 OSPF 外部路由:

注意 在这里使用了 **exact** 关键词, 因为如果不这么做的话, 由于所有路由都匹配 0.0.0.0/0, 整个 RIB 都会被显示出来。

```
cjones@R3> show route 0.0.0.0/0 exact
inet 0: 7 destinations, 7 routes (7 active, 0 holddown, 0 hidden)
+ = Active Route, - = Last Active, * = Both
0.0.0.0/0          *[OSPF/150] 00:04:35, metric 0, tag 0
                   to 10.42.0.1 via ge-0/0/1.0
                   > to 10.42.0.2 via ge-0/0/1.0
```

配置 R1 和 R2 上到 ISP-A 和 ISP-B 的 EBGP

1. 配置 R1(AS 65001)与 ISP-A (AS 111)建立对等:

```
cjones@R1> configure
Entering configuration mode
[edit]
cjones@R1# set routing-options autonomous-system 65001
[edit]
cjones@R1# set protocols bgp group ISP-A type external neighbor 172.16.0.1 peer-as 111
```

2. 检查 R1 上的 BGP 配置, 提交生效:

```
[edit]
cjones@R1# show routing-options autonomous-system
65001;
[edit]
cjones@R1# show protocols bgp
group ISP-A {
    type external;
    neighbor 172.16.0.1 {
        peer-as 111;
    }
}
[edit]
cjones@R1# commit and-quit
commit complete
Exiting configuration mode
```

3. 配置 R2(AS 65001)与 ISP-B (AS 222)建立对等:

```
cjones@R2> configure
Entering configuration mode
[edit]
cjones@R2# set routing-options autonomous-system 65001
[edit]
cjones@R2# set protocols bgp group ISP-B type external neighbor 172.18.0.1 peer-as 222
```

4. 检查 R2 上的 BGP 配置, 提交生效:

```
[edit]
cjones@R2# show routing-options autonomous-system
65001;
[edit]
cjones@R2# show protocols bgp
group ISP-B {
    type external;
    neighbor 172.18.0.1 {
        peer-as 222;
    }
}
```

```
[edit]
cjones@R2# commit and-quit
commit complete
Exiting configuration mode
```

验证 EBGp 配置

1. 验证 R1 与 ISP-A 的 EBGp 对等关系:

```
cjones@R1> show bgp neighbor 172.16.0.1
Peer: 172.16.0.1+57730 AS 111 Local: 172.16.0.2+179 AS 65001
Type: External State: Established Flags: <ImportEval Sync>
Last State: OpenConfirm Last Event: RecvKeepAlive
Last Error: None
Options: <Preference PeerAS Refresh>
Holdtime: 90 Preference: 170
Number of flaps: 0
Peer ID: 172.16.0.1 Local ID: 1.1.1.1 Active Holdtime: 90
Keepalive Interval: 30 Peer index: 0
BFD: disabled, down
Local Interface: ge-0/0/0.0
```

2. 检查 R1 上的 Adj-RIB-In 表, 验证 R1 收到了由 ISP-A 公告的 111.111.111.0/24 路由:

```
cjones@R1> show route receive-protocol bgp 172.16.0.1
inet 0: 11 destinations, 13 routes (11 active, 0 holddown, 0 hidden)
Prefix Nexthop MED Lclpref AS path
* 111.111.111.0/24 172.16.0.1 111 I
```

3. 在 R1 上验证由 ISP-A 学习到的 BGP 路由已经被安装到 RIB 中:

```
cjones@R1> show route protocol bgp
inet 0: 11 destinations, 13 routes (11 active, 0 holddown, 0 hidden)
+ = Active Route, - = Last Active, * = Both
111.111.111.0/24 *[BGP/170] 00:01:30, localpref 100
AS path: 111 I
> to 172.16.0.1 via ge-0/0/0.0
```

4. 验证 R2 与 ISP-B 的 EBGp 对等关系:

```
cjones@R2> show bgp neighbor 172.18.0.1
Peer: 172.18.0.1+179 AS 222 Local: 172.18.0.2+56620 AS 65001
Type: External State: Established Flags: <ImportEval Sync>
Last State: OpenConfirm Last Event: RecvKeepAlive
Last Error: None
Options: <Preference PeerAS Refresh>
Holdtime: 90 Preference: 170
Number of flaps: 0
Peer ID: 172.18.0.1 Local ID: 2.2.2.2 Active Holdtime: 90
Keepalive Interval: 30 Peer index: 0
BFD: disabled, down
Local Interface: ge-0/0/0.0
```

5. 检查 R2 上的 Adj-RIB-In 表, 验证 R2 收到了由 ISP-B 公告的 111.111.111.0/24 路由:

```
cjones@R2> show route receive-protocol bgp 172.18.0.1
inet 0: 11 destinations, 13 routes (11 active, 0 holddown, 0 hidden)
Prefix Nexthop MED Lclpref AS path
* 111.111.111.0/24 172.18.0.1 222 I
```

6. 在 R2 上验证由 ISP-B 学习到的 BGP 路由已经被安装到 RIB 中:

```
cjones@R2> show route protocol bgp
inet 0: 11 destinations, 13 routes (11 active, 0 holddown, 0 hidden)
+ = Active Route, - = Last Active, * = Both
```

```
111.111.111.0/24    *[BGP/170] 00:02:42, localpref 100
                  AS path: 222 I
                  > to 172.18.0.1 via ge-0/0/0.0
```

在 R1 和 R2 上配置 IBGP

1. 在 R1 上配置到 R2 的 IBGP，使用 loopback 端口建立对等关系:

```
cjones@R1> configure
Entering configuration mode
[edit]
cjones@R1# set protocols bgp group IBGP type internal neighbor 2.2.2.2 local-address 1.1.1.1 peer-as 65001
```

2. 在 R1 上配置静态路由以确保到 R2 loopback 地址的可达性:

因为 OSPF 已经在运行，所以这一步不是必须的，在此只是起演示作用。由于我们倾向于让 OSPF 路由被优先使用，所以我们赋予静态路由更高的 preference 值。这样只有在 IGP 失效的时候，这条浮动静态路由才会被使用。

```
[edit]
cjones@R1# set routing-options static route 2.2.2.2/32 next-hop 10.42.0.2 preference 254
```

3. 在 R1 上配置策略，将所有送往 R2 的 IBGP 前缀的下一跳更改为自身的端口地址:

```
[edit]
cjones@R1# set policy-options policy-statement NHS then next-hop self
```

4. 在 R1 上将 NHS 策略应用到对 R2 的对等连接上:

```
[edit]
cjones@R1# set protocols bgp group IBGP export NHS
```

5. 检查在 R1 上进行的更改，提交生效:

```
[edit]
cjones@R1# show | compare
[edit routing-options static]
    route 0.0.0.0/0 { ... }
+   route 2.2.2.2/32 {
+       next-hop 10.42.0.2;
+       preference 254;
+   }
[edit protocols bgp]
    group ISP-A { ... }
+   group IBGP {
+       type internal;
+       export NHS;
+       neighbor 2.2.2.2 {
+           local-address 1.1.1.1;
+           peer-as 65001;
+       }
+   }
[edit policy-options]
+   policy-statement NHS {
+       then {
+           next-hop self;
+       }
+   }
[edit]
cjones@R1# commit and-quit
commit complete
Exiting configuration mode
```

6. 在 R2 上配置到 R1 的 IBGP，使用 loopback 端口建立对等关系:

```
cjones@R2> configure
Entering configuration mode
[edit]
cjones@R2# set protocols bgp group IBGP type internal neighbor 1.1.1.1 local-address 2.2.2.2 peer-as 65001
```

7. 在 R2 上配置静态路由以确保到 R1 loopback 地址的可达性:

```
[edit]
cjones@R2# set routing-options static route 1.1.1.1/32 next-hop 10.42.0.1 preference 254
```

8. 在 R2 上配置策略, 将所有送往 R1 的 IBGP 前缀的下一跳更改为自身的端口地址:

```
[edit]
cjones@R2# set policy-options policy-statement NHS then next-hop self
```

9. 在 R2 上将 NHS 策略应用到对 R1 的对等连接上:

```
[edit]
cjones@R2# set protocols bgp group IBGP export NHS
```

10. 检查在 R2 上进行的更改, 提交生效:

```
[edit]
cjones@R2# show | compare
[edit routing-options static]
    route 0.0.0.0/0 { ... }
+   route 1.1.1.1/32 {
+       next-hop 10.42.0.1;
+       preference 254;
+   }
[edit protocols bgp]
    group ISP-B { ... }
+   group IBGP {
+       type internal;
+       export NHS;
+       neighbor 1.1.1.1 {
+           local-address 2.2.2.2;
+           peer-as 65001;
+       }
+   }
[edit policy-options]
+   policy-statement NHS {
+       then {
+           next-hop self;
+       }
+   }
[edit]
cjones@R2# commit and-quit
commit complete
Exiting configuration mode
```

验证 R1 与 R2 之间的 IBGP

1. 验证 R1 与 R2 之间的 IBGP 邻接关系:

```
cjones@R1> show bgp neighbor 2.2.2.2
Peer: 2.2.2.2+63702 AS 65001 Local: 1.1.1.1+179 AS 65001
Type: Internal State: Established Flags: <Sync>
Last State: OpenConfirm Last Event: RecvKeepAlive
Last Error: None
Export: [ NHS ]
Options: <Preference LocalAddress PeerAS Refresh>
Local Address: 1.1.1.1 Holdtime: 90 Preference: 170
Number of flaps: 0
```

```
Peer ID: 2.2.2.2          Local ID: 1.1.1.1          Active Holdtime: 90
```

2. 验证 R1 从 R2 收到了前缀，其下一跳是正确的：

```
cjones@R1> show route receive-protocol bgp 2.2.2.2
inet 0: 11 destinations, 13 routes (11 active, 0 holddown, 0 hidden)
  Prefix                Nexthop              MED      Lclpref    AS path
* 111.111.111.0/24      2.2.2.2              100       222 I
```

3. 在 R1 上验证来自 R2 的前缀被安装到了 RIB:

```
cjones@R1> show route protocol bgp 111.111.111.0/24
inet 0: 10 destinations, 13 routes (10 active, 0 holddown, 0 hidden)
+ = Active Route, - = Last Active, * = Both
111.111.111.0/24  *[BGP/170] 00:21:51, localpref 100
                  AS path: 111 I
                  > to 172.16.0.1 via ge-0/0/0.0
                  [BGP/170] 00:02:18, localpref 100, from 2.2.2.2
                  AS path: 222 I
                  > to 10.42.0.2 via ge-0/0/1.0
```

4. 验证 R2 从 R1 收到了前缀，其下一跳是正确的：

```
cjones@R2> show route receive-protocol bgp 1.1.1.1
inet 0: 11 destinations, 13 routes (11 active, 0 holddown, 0 hidden)
  Prefix                Nexthop              MED      Lclpref    AS path
* 111.111.111.0/24      1.1.1.1              100       111 I
```

5. 在 R2 上验证来自 R1 的前缀被安装到了 RIB:

```
cjones@R2> show route protocol bgp 111.111.111.0/24
inet 0: 10 destinations, 13 routes (10 active, 0 holddown, 0 hidden)
+ = Active Route, - = Last Active, * = Both
111.111.111.0/24  *[BGP/170] 00:03:28, localpref 100
                  AS path: 222 I
                  > to 172.18.0.1 via ge-0/0/0.0
                  [BGP/170] 00:23:01, localpref 100, from 1.1.1.1
                  AS path: 111 I
                  > to 10.42.0.1 via ge-0/0/1.0
```

将聚合前缀 10.42.0.0/16 公告到 AS 111 和 AS 222

公告聚合路由在 Junos 中是相当简单的，虽然需要比在 IOS 中更多的配置，但是却都是使用统一的策略架构，这个架构你现在应该已经慢慢熟悉了。

你首先必须创建聚合路由然后才能公告出去，其语法与定义静态路由基本上是相同的。一旦聚合路由被创建，你就可以在策略中匹配它，进而将其应用于 BGP 的出口策略。

重要的是要注意在策略的末尾有一条 **reject** 条款，这样才能保证 BGP 不会公告除了汇聚路由之外的任何东西。

1. 在 R1 上创建汇聚路由：

```
cjones@R1> configure
Entering configuration mode
[edit]
cjones@R1# set routing-options aggregate route 10.42.0.0/16
```

2. 在 R1 上配置一条策略，匹配聚合路由并接受它，但是拒绝所有其它路由：

```
[edit]
cjones@R1# edit policy-options policy-statement AGG_TO_ISP
[edit policy-options policy-statement AGG_TO_ISP]
cjones@R1# set term ACCEPT_AGG from protocol aggregate
[edit policy-options policy-statement AGG_TO_ISP]
cjones@R1# set term ACCEPT_AGG from route-filter 10.42.0.0/16 exact
```

```
[edit policy-options policy-statement AGG_TO_ISP]
cjones@R1# set term ACCEPT_AGG then accept
[edit policy-options policy-statement AGG_TO_ISP]
cjones@R1# set term REJECT_OTHERS then reject
[edit policy-options policy-statement AGG_TO_ISP]
cjones@R1# top
```

3. 将这条策略应用为 BGP 配置的组/邻居的输出策略:

```
[edit]
cjones@R1# set protocols bgp group ISP-A neighbor 172.16.0.1 export AGG_TO_ISP
```

4. 检查 R1 的配置并提交生效:

```
[edit]
cjones@R1# show | compare
[edit routing-options]
+ aggregate {
+   route 10.42.0.0/16;
+ }
[edit protocols bgp group ISP-A neighbor 172.16.0.1]
+ export AGG_TO_ISP;
[edit policy-options]
+ policy-statement AGG_TO_ISP {
+   term ACCEPT_AGG {
+     from {
+       protocol aggregate;
+       route-filter 10.42.0.0/16 exact;
+     }
+     then accept;
+   }
+   term REJECT_OTHERS {
+     then reject;
+   }
+ }
[edit]
cjones@R1# commit and-quit
commit complete
Exiting configuration mode
```

5. 在 R2 上创建汇聚路由:

```
cjones@R2> configure
Entering configuration mode
[edit]
cjones@R2# set routing-options aggregate route 10.42.0.0/16
```

6. 在 R2 上配置一条策略, 匹配聚合路由并接受它, 但是拒绝所有其它路由。 要注意, 如果没有配置 reject 语句的话, 表示隐式地接受。

```
[edit]
cjones@R2# edit policy-options policy-statement AGG_TO_ISP
[edit policy-options policy-statement AGG_TO_ISP]
cjones@R2# set term ACCEPT_AGG from protocol aggregate
[edit policy-options policy-statement AGG_TO_ISP]
cjones@R2# set term ACCEPT_AGG from route-filter 10.42.0.0/16 exact
[edit policy-options policy-statement AGG_TO_ISP]
cjones@R2# set term ACCEPT_AGG then accept
[edit policy-options policy-statement AGG_TO_ISP]
cjones@R2# set term REJECT_OTHERS then reject
[edit policy-options policy-statement AGG_TO_ISP]
cjones@R2# top
```

7. 将这条策略应用为 BGP 配置的组/邻居的输出策略:

```
[edit]
```

```
cjones@R2# set protocols bgp group ISP-B neighbor 172.18.0.1 export AGG_TO_ISP
```

8. 检查 R2 的配置并提交生效:

```
[edit]
cjones@R2# show | compare
[edit routing-options]
+ aggregate {
+   route 10.42.0.0/16;
+ }
[edit protocols bgp group ISP-B neighbor 172.18.0.1]
+ export AGG_TO_ISP;
[edit policy-options]
+ policy-statement AGG_TO_ISP {
+   term ACCEPT_AGG {
+     from {
+       protocol aggregate;
+       route-filter 10.42.0.0/16 exact;
+     }
+     then accept;
+   }
+   term REJECT_OTHERS {
+     then reject;
+   }
+ }
[edit]
cjones@R2# commit and-quit
commit complete
Exiting configuration mode
```

验证聚合已经被送往 AS 111 和 AS 222

1. 检查 R1 上的 Adj-RIB-Out 表:

```
cjones@R1> show route advertising-protocol bgp 172.16.0.1
inet 0: 11 destinations, 14 routes (11 active, 0 holddown, 0 hidden)
  Prefix                Nexthop              MED      Lclpref    AS path
* 10.42.0.0/16          Self                  0         0          I
```

2. 检查 R2 上的 Adj-RIB-Out 表:

```
cjones@R2> show route advertising-protocol bgp 172.18.0.1
inet 0: 11 destinations, 14 routes (11 active, 0 holddown, 0 hidden)
  Prefix                Nexthop              MED      Lclpref    AS path
* 10.42.0.0/16          Self                  0         0          I
```

使入向流量优先通过 ISP-B 进入网络

在 Junos 中令流量进入 AS 是通过 AS 路径预置来完成的, 而其配置则是已经为大家所喜闻乐见的统一策略架构。在本例中, 我们已经有了将汇聚路由输出到 ISP-A 的策略, 所以只需要在此基础上添加即可。

1. 在 R1 上修改输出策略以预置 AS:

```
cjones@R1> configure
Entering configuration mode
[edit]
cjones@R1# set policy-options policy-statement AGG_TO_ISP term ACCEPT_AGG then as-path-prepend
"65001 65001"
[edit]
cjones@R1# commit and-quit
commit complete
Exiting configuration mode
```

验证 AS 路径预置配置已经生效

在 Junos 中你可以轻易地查看 Adj-RIB-Out 表来检查你所修改的 AS 路径, 这张表显示了在策略应用之后通过 BGP 公告的前缀所发生的变化。

检查 Adj-RIB-Out 表:

```
cjones@R1> show route advertising-protocol bgp 172.16.0.1
inet 0: 11 destinations, 14 routes (11 active, 0 holddown, 0 hidden)
  Prefix                Nexthop          MED      Lclpref   AS path
* 10.42.0.0/16          Self              0         0         65001 65001 [65001] I
```

使出向流量优先通过 ISP-B 离开网络

1. 这里我们再次使用策略来修改由 BGP 学到的前缀, 在 R2 上使用一条简单的输入策略就可以提高来自 ISP-B 所有路由的 local preference。:

```
cjones@R2> configure
Entering configuration mode
[edit]
cjones@R2# set policy-options policy-statement ISPB-LOCALPREF then local-preference 110

2. 在 R2 上将这条策略应用为 BGP 组/邻居的输入策略, 并提交生效:

[edit]
cjones@R2# set protocols bgp group ISP-B neighbor 172.18.0.1 import ISPB-LOCALPREF
[edit]
cjones@R2# commit and-quit
commit complete
Exiting configuration mode
```

验证 local preference 已经被修改

1. 检查 R2 上的 RIB。要注意: 此时在 R2 上, 该 prefix 将不会再从 R1 学到。因为现在 R1 上到此 prefix 的最佳路由通过 R2 学到, 因此 R1 不会再把该路由重发布回 R2。

```
cjones@R2> show route protocol bgp
inet 0: 11 destinations, 13 routes (11 active, 0 holddown, 0 hidden)
+ = Active Route, - = Last Active, * = Both
111.111.111.0/24    *[BGP/170] 00:51:34, localpref 110
                    AS path: 222 I
                    > to 172.18.0.1 via ge-0/0/0.0
```

2. 在 R1 上检查 RIB, 确认首选路径是通过 R2 去往 ISP-B:

```
cjones@R1> show route protocol bgp
inet 0: 11 destinations, 14 routes (11 active, 0 holddown, 0 hidden)
+ = Active Route, - = Last Active, * = Both
111.111.111.0/24    *[BGP/170] 00:02:42, localpref 110, from 2.2.2.2
                    AS path: 222 I
                    > to 10.42.0.2 via ge-0/0/1.0
                    [BGP/170] 01:09:57, localpref 100
                    AS path: 111 I
                    > to 172.16.0.1 via ge-0/0/0.0
```

总结

在本章中我们使用 IOS 和 Junos 对一个简单网络进行了配置，并比较了两种操作系统之间的差异。很明显 Junos 在很大程度上依赖于策略配置来完成对 IGP 和 BGP 路由的操控。但是你也可以注意到在 Junos 配置的层次化中流露出的优雅。它不是 C 语言编程，也不比 IOS 更复杂或者更难掌握，它只是更强大。

你在此学到的知识可以应用与所有 Junos 设备，包括路由器、交换机、防火墙等等。在书末的资源列表也能为你提供帮助。

本书由比较基本任务开始，进而比较基本配置，最后比较构建简单网络拓扑，至此你应该已经对 Junos 有了一定的认识，迫不及待要投身其中了吧。组网的学问随着转向 Junos 而变得更简单了。

跟住去边度

www.juniper.net/dayone

在此可以免费下载本书的 PDF 和其它电子书版本。你也可以浏览其它的 Junos Day One 小册子，包括：

- Day One: Exploring the Junos CLI¹³
- Day One: Configuring Junos Basics
- Day One: Junos Tips, Techniques, and Templates
- This Week: Hardening Junos Devices

<http://forums.juniper.net/t5/IOS-to-Junos-I2J-Tips-Contest/bd-p/I2JTips>

由瞻博网络赞助的 J-Net 社区论坛专注于共享信息、最佳实践以及和有关瞻博网络产品、技术和解决方案的问题。这里你可以看到数十个有关从 IOS 到 Junos 转移技巧竞赛的参赛作品。

<http://www.juniper.net/techpubs/software/junos>

Junos 技术文档为你提供了解和配置 Junos 的全方位信息。

https://learningportal.juniper.net/juniper/user_activity_info.aspx?id=3310

Junos as a Second Language。这个课程专为熟悉 IOS 的工程师设计，在现有 IOS 配置知识的基础上，提供了对 Junos 高层次的概览，讲述其工作方式以及与 IOS 的差别。

<http://itunes.apple.com/us/podcast/junos-as-a-second-language/id276663160>

这是上述课程的 Podcast。

¹³译者注：该小册子的繁体中文版为 [《第一次使用就上手：Junos CLI 大探索》](#)。