dns服务

作者：牟建波（135342820@qq.com）

时间：2025-05-31

描述：日常自学笔记

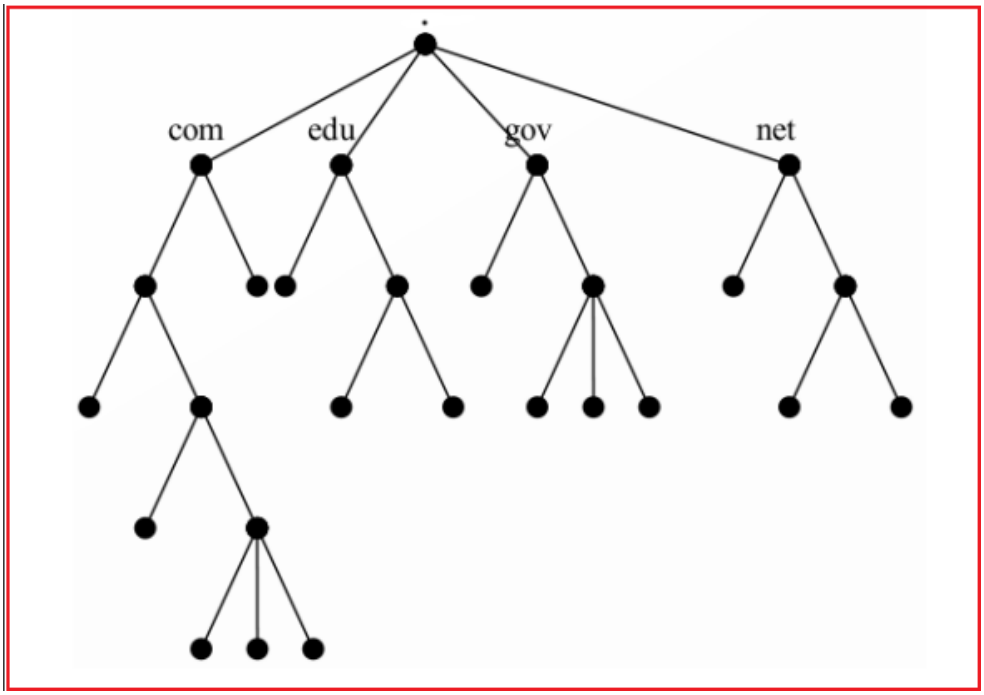
1.DNS基本概念

`DNS（Domain Name System，域名系统）`：是互联网的“电话簿”，它用于将人类易于记忆的域名（如 www.baidu.com）转换为计算机能够理解的 IP 地址（如 192.0.2.1）。DNS 使得人们能够通过域名访问网站，直接记域名比IP地址方便，而不必关心背后的IP地址

作用：

- **域名映射到IP地址**：DNS使得用户在浏览器中输入域名时，能够查询到相应的IP地址，从而连接到目标服务器。
- **反向解析**：除了域名到IP的正向解析，DNS 还支持 IP 地址到域名的反向解析，这对网络安全和故障排查等非常重要。
- **提供负载均衡**：通过 DNS，可以实现不同 IP 地址的轮换，帮助分散流量，提升访问速度和可靠性。
- **简化网络配置**：通过 DNS，管理员只需使用域名管理服务，简化了网络配置和维护。

2.DNS结构



2.1 根域 .（root）

在整个 DNS 系统的最上方一定是 .（小数点）这个 DNS 服务器（称为 root），也叫"根域"。它们不直接存储域名和IP地址的映射，而是存储指向顶级域（TLD）服务器的信息。

根域 （13台 全世界只有13台。1台为主根服务器，放置在美国。其余12台均为辅根服务器，其中9台放置在美国，欧洲2台，位于英国和瑞典，亚洲1台，位于日本。）

2.2 顶级域DNS服务器(TLD DNS Servers)

顶级域（如 `.com`，`.org`，`.net`，`.cn` 等）的DNS服务器存储了关于某个域名下的权威DNS服务器的信息。

例如，`.com` 域的TLD服务器会告诉你要去查找与某个 `.com` 域名相关的权威DNS服务器。```

- 常见的顶级域及国家

代码块

```
.com 商业机构
.net 网络
.org 非商业机构 www.centos.org www.kernel.org
.edu 教育机构
.gov 政府机关
.cn 中国域名
.us 美国域名
.ai 人工智能
.io 云计算
.mil 军事机构
```

除了顶级域名外， 还有二级域名（baidu.com）、三级域名（smartgo.net.cn）、四级域名（it.smartgo.net.cn）等

2.3 权威DNS服务器（Authoritative DNS Servers）

权威DNS服务器存储了某个域名的确切信息（如 IP 地址）。

当DNS查询请求到达这些服务器时，服务器会直接返回查询结果。它们对自己所管理的域名负责。

除了这些类型的DNS服务器外，其实还有递归DNS服务器、缓存DNS服务器、前向DNS服务器等

3.域名注册机构

国内基本用 阿里云 的 中国万网

代码块

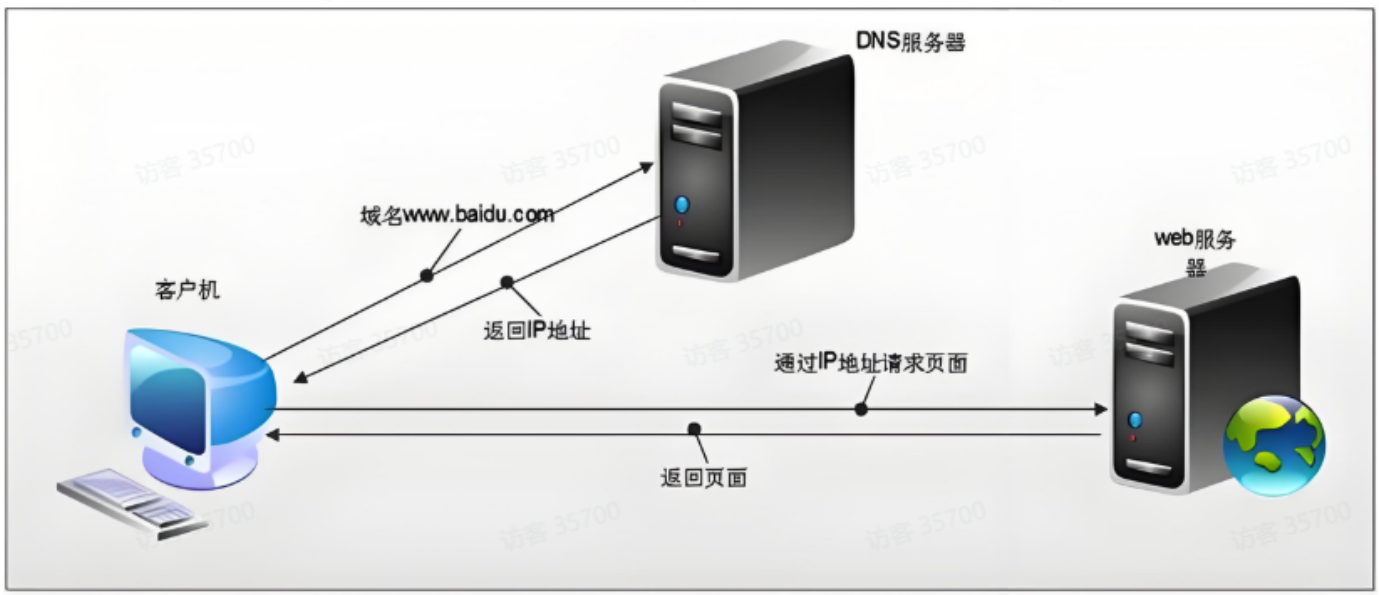
```
1. GoDaddy
网址：https://www.godaddy.com
特点：GoDaddy 是全球最大的域名注册商之一，提供多种域名后缀（TLD），并且有丰富的附加服务（如网站托管、SSL证书、邮件服务等）

2. Namecheap
网址：https://www.namecheap.com
特点：Namecheap 以价格实惠和良好的客户服务著称，支持多种域名后缀，且界面简单易用，适合初学者和中小企业

3. Google Domains
网址：https://domains.google
特点：由 Google 提供的域名注册服务，操作简单，支持许多不同的域名后缀，并与 Google 的其他服务（如 G Suite）紧密集成

4. Bluehost
网址：https://www.bluehost.com
特点：主要提供网站托管服务的公司，但也提供域名注册，适合需要托管服务的用户。通常提供域名注册折扣，适合首次购买网站建设套餐的用户

5. 阿里云（Alibaba Cloud）
网址：https://www.aliyun.com
特点：阿里云提供域名注册和云计算服务，特别适合需要在中国和亚太地区进行域名注册的用户。支持多种后缀，包括中文域名

6. 腾讯云
网址：https://cloud.tencent.com
特点：腾讯云是国内领先的云计算和域名注册服务商，适合在中国及全球范围内注册域名

7. Dynadot
```

```
网址：https://www.dynadot.com
特点：提供清晰简洁的界面，支持多种域名后缀。价格适中，提供一站式的域名管理服务

8. 1&1 IONOS
网址：https://www.ionos.com
特点：提供域名注册、托管和其他IT解决方案，适合需要综合服务的用户，尤其是在欧洲市场有较高的影响力

9. Hover
网址：https://www.hover.com
特点：提供简洁的域名购买和管理界面，专注于提供优质的客户服务和隐私保护，价格较为合理
```

4.DNS工作原理

DNS解析的过程通常包括以下几个步骤：

1. **客户端请求DNS解析**：用户的设备（如计算机或手机）发起DNS查询请求，询问特定域名（如`www.example.com`）对应的IP地址。
2. **递归查询**：如果查询的DNS服务器是递归DNS服务器，它将从根DNS服务器开始查找所需的信息。递归DNS服务器通过多个步骤向下查询，直到找到权威DNS服务器并获取IP地址。
3. **缓存机制**：递归DNS服务器通常会缓存查询结果，缓存期限根据DNS记录的TTL（Time to Live）值来决定。这样，后续的相同查询可以直接从缓存中获取结果，提高响应速度。
4. **返回最终结果**：递归DNS服务器或权威DNS服务器将最终的IP地址返回给客户端，客户端就可以通过该IP地址与目标服务器建立连接。



5.自建DNS服务器

目前公司内部，有很多的内部网站，仅能通过内部网络才可以访问到，但是每一个系统如果让员工通过IP访问，实在太麻烦了，公司希望建立一个内部DNS服务器，通过设置对应的域名，采用内部DNS服务器负责转发，同时该服务器也要支持访问外部网络

- node1：客户端
- node2：DNS服务器

5.1 DNS服务器软件介绍

DNS服务器的搭建更推荐使用 `BIND`

软件名称	类型	优点	缺点
BIND	权威/递归	功能强大，社区活跃，广泛使用	配置复杂，学习曲线陡峭
Unbound	递归	快速、安全、轻量	不支持权威DNS
dnsmasq	递归+DHCP	轻便易用，适合小型网络	功能不如BIND强大
PowerDNS	权威/递归	数据库支持好，适合自动化部署	学习成本略高
Knot DNS	权威	高性能、现代特性支持好	配置不如BIND普及

场景	建议软件
家庭/小型网络	dnsmasq
企业级递归DNS	Unbound / BIND（递归模式）
企业级权威DNS	BIND / PowerDNS / Knot DNS
自动化/数据库集成	PowerDNS
高性能、大规模权威DNS	Knot DNS

5.2 在DNS服务器上安装BIND

代码块

```
dnf install -y bind bind-utils
# bind：提供DNS服务器功能
```

```
3   # bind-utils：提供一些DNS查询工具，比如dig、nslookup、host
```

```
[root@node2 share]# dnf -y install bind bind-utils
Last metadata expiration check: 0:27:49 ago on Sun 17 Nov 2024 05:26:41 PM CST.
Dependencies resolved.
================================================================================
 Package                 Architecture    Version                 Repository  Size
================================================================================
Installing:
 bind                    x86_64          32:9.16.23-24.el9       appstream  505 k
 bind-utils              x86_64          32:9.16.23-24.el9       appstream  210 k
Installing dependencies:
 bind-dnssec-doc         noarch          32:9.16.23-24.el9       appstream   46 k
 bind-libs               x86_64          32:9.16.23-24.el9       appstream  1.2 M
 bind-license            noarch          32:9.16.23-24.el9       appstream   14 k
 fstrm                   x86_64          0.6.1-3.el9             appstream   28 k
 libmaxminddb            x86_64          1.5.2-4.el9             appstream   33 k
 libuv                   x86_64          1:1.42.0-2.el9          appstream  148 k
 protobuf-c              x86_64          1.3.3-13.el9            baseos      35 k
 python3-bind            noarch          32:9.16.23-24.el9       appstream   68 k
 python3-ply             noarch          3.11-14.el9             baseos     106 k
Installing weak dependencies:
 bind-dnssec-utils       x86_64          32:9.16.23-24.el9       appstream  118 k

Transaction Summary
================================================================================
Install  12 Packages
```

5.3 配置BIND主配置文件

代码块

```
1   # BIND主配置文件：/etc/named.conf
2   vim /etc/named.conf
```

代码块

```
1   # 以下为配置文件的默认配置信息
2   options {
3           listen-on port 53 { 127.0.0.1; };      # 指定 DNS 服务监听的 IP 地址 IPV4（内网地址）
4           listen-on-v6 port 53 { ::1; };         # 指定 DNS 服务监听的 IP 地址 IPV6 （内网地址）
5           directory       "/var/named";          # 设置 BIND 配置和区域文件存储的目录
6           dump-file       "/var/named/data/cache_dump.db";      # 指定 DNS 缓存的转储文件路径，存储服务器缓存的所有 DNS 记录
7           statistics-file "/var/named/data/named_stats.txt";    # 指定存储 BIND 统计信息的文件路径
8           memstatistics-file "/var/named/data/named_mem_stats.txt";      # 存储内存统计信息的文件路径
9           secroots-file   "/var/named/data/named.secroots";      # 存储安全根密钥的文件路径，用于 DNSSEC（DNS 安全扩展）
10          recursing-file  "/var/named/data/named.recursing";      # 存储递归查询状态的文件路径
11          allow-query     { localhost; };     # 限制查询源地址，只允许来自内网 IP 地址的查询
12
13          recursion yes;     # 启用递归查询,允许该 DNS 服务器处理外部域名查询。如果是权威 DNS 服务器，可以禁用此选项
14
```

```
        dnssec-validation yes;     # 启用 DNSSEC（DNS 安全扩展）验证，增加 DNS 查询的安全性

        managed-keys-directory "/var/named/dynamic";     # 存储动态管理的密钥（如 DNSSEC 密钥）的目录
        geoip-directory "/usr/share/GeoIP";     # 存储 GeoIP 数据的目录，用于地理位置相关的 DNS 配置

        pid-file "/run/named/named.pid";     # 存储 BIND 进程 ID 的文件路径
        session-keyfile "/run/named/session.key";     # 存储会话密钥的文件路径，用于 BIND 的会话加密


        include "/etc/crypto-policies/back-ends/bind.config";     # 包含与加密策略相关的配置文件，启用系统级别的加密策略
};

# 日志配置块
logging {
        channel default_debug {
                file "data/named.run";
                severity dynamic;
        };
};

# 根域区域配置
zone "." IN {
        type hint;
        file "named.ca";
};
```

代码块

```
# 调整以下内容(不要直接拷贝复制)：
options {
    listen-on port 53 { 127.0.0.1; 192.168.88.0/24; };  # 内部网络地址
    listen-on-v6 { none; };                             # 禁用IPv6，如果不需要的话
    allow-query    { 127.0.0.1; 192.168.88.0/24; };     # 允许来自内部网络的查询

    recursion yes;     # 启用递归查询

    forwarders {
        8.8.8.8;    # Google DNS
        114.114.114.114;    # 国内移动、电信和联通通用的dns
    };

```

```
    dnssec-validation no;

    # 控制区域传输权限，禁止外部直接访问区域
    allow-transfer { none; };
};

说明：
forwarders：配置转发 DNS 请求到外部 DNS 服务器（例如 Google DNS）

为啥设置 allow-transfer { none; }?
答：此配置是为了 增强安全性，防止未授权的服务器执行区域传输，从而保护 DNS 服务器的区域数据不被泄露或滥用

区域传输是 DNS 服务器之间的一种机制，用于将一个 DNS 区域的数据从主 DNS 服务器（master）同步到从 DNS 服务器（slave）。这通常用于设置 主从 DNS 服务器 配置，在多个 DNS 服务器之间同步域名记录
```

```
options {
        listen-on port 53 { 127.0.0.1; 192.168.88.0/24; };
        listen-on-v6 port 53 { none; };
        directory       "/var/named";
        dump-file       "/var/named/data/cache_dump.db";
        statistics-file "/var/named/data/named_stats.txt";
        memstatistics-file "/var/named/data/named_mem_stats.txt";
        secroots-file   "/var/named/data/named.secroots";
        recursing-file  "/var/named/data/named.recursing";
        allow-query     { 127.0.0.1; 192.168.88.0/24; };

        /*
         - If you are building an AUTHORITATIVE DNS server, do NOT enable recursion.
         - If you are building a RECURSIVE (caching) DNS server, you need to enable
           recursion.
         - If your recursive DNS server has a public IP address, you MUST enable access
           control to limit queries to your legitimate users. Failing to do so will
           cause your server to become part of large scale DNS amplification
           attacks. Implementing BCP38 within your network would greatly
           reduce such attack surface
        */
        recursion yes;

        forwarders {
                8.8.8.8;
                114.114.114.114;
         };

        dnssec-validation no;

        managed-keys-directory "/var/named/dynamic";
        geoip-directory "/usr/share/GeoIP";

        pid-file "/run/named/named.pid";
        session-keyfile "/run/named/session.key";

        /* https://fedoraproject.org/wiki/Changes/CryptoPolicy */
        include "/etc/crypto-policies/back-ends/bind.config";

        allow-transfer { none; };
};
```

5.4 配置内部区域注册文件

假设目前有内部网站需要配置内部转发：`internal.local 192.168.88.101`

内部正向解析注册文件：通过域名(主机名)解析到对应的IP地址

代码块

```
vim /etc/named.rfc1912.zones
# 添加以下内容:
zone "internal.local" IN {
    type master;
    file "/var/named/internal.local.db";
    allow-update { none; };
};

说明:
zone "internal.local" IN {
    表示一个 DNS 区域的声明
    internal.local 是此 DNS 区域的名称
    IN 指 Internet 类别，常用默认值

type master;
    指定此区域的类型为 主（master）区域
    主区域是该域名的权威数据源，DNS 数据直接从此服务器的配置文件加载

file "/var/named/internal.local.db";
    定义该区域的区域数据文件位置
    文件 internal.local.db 包含该区域的记录（如 A、MX、NS 等）

allow-update { none; };
    指定此区域是否允许动态更新
    none 意味着不允许任何动态更新，域名记录只能通过手动修改文件更新
```

```
zone "1.0.0.127.in-addr.arpa" IN {
        type master;
        file "named.loopback";
        allow-update { none; };
};

zone "0.in-addr.arpa" IN {
        type master;
        file "named.empty";
        allow-update { none; };
};

zone "internal.local" IN {
        type master;
tab缩进 file "/var/named/internal.local.db";
        allow-update { none; };
};
```

代码块

```
vim /var/named/internal.local.db
# 添加以下内容:
$TTL 86400
@    IN    SOA   ns1.internal.local. admin.internal.local. (
                  2024011701 ; Serial
                  3600       ; Refresh
                  1800       ; Retry
                  1209600    ; Expire
                  86400 )    ; Minimum TTL

      IN    NS    ns1.internal.local.
ns1    IN    A     192.168.88.102
web1   IN    A     192.168.88.101
web2   IN    A     192.168.88.101
@      IN    A     192.168.88.101


说明:
$TTL 默认 TTL (Time To Live)。
所有资源记录的默认缓存时间，单位为秒。
这里设置为 86400（一天）。客户端在缓存数据时会遵循此值。

SOA 记录 (Start of Authority)
        @: 当前区域的根域（即 internal.local）。
        IN: 表示 Internet 类别。
```

```
        SOA: 开始授权记录，定义该区域的关键元信息。
        ns1.internal.local.          主域名服务器的 FQDN（全限定域名）。
    admin.internal.local.          管理员的邮箱地址，@ 替换为 .（即 admin@internal.local）。
    2024011701          序列号，每次修改区域文件时需递增，用于从服务器检测更新。
    3600          刷新时间，从服务器多久检查主服务器是否有更新（秒）。
    1800          重试时间，从服务器在刷新失败后再次尝试的等待时间（秒）。
    1209600          过期时间，从服务器在无法联系主服务器时数据失效的时间（秒）。
    86400          最小 TTL，未覆盖的记录的默认缓存时间（秒）。

NS 记录（Name Server）
        指定该区域的域名服务器地址。
        ns1.internal.local.: 表示 internal.local 的主域名服务器。

A 记录（Address）：定义域名到 IPv4 地址的映射。
        ns1.internal.local. 解析为 192.168.88.102
        web1.internal.local. 和 web2.internal.local. 都解析为 192.168.88.101


扩展常见的DNS记录类型：
A 记录（Address Record）：将域名映射到 IPv4 地址。
AAAA 记录：将域名映射到 IPv6 地址。
CNAME 记录（Canonical Name Record）：为域名提供别名。
MX 记录（Mail Exchange Record）：指定邮件服务器的地址。
NS 记录（Name Server Record）：指定域名的权威 DNS 服务器。
SOA 记录（Start of Authority Record）：定义 DNS 区域的起始权威信息。
PTR 记录（Pointer Record）：用于反向 DNS 查找，将 IP 地址映射回域名。
TXT 记录（Text Record）：存储任意文本数据，常用于 SPF、DKIM 等验证机制。
```

```
$TTL 86400
@    IN    SOA   ns1.internal.local. admin.internal.local. (
                  2024011701 ; Serial
                  3600       ; Refresh
                  1800       ; Retry
                  1209600    ; Expire
                  86400 )    ; Minimum TTL

     IN    NS    ns1.internal.local.
ns1    IN    A     192.168.88.102
web1   IN    A     192.168.88.101
web2   IN    A     192.168.88.101
~
```

☁ 内部反向解析注册文件：通过IP能够解析到对应的域名(主机名)

代码块

```
vim /etc/named.rfc1912.zones
# 添加以下配置
```

```
zone "88.168.192.in-addr.arpa" IN {
    type master;
    file "/var/named/192.168.88.rev";
    allow-update { none; };
};

说明:
zone "88.168.192.in-addr.arpa" IN {
这是一个 反向区域（reverse zone）的声明。
反向区域用于将 IP 地址（IPv4）转换为域名，这和正向 DNS 查询（将域名转换为 IP 地址）是相反的。
88.168.192.in-addr.arpa：这是 192.168.88.x IP 地址段的反向区域名称。反向查找区域的命名规则是：将 IP 地址的每个八位字节倒序并加上 .in-addr.arpa 后缀。例如，192.168.88.x 的反向区域名称就是 88.168.192.in-addr.arpa。

type master;
指定该区域是 主（master）区域，也就是说，这是该区域的权威 DNS 服务器，并且数据会从本地文件加载

file "192.168.88.rev";
这是该区域的区域数据文件路径。
192.168.88.rev 文件包含了反向解析记录，用于将 IP 地址（如 192.168.88.101）映射到对应的域名

allow-update { none; };
allow-update 指定是否允许动态更新。在这里设置为 none，意味着不允许任何动态更新
这是一种安全配置，防止未经授权的客户端修改 DNS 记录
```

```
zone "internal.local" IN {
        type master;
        file "/var/named/internal.local.db";
        allow-update { none; };
};

zone "88.168.192.in-addr.arpa" IN {
    type master;
    file "192.168.88.rev";
    allow-update { none; };
};
```

代码块

```
vim /var/named/192.168.88.rev
# 添加以下配置
$TTL 86400
@    IN    SOA   ns1.internal.local. admin.internal.local. (
                  2024011701 ; Serial
                  3600       ; Refresh
                  1800       ; Retry
```

```
                    1209600    ; Expire
                    86400 )    ; Minimum TTL

        IN    NS    ns1.internal.local.
102     IN    PTR   ns1
101     IN    PTR   web1
101     IN    PTR   web2
101     IN    PTR   @
```

语法检查

代码块

```
# 配置文件语法检查
named-checkconf /etc/named.conf

# 如果报了错误，一般都是由于配置文件丢失内容导致语法结构不对，请检查配置文件
/etc/named.conf  【大概率是该文件的问题】
/etc/named.rfc1912.zones

# 区域文件语法检查
named-checkzone internal.local /var/named/internal.local.db
named-checkzone 88.168.192.in-addr.arpa /var/named/192.168.88.rev
```

```
[root@node2 ~]# named-checkconf /etc/named.conf
[root@node2 ~]#
```

```
[root@node2 ~]# named-checkzone internal.local /var/named/internal.local.db
zone internal.local/IN: loaded serial 2024011701
OK
[root@node2 ~]# named-checkzone 88.168.192.in-addr.arpa /var/named/192.168.88.rev
zone 88.168.192.in-addr.arpa/IN: loaded serial 2024011701
OK
[root@node2 ~]#
```

5.5 启动BIND服务

代码块

```
systemctl start named
systemctl enable --now named
systemctl status named
```

```
[root@node2 etc]# systemctl start named
[root@node2 etc]# systemctl enable named
Created symlink /etc/systemd/system/multi-user.target.wants/named.service → /usr/lib/systemd/system/named.service.
[root@node2 etc]# systemctl status named
● named.service - Berkeley Internet Name Domain (DNS)
     Loaded: loaded (/usr/lib/systemd/system/named.service; enabled; preset: disabled)
     Active: active (running) since Fri 2024-12-20 15:54:01 CST; 12s ago
   Main PID: 119450 (named)
      Tasks: 14 (limit: 22929)
     Memory: 34.8M
        CPU: 60ms
     CGroup: /system.slice/named.service
             └─119450 /usr/sbin/named -u named -c /etc/named.conf

12月 20 15:54:01 node2.itcast.cn named[119450]: FORMERR resolving './NS/IN': 192.112.36.4#53
12月 20 15:54:01 node2.itcast.cn named[119450]: FORMERR resolving './NS/IN': 202.12.27.33#53
12月 20 15:54:01 node2.itcast.cn named[119450]: FORMERR resolving './NS/IN': 192.33.4.12#53
12月 20 15:54:01 node2.itcast.cn named[119450]: FORMERR resolving './NS/IN': 198.97.190.53#53
12月 20 15:54:01 node2.itcast.cn named[119450]: FORMERR resolving './NS/IN': 192.5.5.241#53
12月 20 15:54:01 node2.itcast.cn named[119450]: FORMERR resolving './NS/IN': 193.0.14.129#53
12月 20 15:54:01 node2.itcast.cn named[119450]: FORMERR resolving './NS/IN': 192.203.230.10#53
12月 20 15:54:01 node2.itcast.cn named[119450]: FORMERR resolving './NS/IN': 192.36.148.17#53
12月 20 15:54:01 node2.itcast.cn named[119450]: FORMERR resolving './NS/IN': 199.7.83.42#53
12月 20 15:54:01 node2.itcast.cn named[119450]: resolver priming query complete
[root@node2 etc]#
```

5.6 配置防火墙

代码块

```
# 确保防火墙允许 53 端口的 UDP 和 TCP 流量，这样 DNS 请求才能到达服务器
firewall-cmd --zone=public --add-port=53/udp --permanent
firewall-cmd --zone=public --add-port=53/tcp --permanent
firewall-cmd --reload

或者:
firewall-cmd --zone=public --add-service=dns --permanent
firewall-cmd --reload

# 查看规则信息:
firewall-cmd --list-all
```

```
[root@node2 etc]# firewall-cmd --zone=public --add-service=dns --permanent
success
[root@node2 etc]# firewall-cmd --reload
success
[root@node2 etc]# firewall-cmd --list-all
public (active)
  target: default
  icmp-block-inversion: no
  interfaces: ens160
  sources:
  services: cockpit dhcpv6-client dns nfs samba ssh
  ports:
  protocols:
  forward: yes
  masquerade: no
  forward-ports:
  source-ports:
  icmp-blocks:
  rich rules:
[root@node2 etc]#
```

5.7 客户端配置

代码块

```
# 配置客户端的 DNS 服务器地址，让内部网络中的设备指向你新搭建的 DNS 服务器
[node1]
vim /etc/NetworkManager/system-connections/ens160.nmconnection

# 修改完成后， 重启网卡
systemctl restart NetworkManager

nmcli device show ens160
```

```
[connection]
id=ens160
uuid=96d5d4dd-0bac-3463-914e-53ae009df502
type=ethernet
autoconnect-priority=-999
interface-name=ens160
timestamp=1732892131

[ethernet]

[ipv4]
method=manual
addresses=192.168.88.101/24
gateway=192.168.88.2
dns=192.168.88.102;

[ipv6]
method=ignore

[proxy]
```

```
[root@node1 ~]# nmcli device show ens160
GENERAL.DEVICE:                         ens160
GENERAL.TYPE:                           ethernet
GENERAL.HWADDR:                         00:0C:29:F6:8B:1D
GENERAL.MTU:                            1500
GENERAL.STATE:                          100（已连接）
GENERAL.CONNECTION:                     ens160
GENERAL.CON-PATH:                       /org/freedesktop/NetworkManager/ActiveConnection/2
WIRED-PROPERTIES.CARRIER:               开
IP4.ADDRESS[1]:                         192.168.88.101/24
IP4.GATEWAY:                            192.168.88.2
IP4.ROUTE[1]:                           dst = 192.168.88.0/24, nh = 0.0.0.0, mt = 100
IP4.ROUTE[2]:                           dst = 0.0.0.0/0, nh = 192.168.88.2, mt = 100
IP4.DNS[1]:                             192.168.88.102
IP6.ADDRESS[1]:                         fe80::20c:29ff:fef6:8b1d/64
IP6.GATEWAY:                            --
IP6.ROUTE[1]:                           dst = fe80::/64, nh = ::, mt = 256
[root@node1 ~]#
```

5.8 测试服务器功能

代码块

```
# 在node1安装bind-utils
dnf install -y bind-utils

# 使用dig或nslookup命令测试DNS查询，确保内部域名解析工作正常
dig web1.internal.local
或
nslookup web1.internal.local
```

```
[root@node1 ~]# dig web1.internal.local

; <<>> DiG 9.16.23-RH <<>> web1.internal.local
;; global options: +cmd
;; Got answer:
;; WARNING: .local is reserved for Multicast DNS
;; You are currently testing what happens when an mDNS query is leaked to DNS
;; ->>HEADER<<- opcode: QUERY, status: NOERROR, id: 58612
;; flags: qr aa rd ra; QUERY: 1, ANSWER: 1, AUTHORITY: 0, ADDITIONAL: 1

;; OPT PSEUDOSECTION:
; EDNS: version: 0, flags:; udp: 1232
; COOKIE: 5cfb6a9b08488ca401000000676524ebb9c99c81e1a6508c (good)
;; QUESTION SECTION:
;web1.internal.local.           IN      A

;; ANSWER SECTION:
web1.internal.local.    86400   IN      A       192.168.88.101

;; Query time: 0 msec
;; SERVER: 192.168.88.102#53(192.168.88.102)
;; WHEN: Fri Dec 20 16:03:55 CST 2024
;; MSG SIZE  rcvd: 92

[root@node1 ~]#
```