DNS

Рассмотрим принцип работы DNS, типы записей и базовую настройку BIND.

Оглавление

Доменные имена

Аббревиатура DNS расшифровывается как Domain Name System. Если говорить в общем, то это глобальное распределённое хранилище ключей и значений. Один из наиболее распространённых ключей — имя хоста (hostname), а соответствующее ему значение — IP-адрес. Изначально основная цель DNS — преобразование (резолв) доменных имен в IP-адреса и наоборот — IP в DNS. Но сейчас этот функционал гораздо шире.

Без DNS, хостнеймов и доменных имён нам пришлось бы ссылаться на все ресурсы в интернете по их IP-адресам, так как само по себе сетевое взаимодействие происходит между двумя IP-адресами, а не доменными именами. Получается, что пользователям пришлось бы запоминать IP-адреса их любимых ресурсов, что, конечно, тренирует память, но не совсем удобно. Реклама выглядела бы так:

Получается, что именно людям необходимы и имена хостов и служба доменных имен, а также универсальный механизм для сопоставления между адресами и именами, чтобы наши устройства получали знания обо всех именах в интернете.

Существует также ещё один аспект применения DNS, появившийся сравнительно недавно, который выходит за рамки протокола просто для сопоставления имён с IP-адресами и обратно. DNS теперь будет всё чаще использоваться для публикации метаданных.

Благодаря своей повсеместности и хорошей масштабируемости, особенно в сочетании с DNSSEC для аутентификации ответов, DNS хорошо подходит для публикации каких-либо других данных, которые будут искать приложения и клиенты. Например, необходимых для процессов аутентификации, репутации и шифрования. Это могут быть сертификаты X.509, ключи PGP / GPG, списки DNS-based Real-Time Blackhole Lists (RBLs), и response policy zones (RPZs). Относительно широко распространённая адаптация SPF и DKIM для работы почты и борьбы со спамом как раз подтверждает это новое применение DNS.

Что такое доменное имя

Как вы уже знаете, доменное имя — буквенно-цифровая строка, которая сопоставляется при помощи системы доменных имён (DNS) с другими данными — например с IP-адресом.

Итак, спрашивая, что такое домен, мы получаем достаточно очевидный ответ — *example.com*.

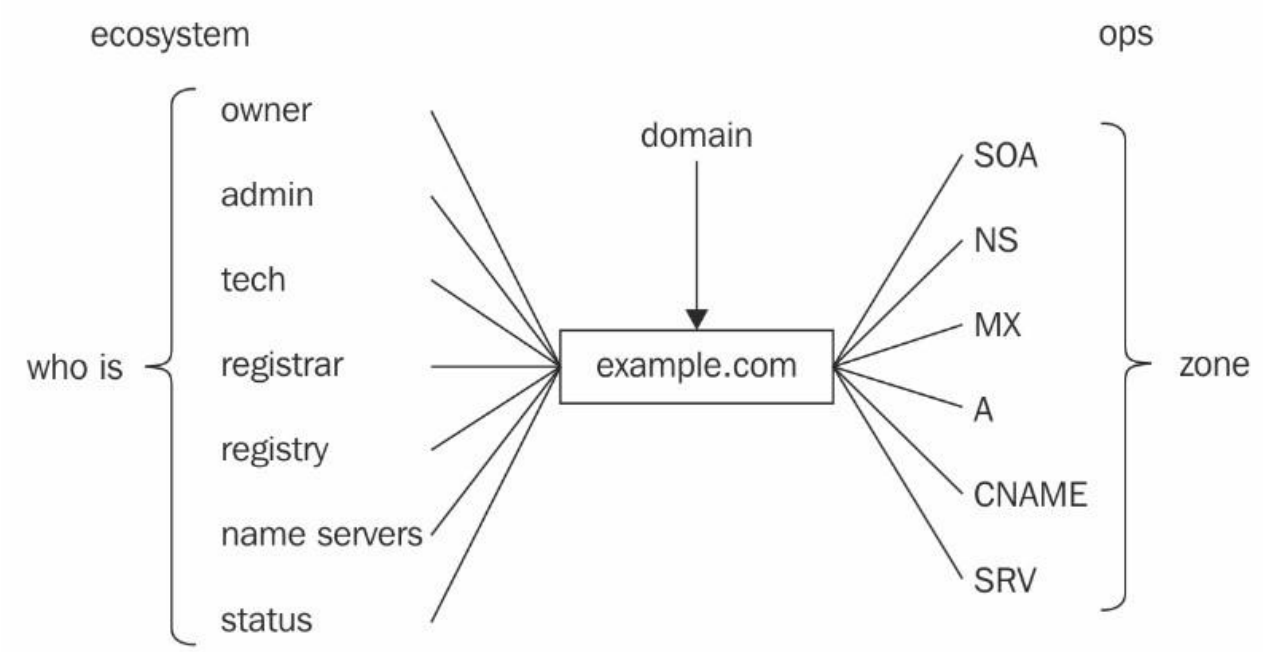
Однако, когда мы углубляемся в специфику протокола DNS и документацию, которая его описывает, мы начинаем сталкиваться с некоторыми странными нюансами с точки зрения формальной спецификации домена по сравнению с тем, что вы на самом деле можете зарегистрировать онлайн у какого-либо регистратора домена как ваше доменное имя.

Подчёркивания разрешены в доменных именах и некоторые типы DNS-записей используют их, например, тип записи SRV:

```
_xmpp-client._tcp.example.com.
```

Но если вы пойдёте к регистратору доменных имён и попробуете зарегистрировать имя с подчёркиванием, сделать это у вас не получится. Вы можете использовать дефис, но только если имя не начинается с дефиса и не заканчивается им.

Это один из примеров разницы между тем, что называется «экосистемой доменных имён», которая включает в себя регистраторов, реестры и надзорные органы, и самой службой DNS, которая работает на ваших серверах и клиентах:



Как видно на картинке, у нас для доменного имени существуют 2 части, одна из которых административная. В ней записаны регистратор доменного имени, владелец этого имени, статус и так далее. Часть справа также содержит различные типы записей, но они необходимы уже для преобразования доменного имени.

Generic TLDs (gTLD)

Изначально домены .COM, .NET и .ORG имели некую логику в названии (.COM для коммерческих организаций, .NET для сетевой инфраструктуры и .ORG для некоммерческих организаций), но сегодня эти различия размыты, и кто угодно может регистрировать любые доменные имена вне зависимости от страны расположения и цели использования домена.

Например, домен .CO, который зачастую используется для обозначения компании (CO = COmpany), на самом деле является доменом страны Colombia. Домен .tv, не был придуман для обозначения телевидения, а, так же как и .CO, является country code top layer domain (ccTLD) принадлежащий маленькому островному государству Tuvalu. Ещё одним примером может служить .WS = Website, являющийся ccTLD страны Western Samoa.

Получается, что домены, которые изначально были придуманы как ccTLD, выступают в роли generic TLD, не привязываясь к какой-либо определенной стране. A generic TLD (gTLD) не обязательно подразумевают именно то использование, которое предполагалось при создании домена.

Country Code TLDs (ccTLD)

Каждая страна в мире имеет свой собственный TLD в виде двухбуквенного обозначения, описанного в стандарте ISO31664. При этом ещё не все домены были делегированы регистраторам.

Делегированием ccTLD доменов на соответствующие DNS-сервера занимается ICANN (Internet Corporation for Assigned Names and Numbers), при этом каждая страна вправе установить свои собственные правила управления своими ccTLD.

Некоторые ccTLD, такие как канадский .CA, китайский .CN, или американский .US, требуют локального присутствия, то есть, домены внутри этих TLD могут быть зарегистрированы гражданами этих стран для внутренних компаний.

Остальные страны не столь строги в правилах регистрации доменных имён и, как уже говорилось ранее, кто угодно может зарегистрировать домен внутри ccTLD для любых целей.

New TLDs

В 2013 году официально стало возможным регистрировать практически любой TLD в ICANN. В 2014м любая компания могла подать заявку на регистрацию домена верхнего уровня (TLD), который ей хочется. Это привело к появлению тысяч новых TLD: .wtf., website, .press, .rocks, .support, .email, .pics, .red, .blue

Полный список TLD.

Internationalized TLDs (IND TLD)

Интернационализированные доменные имена содержат символы, которые находятся за пределами обычного буквенно-цифрового набора, например, символы с акцентами или не англоязычные объекты.

Поскольку метки в DNS кодируются в ASCII, эти типы объектов должны быть преобразованы в представление ASCII, прежде чем их можно будет использовать в системе DNS. Это достигается при помощи преобразования их в punycode.

Punycode использует функцию, вызываемую в ASCII для удаления символов, которые необходимо кодировать, и добавляет их потом к оставшейся строке, разделяя части дефисом. Процесс описан в RFC 3492 (https://www.ietf.org/rfc/rfc3492.txt). Благодаря этому мы можем открывать такие сайты, как motörhead.com, в котором, как вы видите, есть не-ASCII символ ö.

При помощи punycode слово motörhead будет преобразовано в motorhead-p4a. Одновременно мы должны объяснить DNS-серверу, что это не просто доменное имя с дефисом, а по факту преобразованное имя с non-ASCII символом. Это делается при помощи добавления префикса xn--.
Получается, что motörhead.com и xn--motorhead-p4a.com — одно и то же.
Утилитами для перевода таких доменных имён можно воспользоваться онлайн, например,

Рассмотрим на примере IND TLD Гонконга .香港. Попробуем узнать, какие name servers (ns) отвечают за этот домен:

```
$ host -t ns .香港
host: '.香港' is not a legal name (empty label)
```

Переведём этот домен в punycode при помощи упомянутого сайта и получим значение .xn--j6w193g.

```
$ host -t ns xn--j6w193g xn--j6w193g
name server v.hkirc.net.hk. xn--
j6w193g name server u.hkirc.net.hk.
xn--j6w193g name server
t.hkirc.net.hk. xn--j6w193g name
server d.hkirc.net.hk. xn--j6w193g
name server c.hkirc.net.hk. xn--
j6w193g name server x.hkirc.net.hk.
xn--j6w193g name server
y.hkirc.net.hk. xn--j6w193g name
server z.hkirc.net.hk.
```

Infrastructure TLDs

Один из самых распространённых и известных примеров инфраструктурных TLD — .arpa (Address and Routing Parameter Area). Это самый первый TLD, который должен был стать временной мерой для облегчения перехода от первоначальной сети Arpanet к системе доменных имён, которую мы имеем сегодня.

Как это обычно и происходит со временными решениями, процесс застрял в первой итерации и зона .arpa используется и по сей день:

- .in-addr.arpa — обратная зона, используется для преобразования IP-адреса в доменное имя,
- .e164.arpa используется для маппинга телефонных номеров, ● .ip6.arpa — обратная зона для IPv6.

Типы DNS-серверов

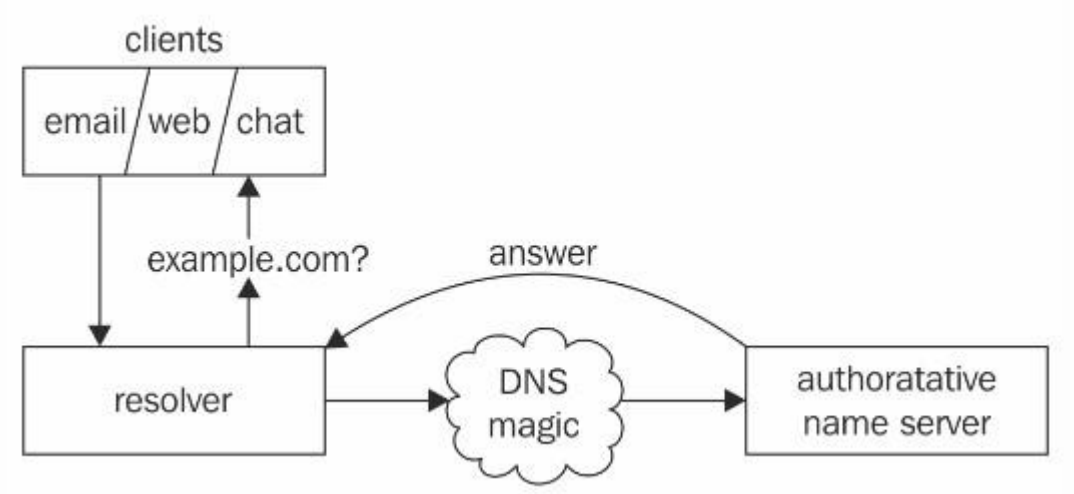
Неймсерверы могут быть выделены в две достаточно широкие группы в зависимости от того, какую функцию они выполняют:

- Резолверы или рекурсоры (resolvers или recursors) — делают запросы от имени своих клиентов и возвращают им ответы.
- Авторитетные DNS-серверы (authoritative nameservers) хранят у себя файлы зон, за которые отвечают. Отвечают на запросы, поступающие для этих зон.

.

DNS resolvers

Каждый раз, когда вы отправляете электронное письмо, посещаете веб-страницу, отправляете или получаете мгновенное сообщение, текст, или SMS, или что-либо ещё, связанное с интернетом, ваше приложение должно знать, где именно находится сервер/телефонный шлюз/почтовый сервер. Так как приложение для нашего удобства использует доменные имена, а соединение происходит при помощи IP-адресов, оно должно обратиться с запросом к DNS-серверу. Компьютеры, приложения или службы, которые задают такого рода вопросы, называются резолверами.



На схеме видно, что клиент (например, браузер, чат или почтовый клиент) обращается к резолверу, которым может быть отдельная служба в операционной системе, уже взаимодействующей с DNS-инфраструктурой.

Stub resolver (тупиковый резолвер)

Находятся на устройстве/операционной системе и обрабатывают запросы DNS для операционной системы. Самим приложениям обычно не нужно беспокоиться, откуда поступят ответы на их DNS-запросы, или как они их получат. Всё, что они делают — используют встроенные вызовы к операционной системе для получения IP-адреса, например, gethostbyaddr() или gethostbyname().

Операционная система получает запрос от приложения (query) и, используя урезанный набор функций, обращается к DNS-серверу (например, DNS-сервер провайдера) который уже умеет полноценно взаимодействовать с инфраструктурой DNS и, следовательно, преобразовать доменное имя в IP-адрес.

Caching resolver (кеширующий резолвер)

Кеширующий резолвер добавляет ответы себе в кеш, чтобы обслуживать последующие запросы на такое же имя уже из кеша. Это позволяет сократить время на поиск IP-адреса и снизить нагрузку на DNS-инфраструктуру. Примером такого сервера может служить ваш домашний роутер.

Full resolver (полный резолвер)

В отличие от stub resolver, полный резолвер может проследовать по всему пути разрешения доменного имени и вернуть ответ клиенту —тупиковому резолверу. Полные резолверы — серверы имён, которые находят ответы на запросы DNS, а не перенаправляют их куда-либо ещё. Кроме этого, полные резолверы могут кешировать уже сделанные запросы, чтобы отвечать на аналогичные запросы других клиентов из кеша (это ускоряет поиск IP-адреса), что напрямую влияет на удовлетворённость пользователя.

Ответ на запрос кешируется на время, которое указано в ответе от авторитетного сервера. Параметр, который за это отвечает, называется Time To Live (TTL). Со стороны авторитетного DNS-сервера TTL может задаваться на всю зону целиком, либо на каждую из записей внутри зоны отдельно (Resource Records — RR), но сам резолвер видит только конечный результат. Например, при обращении к локальному резолверу мы увидим, что запись имеет TTL 300 секунд, так как используется локальный резолвер, и в настройках операционной системы указано время кеширования 300 сек.

```
$ dig geekbrains.ru
;; QUESTION SECTION:
;geekbrains.ru.                 IN      A

;; ANSWER SECTION:
geekbrains.ru.          300     IN      A       5.61.239.22 geekbrains.ru.
300     IN      A       5.61.239.21
```

Поэтому все последующие запросы будут обработаны из локального кеша операционной системы. Как только таймер TTL дойдет до нуля, опять потребуется квери авторитетного DNS-сервера geekbrains.ru, а это займёт какое-то время.

Negative cache

Когда резолвер отправляет запрос (query) на имя, которое не существует (имя может и существовать, но не для запрошенного типа записи), он получит отрицательный ответ, содержащий start-of-authority-запись от авторитетного DNS-сервера. Это значит, что если мы спрашиваем несуществующую запись host.example.com, то авторитетная секция ответа будет включать в себя SOA-запись домена example.com

```
$ dig host.example.com @8.8.8.8
```

```
; <<>> DiG 9.10.3-P4-Ubuntu <<>> host.example.com
@8.8.8.8 ;; global options: +cmd ;; Got answer:
;; ->>HEADER<<- opcode: QUERY, status: NXDOMAIN, id: 28000
;; flags: qr rd ra ad; QUERY: 1, ANSWER: 0, AUTHORITY: 1, ADDITIONAL: 1

;; OPT PSEUDOSECTION:
; EDNS: version: 0, flags:; udp: 512
;; QUESTION SECTION:
;host.example.com.              IN      A

;; AUTHORITY SECTION: example.com.  853   IN    SOA   ns.icann.org.
noc.dns.icann.org. 2019101506 7200 3600 1209600 3600
```

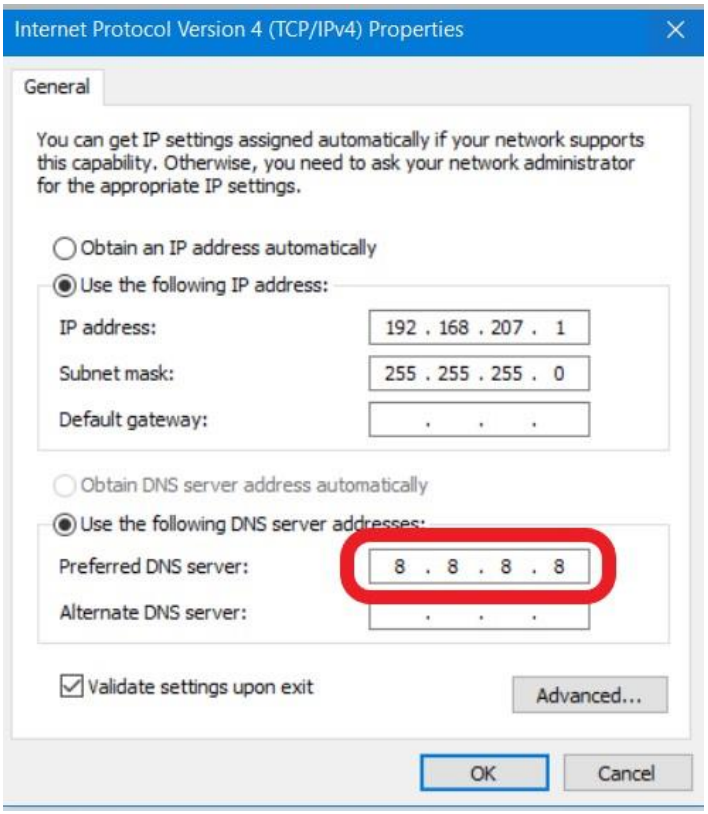
Отрицательный ответ может быть получен, если запрашиваемое имя не существует (NXDOMAIN) или если оно существует не с запрошенным типом RR (например, мы запрашиваем тип записи CNAME, а запись на такое имя только A). В последнем случае код ответа будет NOERROR, но наличие SOA RR в ответе подразумевает NODATA, то есть, ответа на заданный запрос нет. Например, мы знаем, что существует запись A для домена example.com; запросим CNAME-запись:

```
$ dig example.com @8.8.8.8 CNAME

; <<>> DiG 9.10.3-P4-Ubuntu <<>> example.com @8.8.8.8
CNAME ;; global options: +cmd ;; Got answer:
;; ->>HEADER<<- opcode: QUERY, status: NOERROR, id: 29662
;; flags: qr rd ra ad; QUERY: 1, ANSWER: 0, AUTHORITY: 1, ADDITIONAL: 1

;; OPT PSEUDOSECTION:
; EDNS: version: 0, flags:; udp: 512
;; QUESTION SECTION:
;example.com.                   IN      CNAME

;; AUTHORITY SECTION: example.com.  663   IN    SOA   ns.icann.org.
noc.dns.icann.org. 2019101506 7200 3600 1209600 3600
```

Как видите, статус ответа — NOERROR, но ответа на вопрос мы не видим, вместо него находится Authority section с SOA для домена example.com.

Значение TTL в негативном кеше — время, в течение которого резолвер не будет перезапрашивать информацию у авторитетного DNS-сервера. Всем своим клиентам резолвер будет отвечать на их запросы отрицательно, что такого домена не существует.

Резолверы часто прозрачны для конечных пользователей. Адрес полного резолвера зачастую указывается в настройках вашей операционной системы либо вручную, либо назначается при помощи протокола DHCP. В Linux настройки хранятся в файле /etc/resolv.conf, либо в конфигурационном файле интерфейса. В Windows:



Authoritative nameservers

Другой компонент магического процесса DNS-лукапа, который приводит к успешному разрешению доменного имени — авторитетные серверы имён. Они содержат данные зон для запрашиваемых имён и отвечают на эти запросы для всех зон, для которых они являются авторитетными. То есть, если мы владеем доменом example.com, у нас должен быть авторитетный DNS-сервер, который будет отвечать за преобразование домена example.com в IP-адрес, а также за преобразование любых субдоменов (abc.example.com, zxc.example.com, whatever.example.com).

Авторитетные DNS-серверы делятся на два вида — master и slave (primary/secondary) — для масштабирования и удобства управления крупными DNS-инфраструктурами.

Master nameserver (primary)

Master — авторитетный сервер имён, который содержит фактическую информацию о DNS-зонах, откуда в дальнейшем все остальные авторитетные серверы имён для нашего домена получают свою копию. Традиционно, поскольку подавляющее большинство серверов имён по-прежнему используют BIND, это означает, что сам DNS-сервер — источник данных о зоне и фактически текстовый файл. Slave DNS-сервера хранят у себя копию этого файла и следят за изменениями у мастера, чтобы в случае изменений получить себе новейшую копию.

Если в качестве DNS-сервера используется не BIND, а, например, PowerDNS, то такое master/slave-взаимодействие будет немного размытым, так как бэкенд для зоны — база данных, из которой мастер и слейв могут читать информацию о зоне.

Hidden master (hidden primary)

Так как сохранение в целостности файла зоны достаточно критично, для крупных имплементаций master DNS-сервер не отвечает на запросы клиентов, более того, он не опубликован в интернете. Такая ситуация называется hidden master.

Всё, чем занимается скрытый мастер — обслуживание файла с зоной и передача обновлённой информации slave-серверам, которые обслуживают запросы конечных клиентов.

Сейчас почти все авторитетные DNS-серверы, доступные из интернета, получают информацию о зонах от hidden master-серверов.

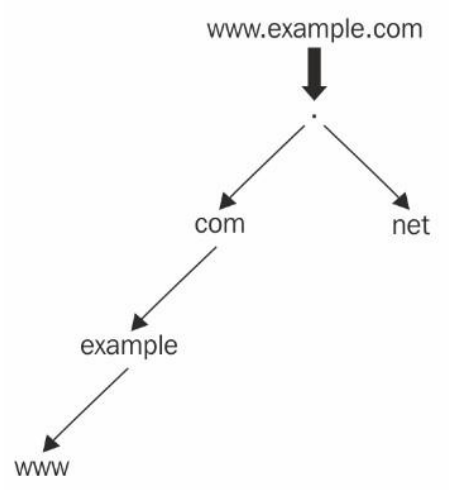
Slave nameserver (secondary)

Секондари неймсервер — сервер, который получает копию информации о зоне от мастера. Если говорить про BIND, это происходит при помощи трансфера зоны Authoritative Transfer — AXFR (rfc 5936) или Incremental Zone Transfer — IXFR (rfc 1995). Как только зона на мастере меняется, он при помощи NOTIFY-пакета для каждой NS-записи в зоне (для всех секондари-серверов) известит об изменениях.

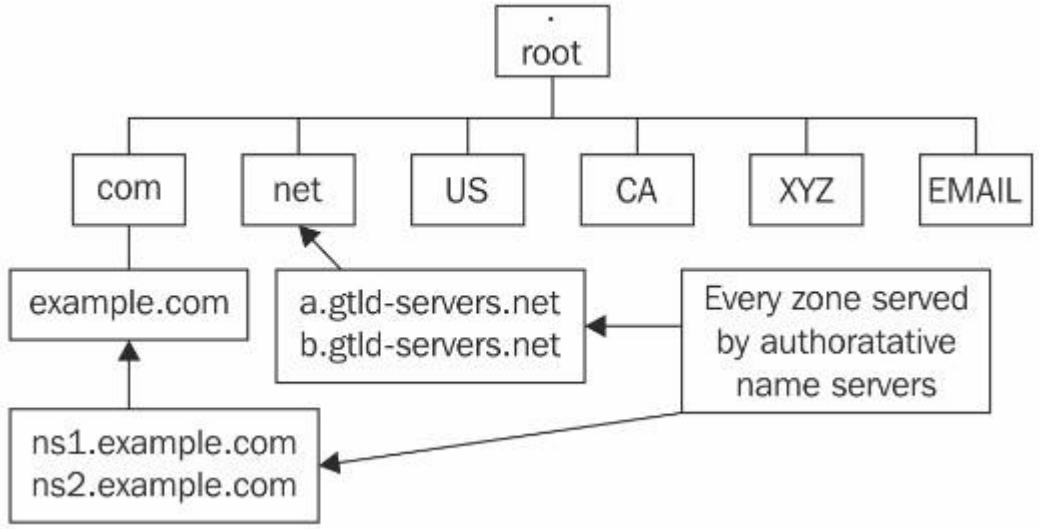
Механизмы передачи информации о зонах встроены в сам протокол DNS. Они обязаны своим происхождением ранним дням существования DNS и были разработаны вместо надежных методов синхронизации данных между серверами, многие из которых существуют сегодня. Один из таких методов — синхронизация базы данных (MySQL, PostgreSQL и т. д.), которым пользуется PowerDNS.

DNS Query

Иерархия доменных имён представляет собой перевёрнутое дерево, вершина которого — точка «.». Кроме этого, каждый из уровней в иерархии доменных имён также разделён точкой, то есть иерархия в доменных именах распространяется справа налево.



За каждый из уровней иерархии может отвечать отдельный набор DNS-серверов. Так происходит для корневых (root) DNS-серверов (которые отвечают за точку в самом конце доменного имени), а также для TLD и доменов второго уровня.



Корневые DNS-серверы, отвечая за точку, находятся на вершине иерархии. Сразу же за ними находятся DNS-серверы, отвечающие за вверенные им TLD. Посмотреть список авторитетных DNS-серверов, отвечающих за соответствующий TLD, можно при помощи утилиты dig. Нас интересует тип записи NS, спрашивать мы будем резолвер 8.8.8.8.

```
$ dig com @8.8.8.8 ns

; <<>> DiG 9.10.3-P4-Ubuntu <<>> com @8.8.8.8 ns
;; global options: +cmd
;; Got answer:
;; ->>HEADER<<- opcode: QUERY, status: NOERROR, id: 12787
;; flags: qr rd ra ad; QUERY: 1, ANSWER: 13, AUTHORITY: 0, ADDITIONAL: 1

;; OPT PSEUDOSECTION:
```

```
; EDNS: version: 0, flags:; udp: 512
;; QUESTION SECTION:
;com.                           IN      NS

;; ANSWER SECTION: com.                    80329   IN      NS      b.gtld-
servers.net. com.                  80329   IN      NS      e.gtld-servers.net.
com.                       80329   IN      NS      k.gtld-servers.net. com.
80329   IN      NS      d.gtld-servers.net. com.                    80329
IN      NS      f.gtld-servers.net. com.                  80329   IN      NS
g.gtld-servers.net. com.                    80329   IN      NS      j.gtld-
servers.net. com.                  80329   IN      NS      l.gtld-servers.net.
com.                       80329   IN      NS      c.gtld-servers.net. com.
80329   IN      NS      i.gtld-servers.net. com.                    80329
IN      NS      m.gtld-servers.net. com.                  80329   IN      NS
h.gtld-servers.net. com.                    80329   IN      NS      a.gtld-
servers.net.


$ dig ru @8.8.8.8 ns

; <<>> DiG 9.10.3-P4-Ubuntu <<>> ru @8.8.8.8 ns ;; global
options: +cmd
;; Got answer:
;; ->>HEADER<<- opcode: QUERY, status: NOERROR, id: 53825
;; flags: qr rd ra ad; QUERY: 1, ANSWER: 5, AUTHORITY: 0, ADDITIONAL: 1

;; OPT PSEUDOSECTION:
; EDNS: version: 0, flags:; udp: 512
;; QUESTION SECTION:
;ru.                            IN      NS

;; ANSWER SECTION: ru.                       18382   IN      NS
a.dns.ripn.net. ru.                     18382   IN      NS
b.dns.ripn.net. ru.                     18382   IN      NS
d.dns.ripn.net. ru.                     18382   IN      NS
e.dns.ripn.net. ru.                     18382   IN      NS
f.dns.ripn.net.
```

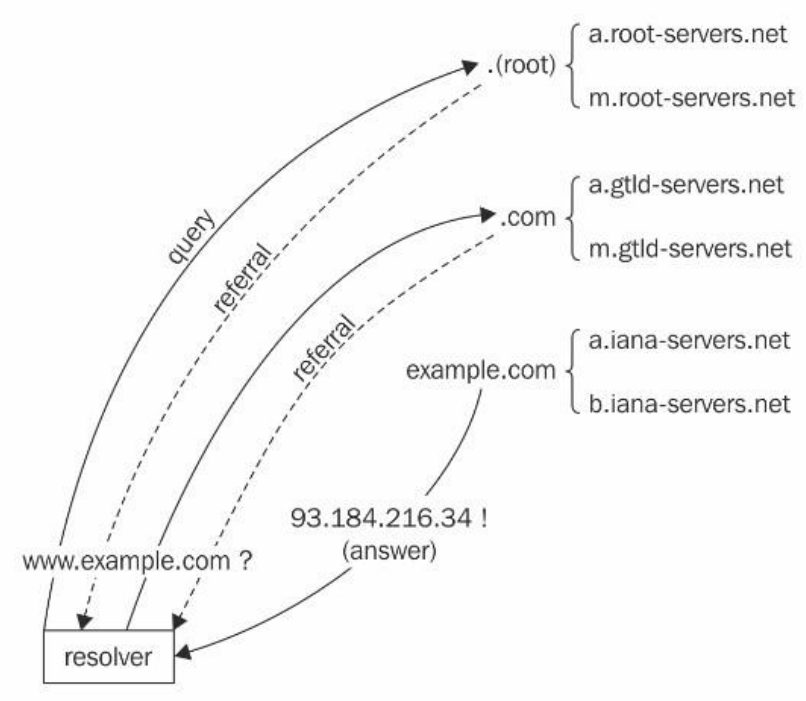
Также мы можем проверить список NS-серверов для новых доменов, например, домена .yandex или корневых DNS, отвечающих за точку.

```
$ dig yandex @8.8.8.8 ns

; <<>> DiG 9.10.3-P4-Ubuntu <<>> yandex @8.8.8.8 ns
;; global options: +cmd
;; Got answer:
;; ->>HEADER<<- opcode: QUERY, status: NOERROR, id: 25845
;; flags: qr rd ra ad; QUERY: 1, ANSWER: 6, AUTHORITY: 0, ADDITIONAL: 1
```

```
;; OPT PSEUDOSECTION:
; EDNS: version: 0, flags:; udp: 512
;; QUESTION SECTION:
;yandex.                                IN      NS

;; ANSWER SECTION: yandex.                  21599   IN      NS
ns2.dns.nic.yandex. yandex.                 21599   IN      NS
ns3.dns.nic.yandex. yandex.                 21599   IN      NS
ns5.dns.nic.yandex. yandex.                 21599   IN      NS
ns4.dns.nic.yandex. yandex.                 21599   IN      NS
ns6.dns.nic.yandex. yandex.                 21599   IN      NS
ns1.dns.nic.yandex.

$ dig . @8.8.8.8 ns

; <<>> DiG 9.10.3-P4-Ubuntu <<>> . @8.8.8.8 ns
;; global options: +cmd
;; Got answer:
;; ->>HEADER<<- opcode: QUERY, status: NOERROR, id: 34120
;; flags: qr rd ra ad; QUERY: 1, ANSWER: 13, AUTHORITY: 0, ADDITIONAL: 1

;; OPT PSEUDOSECTION:
; EDNS: version: 0, flags:; udp: 512
;; QUESTION SECTION:
;.                              IN      NS

;; ANSWER SECTION:
.                       226     IN      NS      m.root-servers.net.
.                       226     IN      NS      b.root-servers.net.
.                       226     IN      NS      c.root-servers.net.
.                       226     IN      NS      d.root-servers.net.
.                       226     IN      NS      e.root-servers.net.
.                       226     IN      NS      f.root-servers.net.
.                       226     IN      NS      g.root-servers.net.
.                       226     IN      NS      h.root-servers.net.
.                       226     IN      NS      a.root-servers.net.
.                       226     IN      NS      i.root-servers.net.
.                       226     IN      NS      j.root-servers.net.
.                       226     IN      NS      k.root-servers.net.
.                       226     IN      NS      l.root-servers.net.
```

Как видно из вывода, за зону отвечают более одного авторитетного DNS-сервера. Одна из причин — распределение нагрузки и повышение отказоустойчивости.

Посмотрим, как резолвер может узнать IP-адрес интересующего его доменного имени. Резолвер, если запись в кеше отсутствует, всегда начинает с вершины иерархии доменных имён — с точки.



Резолвер спрашивает у корневого DNS-сервера, какой IP-адрес у имени example.com. Такой информации у корневого DNS-сервера нет, так как он не является авторитетным DNS для example.com. Единственная информация, которая у него есть — данные о NS-серверах, отвечающих за зону .com. Поэтому вместо ответа на вопрос резолвера, корневой DNS выдаёт информацию о NS-серверах для зоны .com

Резолвер клиента обращается к DNS, отвечающим за зону .com. Эти DNS также не авторитетны для example.com, но они знают, кто отвечает за example.com и дают резолверу информацию о NS-серверах домена example.com.

В следующую итерацию резолвер обращается к авторитетным для example.com DNS-серверам, и они уже отвечают ему IP-адресом.

Тут встает вопрос, откуда резолвер знает, как взаимодействовать с корневыми DNS-серверами, если они ровно так же имеют доменное имя. Для решения проблемы курицы и яйца есть файл /var/named/named.ca:

```
$ cat /var/named/named.ca                                                    +norec

;    <<>>    DiG    9.11.3-RedHat-9.11.3-3.fc27    <<>>    +bufsize=1200
@a.root-servers.net
; (2 servers found)
;; global options: +cmd
;; Got answer:
;; ->>HEADER<<- opcode: QUERY, status: NOERROR, id: 46900
;; flags: qr aa; QUERY: 1, ANSWER: 13, AUTHORITY: 0, ADDITIONAL: 27
```

```
;; OPT PSEUDOSECTION:
; EDNS: version: 0, flags:; udp: 1472
;; QUESTION SECTION:
;.                              IN      NS

;; ANSWER SECTION:
.                       518400  IN      NS      a.root-servers.net.
.                       518400  IN      NS      b.root-servers.net.
.                       518400  IN      NS      c.root-servers.net.
.                       518400  IN      NS      d.root-servers.net.
.                       518400  IN      NS      e.root-servers.net.
.                       518400  IN      NS      f.root-servers.net.
.                       518400  IN      NS      g.root-servers.net.
.                       518400  IN      NS      h.root-servers.net.
.                       518400  IN      NS      i.root-servers.net.
.                       518400  IN      NS      j.root-servers.net.
.                       518400  IN      NS      k.root-servers.net.
.                       518400  IN      NS      l.root-servers.net.
.                       518400  IN      NS      m.root-servers.net.

;; ADDITIONAL SECTION:
a.root-servers.net.     518400  IN      A       198.41.0.4
b.root-servers.net.     518400  IN      A       199.9.14.201
c.root-servers.net.     518400  IN      A       192.33.4.12
d.root-servers.net.     518400  IN      A       199.7.91.13
e.root-servers.net.     518400  IN      A       192.203.230.10
f.root-servers.net.     518400  IN      A       192.5.5.241
g.root-servers.net.     518400  IN      A       192.112.36.4
h.root-servers.net.     518400  IN      A       198.97.190.53
i.root-servers.net.     518400  IN      A       192.36.148.17
j.root-servers.net.     518400  IN      A       192.58.128.30
k.root-servers.net.     518400  IN      A       193.0.14.129
l.root-servers.net.     518400  IN      A       199.7.83.42
m.root-servers.net.     518400  IN      A       202.12.27.33
a.root-servers.net.     518400  IN      AAAA    2001:503:ba3e::2:30
b.root-servers.net.     518400  IN      AAAA    2001:500:200::b
c.root-servers.net.     518400  IN      AAAA    2001:500:2::c
d.root-servers.net.     518400  IN      AAAA    2001:500:2d::d
e.root-servers.net.     518400  IN      AAAA    2001:500:a8::e
f.root-servers.net.     518400  IN      AAAA    2001:500:2f::f
g.root-servers.net.     518400  IN      AAAA    2001:500:12::d0d
h.root-servers.net.     518400  IN      AAAA    2001:500:1::53
i.root-servers.net.     518400  IN      AAAA    2001:7fe::53
j.root-servers.net.     518400  IN      AAAA    2001:503:c27::2:30
k.root-servers.net.     518400  IN      AAAA    2001:7fd::1
l.root-servers.net.     518400  IN      AAAA    2001:500:9f::42
m.root-servers.net.     518400  IN      AAAA    2001:dc3::35
```

Типы записей

Start of Authority (SoA)

Для каждой из зон должна существовать одна Resource Record (RR) типа SoA. В этой записи содержится базовая информация о зоне. Какие DNS-серверы авторитетны для этой зоны, как долго хранить запись в кеше и так далее. Синтаксис выглядит так:

```
<OWNER-NAME>  IN  SOA  <MNAME>  <RNAME>  <SERIAL>  <REFRESH>  <RETRY>  <EXPIRE>
<MINIMUM>
```

Либо в более привычной форме столбиком:

```
<OWNER-NAME IN SOA <MNAME> <RNAME> (
<SERIAL>
<REFRESH>
<RETRY>
<EXPIRE>
<MINIMUM>
)
```

- MNAME (Originating Nameserver) — предполагается, что это хостнейм мастера, который отвечает за эту зону.
- RNAME (Point of Contact) — выглядит как хостнейм, но им не является. Это почта администратора, который ответственен за эту зону. В имени не используется символ @ и он заменён на точку, так как @ — спецсимвол в файле зоны. Например, если почта администратора — admin@example.com, то RNAME будет выглядеть как admin.example.com.
- Serial — один из самых важных параметров. Каждый раз, когда вы изменяете файл зоны, вы должны увеличить серийный номер. Благодаря изменившемуся (в большую сторону) серийному номеру зоны, секондари-серверы понимают, что зона была изменена. Если вы используете DNS-сервер с бэкендом в виде базы данных, значение серийного номера перестаёт быть настолько важным. Само по себе значение серийного номера может быть любым. Неплохой вариант — использование unix timestamp для создания всегда увеличивающегося уникального значения.
- REFRESH — как долго секондари-сервер должен ждать перед тем, как проверить серийный номер мастера. Было придумано изначально, но сейчас для апдейта слейвов используется DNS NOTIFY, поэтому, значение REFRESH не важно.
- RETRY — какое количество времени слейв должен ждать перед тем, как спросить мастера о возможном апдейте зоны. С появлением DNS NOTIFY ситуация аналогична с REFRESH.
- EXPIRY — как долго авторитетный DNS-сервер должен продолжать отвечать на запросы клиентов, даже если он не может проверить обновлённую информацию у мастера. То есть, если

по какой-то причине мастер-сервер не работает, слейв будет продолжать отвечать на запросы клиентов. Это значение обычно достаточно велико и составляет несколько недель.

- MINIMUM — изначально этот параметр указывал резолверам, как долго следует хранить информацию в кеше. В дальнейшем это значение приняло другой смысл — как долго хранить информацию о негативном кеше (rfc 2308).

  Например, для сайта GeekBrains:

```
$ dig geekbrains.ru  -t soa

; <<>> DiG 9.11.4-P2-RedHat-9.11.4-9.P2.el7 <<>> geekbrains.ru -t soa ;; global
options: +cmd
;; Got answer:
;; ->>HEADER<<- opcode: QUERY, status: NOERROR, id: 37102
;; flags: qr rd ra; QUERY: 1, ANSWER: 1, AUTHORITY: 4, ADDITIONAL: 1

;; OPT PSEUDOSECTION:
; EDNS: version: 0, flags:; udp: 4096
;; QUESTION SECTION:
;geekbrains.ru.                 IN      SOA

;; ANSWER SECTION: geekbrains.ru.  900   IN    SOA   ns-1325.awsdns-37.org.
awsdns-hostmaster.amazon.com. 1 7200 900 1209600 900

;; AUTHORITY SECTION:
geekbrains.ru.          172375  IN      NS      ns-1325.awsdns-37.org.
geekbrains.ru.          172375  IN      NS      ns-1618.awsdns-10.co.uk.
geekbrains.ru.          172375  IN      NS      ns-230.awsdns-28.com.
geekbrains.ru.          172375  IN      NS      ns-989.awsdns-59.net.
```

Nameserver (NS)

NS RR — список авторитетных DNS-серверов (а точнее, хостнеймов DNS-серверов). Формат записи:

```
<OWNER-NAME> IN NS <nameserver hostname>
```

Например, для example.com:

```
$ORIGIN example.com.
IN NS a.iana-servers.net.
IN NS b.iana-servers.net.
```

Если NS-серверы находятся внутри зоны, за которую отвечают (в нашем случае — если бы для example.com NS-серверы были бы вида ns1.example.com), то обязательна соответствующая A-запись для хостнейма NS.

A/IPv4 Address

Наиболее часто используемая запись. Представляет собой соответствие доменного имени IP-адресу. Формат записи:

```
<OWNER-NAME> IN A <hostname>
```

OWNER-NAME — имя хоста внутри нашей зоны. Если имя не заканчивается на точку, к нему автоматически будет добавлено значение $ORIGIN для зоны. Одной из наиболее распространённых ошибок в настройке DNS — отсутствие в конце имени точки. Тогда к имени www.example.com добавляется значение $ORIGIN для зоны = example.com, что в итоге приводит к имени www.example.com.example.com. То есть:

```
$ORIGIN example.com. www IN A
192.168.1.1 www1.example.com. IN
A 192.168.1.1 www2.example.com IN
A 192.168.1.1
```

Рассмотрим все 3 записи:

- www не имеет в конце точки. Значит, к этому имени добавится значение $ORIGIN, что приведёт к записи www.example.com IN A 192.168.1.1. Запись валидна, и клиент получит IP-адрес 192.168.1.1, когда спросит у DNS-сервера, какой IP у доменного имени www.example.com.
- www1.example.com. имеет в конце точку. Значит, значение $ORIGIN не будет добавлено к этому имени, что приведёт к верному ответу на запрос клиента, как и в первом случае.
- www2.example.com не имеет в конце точки, а это значит, что запись приводит к имени www2.example.com.example.com. То есть хост, спросив у DNS-сервера, какой IP у www2.example.com, получит в ответ NXDOMAIN, так как такой записи нет.

CNAME/Alias

Запись была придумана как временное решение для переименования хостов, но, как обычно в интернете, всё временное выходит из-под контроля и используется не так, как было задумано. Формат записи:

```
<OWNER-NAME> IN CNAME <cname target>
```

CNAME, или Alias, или canonical name — ссылка (алиас) на другое имя. Например:

```
web.example.com. IN CNAME www.example.com.
```

Тут имя web.example.com указывает на www.example.com. То есть резолвер, отправив квери на web.example.com, получит в ответ CNAME, и ему придётся отправлять квери на www.example.com чтобы узнать IP-адрес ресурса.

Mail Exchanger (MX)

Указывает на Mail Transfer Agent (MTA), ответственный за приём почты для домена. Записей может быть несколько и они имеют значение приоритета. Например:

```
$ dig geekbrains.ru MX

; <<>> DiG 9.10.3-P4-Ubuntu <<>> geekbrains.ru MX
;; global options: +cmd
;; Got answer:
;; ->>HEADER<<- opcode: QUERY, status: NOERROR, id: 47849
;; flags: qr rd ra; QUERY: 1, ANSWER: 1, AUTHORITY: 4, ADDITIONAL: 1

;; OPT PSEUDOSECTION:
; EDNS: version: 0, flags:; udp: 4096
;; QUESTION SECTION:
;geekbrains.ru.                 IN      MX

;; ANSWER SECTION: geekbrains.ru.          300     IN      MX
10 emx.mail.ru.
```

В случае с GeekBrains, вся входящая почта будет отправляться на адрес emx.mail.ru. Чтобы узнать, на какой IP это сделать, приложению придётся отправить ещё один запрос, но для типа записи A:

```
$ dig emx.mail.ru

; <<>> DiG 9.10.3-P4-Ubuntu <<>> emx.mail.ru
;; global options: +cmd
;; Got answer:
;; ->>HEADER<<- opcode: QUERY, status: NOERROR, id: 23524
;; flags: qr rd ra; QUERY: 1, ANSWER: 2, AUTHORITY: 3, ADDITIONAL: 7

;; OPT PSEUDOSECTION:
; EDNS: version: 0, flags:; udp: 4096
;; QUESTION SECTION:
;emx.mail.ru.                   IN      A

;; ANSWER SECTION:
emx.mail.ru.            46      IN      A       94.100.180.180
emx.mail.ru.            46      IN      A       217.69.139.180
```

Значение приоритета используется для повышения отказоустойчивости. Почтовый сервер отправителя будет стараться отправить почту MX-серверу с самым высоким приоритетом (самое низкое значение). Если по какой-то причине сделать это не получилось, будет использован следующий сервер.

TXT/Text

TXT-запись содержит текстовую запись произвольного формата.

```
<OWNER-NAME> IN TXT <free form data>
```

Изначально в TXT-записи были комментарии, но сейчас TXT нужен для:

- Sender Policy Framework (SPF),
- Domain Keys (DKIM),
- DMARC.

Максимальная длина TXT-записи — 255 символов.

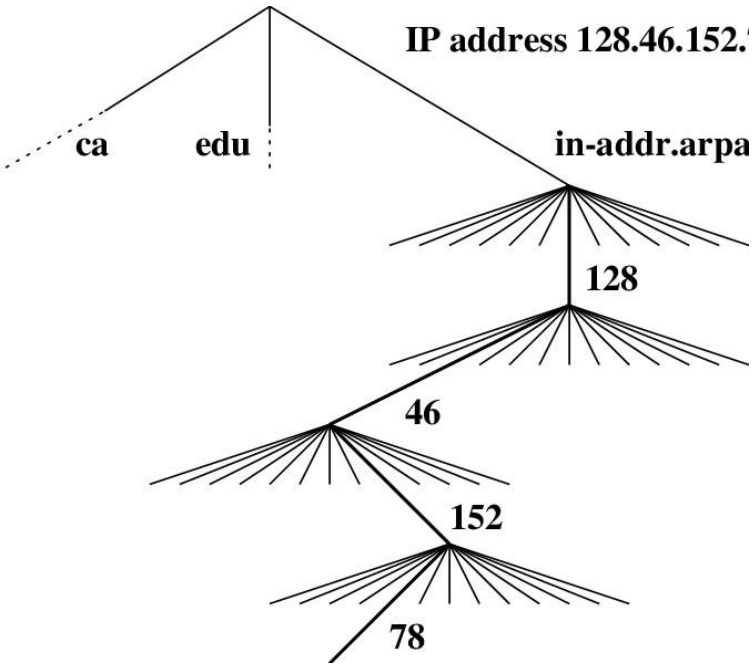
PTR

PTR-запись представляет собой обратную A-запись. То есть, если A-запись нужна для преобразования доменного имени в IP-адрес, PTR преобразует IP-адрес в доменное имя. Формат записи:

```
<OWNER-NAME> IN PTR <hostname>
```

В PTR записи OWNER-NAME — IP-адрес, написанный в обратном порядке. То есть, если у вас был IP-адрес 78.114.91.16, в обратной зоне он будет записан как 16.91.114.78.

Прямая и обратная зоны зачастую находятся на разных DNS-серверах, так как ресурсы принадлежат разным компаниям. Если доменное имя купили вы, то IP-адресом владеет, например, ваш интернет-провайдер. Следовательно, за резолв прямого запроса (имя -> IP) будет отвечать ваш DNS, а за обратный резолв (IP -> имя) — DNS провайдера, владельца IP.



Настройка BIND

Устанавливаем необходимые пакеты:

```
[root@centos7 ~]# yum install bind bind-utils
Dependencies Resolved

================================================================================
 Package                  Arch     Version                       Repository
                                                                          Size
================================================================================
Installing:
  bind                    x86_64   32:9.11.4-9.P2.el7          base   2.3 M
  bind-utils              x86_64   32:9.11.4-9.P2.el7          base   258 k
Installing for dependencies:
audit-libs-python        x86_64   2.8.5-4.el7                 base    76 k bind-
export-libs         x86_64   32:9.11.4-9.P2.el7          base   1.1 M bind-libs
x86_64   32:9.11.4-9.P2.el7         base   154 k checkpolicy             x86_64
2.5-8.el7                  base   295 k libcgroup               x86_64   0.41-
21.el7                base    66 k libsemanage-python       x86_64   2.5-14.el7
base   113 k policycoreutils-python   x86_64   2.5-33.el7                  base
457 k python-IPy               noarch   0.75-6.el7                  base    32
k python-ply                noarch   3.4-11.el7                  base   123 k
setools-libs             x86_64   3.3.8-4.el7                 base   620 k
Updating for dependencies:
  audit                    x86_64   2.8.5-4.el7                 base   256 k
  audit-libs               x86_64   2.8.5-4.el7                 base   102 k
  bind-libs-lite           x86_64   32:9.11.4-9.P2.el7          base   1.1 M
```

```
  bind-license              noarch   32:9.11.4-9.P2.el7          base    88 k
  dhclient                  x86_64   12:4.2.5-77.el7.centos      base   285 k
  dhcp-common               x86_64   12:4.2.5-77.el7.centos      base   176 k
  dhcp-libs                 x86_64   12:4.2.5-77.el7.centos      base   133 k
  policycoreutils           x86_64   2.5-33.el7                  base   916 k
```

```
Transaction Summary
================================================================================
```

Настраиваем демон named. В файлике /etc/named.conf можно создать списки доступа, указать, на какой из адресов открывать сокет и так далее. Если мы хотим ограничить хосты, которые могут отправлять нам квери, следует создать acl^

```
[root@centos7 ~]# vim /etc/named.conf
acl "trusted" {
        127.0.0.1;      # ms1
        192.168.1.41;   # ns2
        192.168.1.51;   # host1
        192.168.1.52;   # host2
};
```

Указываем, на каком адресе слушать сокет. В нашем случае — 127.0.0.1, IP-адрес интерфейса сервера — 192.168.1.33.

```
options { listen-on port 53 { 127.0.0.1;
192.168.1.33; }; #        listen-on-v6 port 53 { ::1;
};
```

Если мы хотим поддерживать трансфер зон между мастером и слейвом (указывая созданный нами acl), надо это явно разрешить. Также следует указать, кто может отправлять нам квери.

```
options { ... allow-transfer
{192.168.1.41; }; ... allow-query {
trusted; }; ...
```

В конце файла добавляем:

```
include "/etc/named/named.conf.local";
```

Теперь настраиваем /etc/named/named.conf.local. В этом файле мы указываем, какие будут зоны. Так как мы создаем файл с нуля, он будет пустым. Рассмотрим на примере домена example.com:

```
[root@centos7 ~]# vim /etc/named/named.conf.local zone
"example.com" { type master; file
"/etc/named/zones/db.example.com"; #файл конфига зоны };
```

Так как мы пользуемся локальной сетью, сама подсеть 192.168.1.0/24 принадлежит нам. А это значит, что PTR-зону на DMS должны создавать мы.

```
zone "1.168.192.in-addr.arpa" { type master; #файл конфига обратной зоны для
file "/etc/named/zones/db.1.168.192";
192.168.1.0/24
    };
```

Создаём файлы зон внутри /etc/named/zones/ директории:

```
[root@centos7 ~]# chmod 755 /etc/named
[root@centos7 ~]# mkdir /etc/named/zones
```

Добавляем записи в файл зон. Заполняем SOA и NS, A-записи:

```
[root@centos7 ~]# vim /etc/named/zones/db.example.com
$TTL    604800
@       IN      SOA     ns1.example.com. admin.example.com. (
          20210806      ; Serial
            604800      ; Refresh
             86400      ; Retry
           2419200      ; Expire
            604800 )    ; Negative Cache TTL
;
; name servers - NS records
     IN      NS      ns1.example.com.
     IN      NS      ns2.example.com.

; name servers - A records ns1.example.com.
IN      A       192.168.1.33 ns2.example.com.
IN      A       192.168.1.41

; 192.168.1.0/24 - A records
```

```
host1.example.com.        IN      A     192.168.1.51
host2.example.com.        IN      A     192.168.1.52
```

Настало время заполнить файл обратной зоны /etc/named/zones/db.1.168.192

```
[root@centos7 ~]# vim /etc/named/zones/db.1.168.192

$TTL    604800
@       IN      SOA     example.com. admin.example.com. (
                      20210806         ; Serial
                        604800         ; Refresh
                         86400         ; Retry
                       2419200         ; Expire
                        604800 )       ; Negative Cache TTL
; name servers
      IN      NS      ns1.example.com.
      IN      NS      ns2.example.com.

; PTR Records
33   IN      PTR     ns1.example.com.    ; 192.168.1.33
41   IN      PTR     ns2.example.com.    ; 192.168.1.41
51   IN      PTR     host1.example.com.  ; 192.168.1.51
52   IN      PTR     host2.example.com.  ; 192.168.1.52
```

Для проверки валидности синтаксиса конфига демона можно пользоваться утилитой named-checkconf.

```
[root@centos7 ~]# named-checkconf
```

Синтаксис зон проверяется другой утилитой — named-checkzone. Она требует ввода имени зоны и файла зоны. Например:

```
[root@centos7 ~]# named-checkzone example.com /etc/named/zones/db.example.com
[root@centos7 ~]# named-checkzone 1.168.192.in-addr.arpa
/etc/named/zones/db.1.168.192
```

После того, как проверили, что синтаксис валиден, не забываем включить сервис:

```
[root@centos7 ~]# systemctl enable named
[root@centos7 ~]# systemctl start named
```

Если потребуется, не забываем настроить файервол:

```
[root@centos7 ~]# firewall-cmd --zone=public --add-service=dns --
permanent
success
[root@centos7 ~]# firewall-cmd --zone=public --add-service=dns
```

Практическое задание

1. Настроить nic teaming между двумя интерфейсами — server1 и server2. Подсеть 192.168.12.0/24 будет находиться теперь на team0-интерфейсе.
2. На интерфейсе team0 сервера server2 назначить статический IP из подсети 192.168.12.0/24.
3. На сервере server2 настроить DHCP-сервер для выдачи динамического IP-адреса интерфейсу team0 сервера server1, а также IP-адрес DNS-сервера 3.3.3.3.
4. При помощи DHCP выдать серверу Server1 2 статических маршрута 4.4.4.4/32 и 5.5.5.0/24 с next hop интерфейса team0 на сервере server2
5. Настроить DNS-сервер для зоны example.com на сервере server3. Создать прямую и обратную зоны, а также несколько записей с разными RR. Убедиться, что только запросы на IP-адрес 3.3.3.3 будут обслуживаться этим DNS-сервером.
6. Настроить фаерволл на серверах server2 и server3, чтобы разрешить только соответствующие запросы (DHCP/DNS).
7. * Настроить slave для DNS-сервера server3. Убедиться, что репликация записей происходит.

Задачи со * предназначены для продвинутых учеников, которым мало обычного ДЗ.

Дополнительные материалы

1. Давайте уже разберёмся в DNS.
2. DNS-сервер BIND (теория).
3. DNSSec: что такое и зачем.

Используемая литература

Для подготовки данного методического пособия были использованы следующие ресурсы:

1. Product Documentation for Red Hat Enterprise Linux 7.
2. DNS and BIND, 5th Edition by Paul Albitz, Cricket Liu ISBN: 9780596100575.