Kubernetes DNS 入门介绍

狄卫华 2018.10.24

目录

1. DNS 基础

1. DNS 基础

- 将域名转换成 IP 地址

```
www.baidu.com   →    115.239.211.112
www.jd.com      →    2001:42d0::200:80:1
```

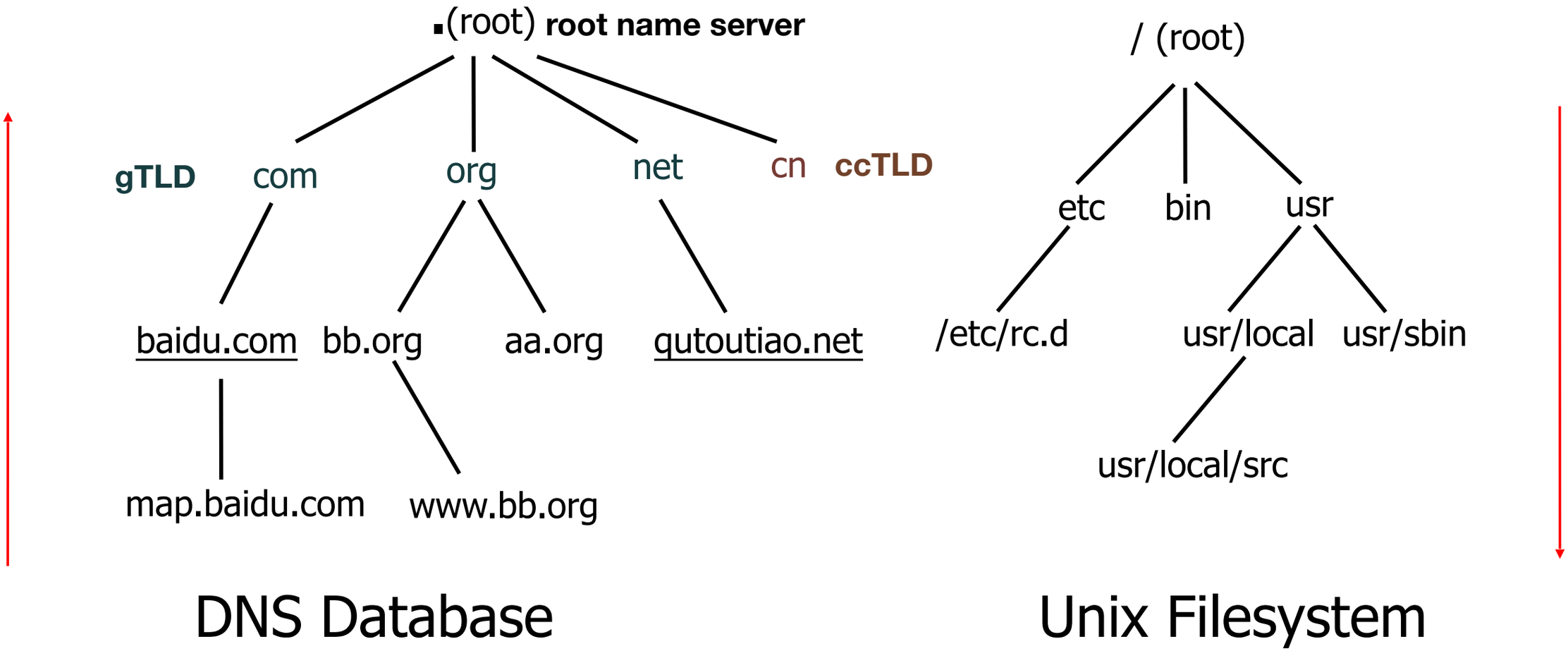
- 反向解析 IP:

```
112.211.239.115.in-addr.arpa.          →    www.baidu.com
1:80:200::42d0:2001.ip6.arpa.          →    www.jd.com
```

- 其他信息

域内如何发送邮件、谁负责维护系统、系统的地域信息等

1.1 DNS 结构


.(root) root name server
gTLD
com
org
net
cn
ccTLD
baidu.com
bb.org
aa.org
qutoutiao.net
map.baidu.com
www.bb.org
/ (root)
etc
bin
usr
/etc/rc.d
usr/local
usr/sbin
usr/local/src
DNS Database
Unix Filesystem

1.1 DNS 结构

- DNS 是分布式数据库
- 命名是层级结构的，查询过程从右往左，结尾包含一个不可见的 “.”
- 通过委托机制来进行查找

1.2 DNS 常见类型

- **A, AAAA:** IPv4, IPv6 address
- **NS:** NameServer 用于指定服务器
- **SOA** Start of Authority
- **MX:** Mail eXchanger
- **CNAME:** Canonical name (alias)
- **PTR:** Reverse information

1.3 DNS 工具

- 解析工具

    1. host name

    2. nslookup name server

    3. **dig** [options] name [@server]

> 默认采用 UDP 协议，如果没有办法查询到完整的信息时，就会再次的以 TCP 这个协定来重新查询

/etc/resolv.conf、/etc/hosts、/etc/nsswitch.conf

普通主机 resolv.conf

```
options timeout:2 attempts:3 rotate single-request-reopen
nameserver 100.100.2.138
nameserver 100.100.2.136
```

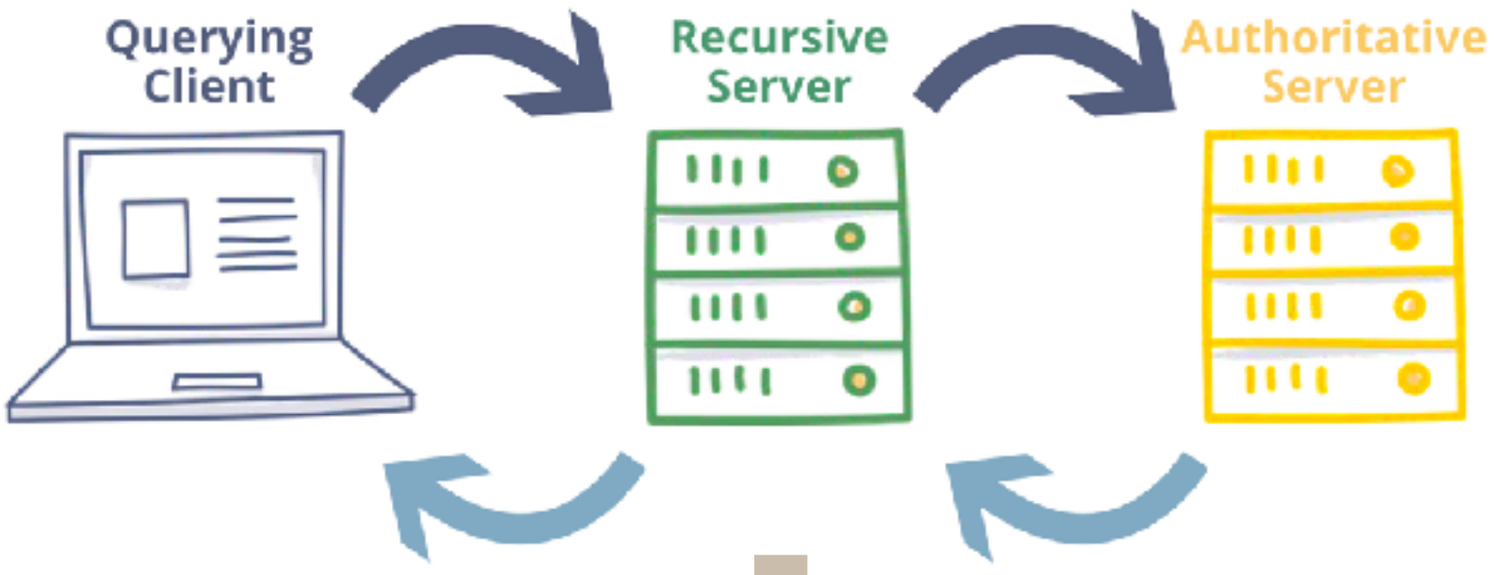
Pod ClusterFirst resolv.conf

```
nameserver 10.128.0.2
search default.svc.cluster.local svc.cluster.local cluster.local
options ndots:5
```

1.4 DNS 查询流程

Only knows root name servers .

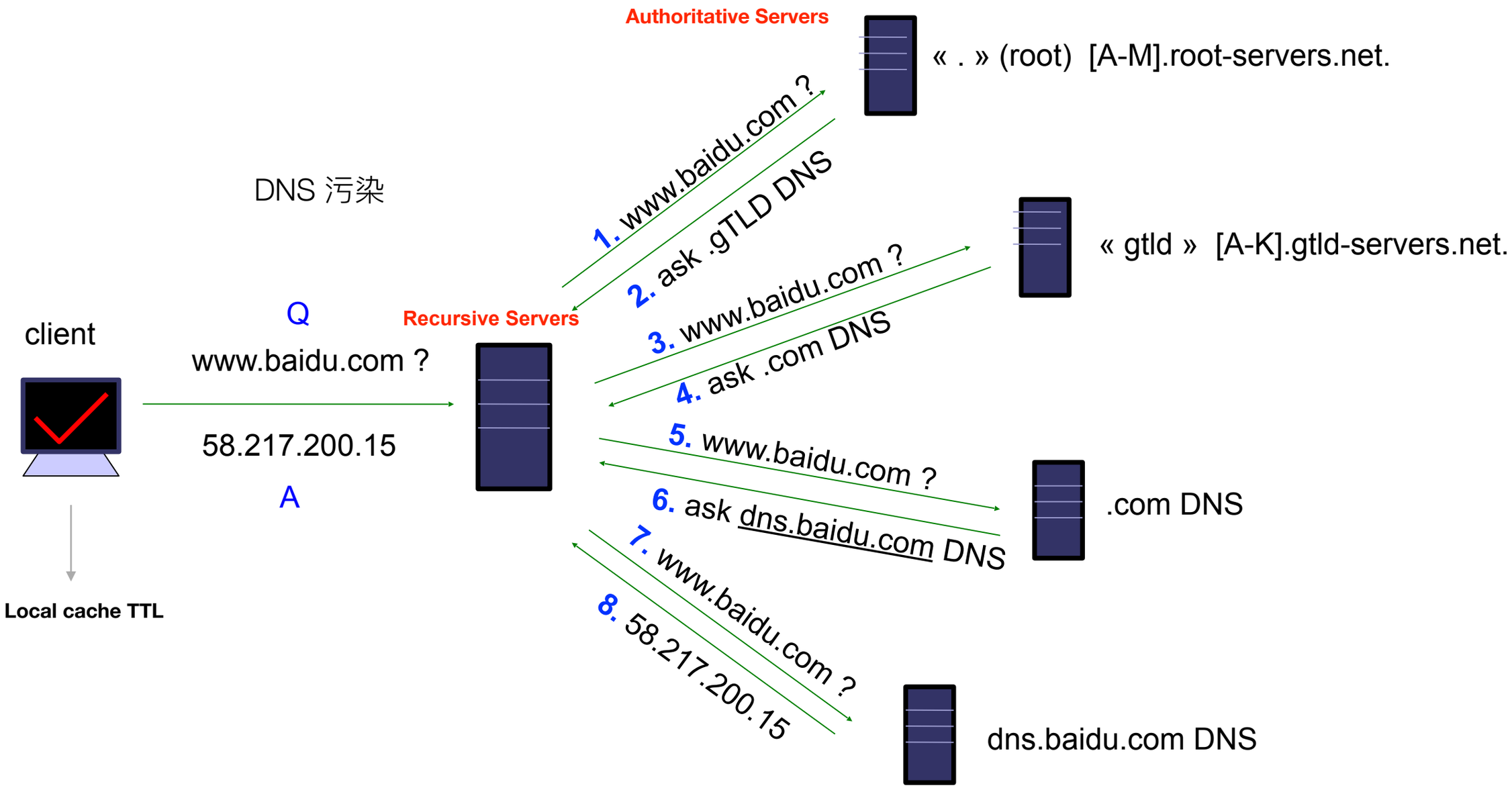
TTL time to live



DNS 中间人攻击
DNS 污染/劫持

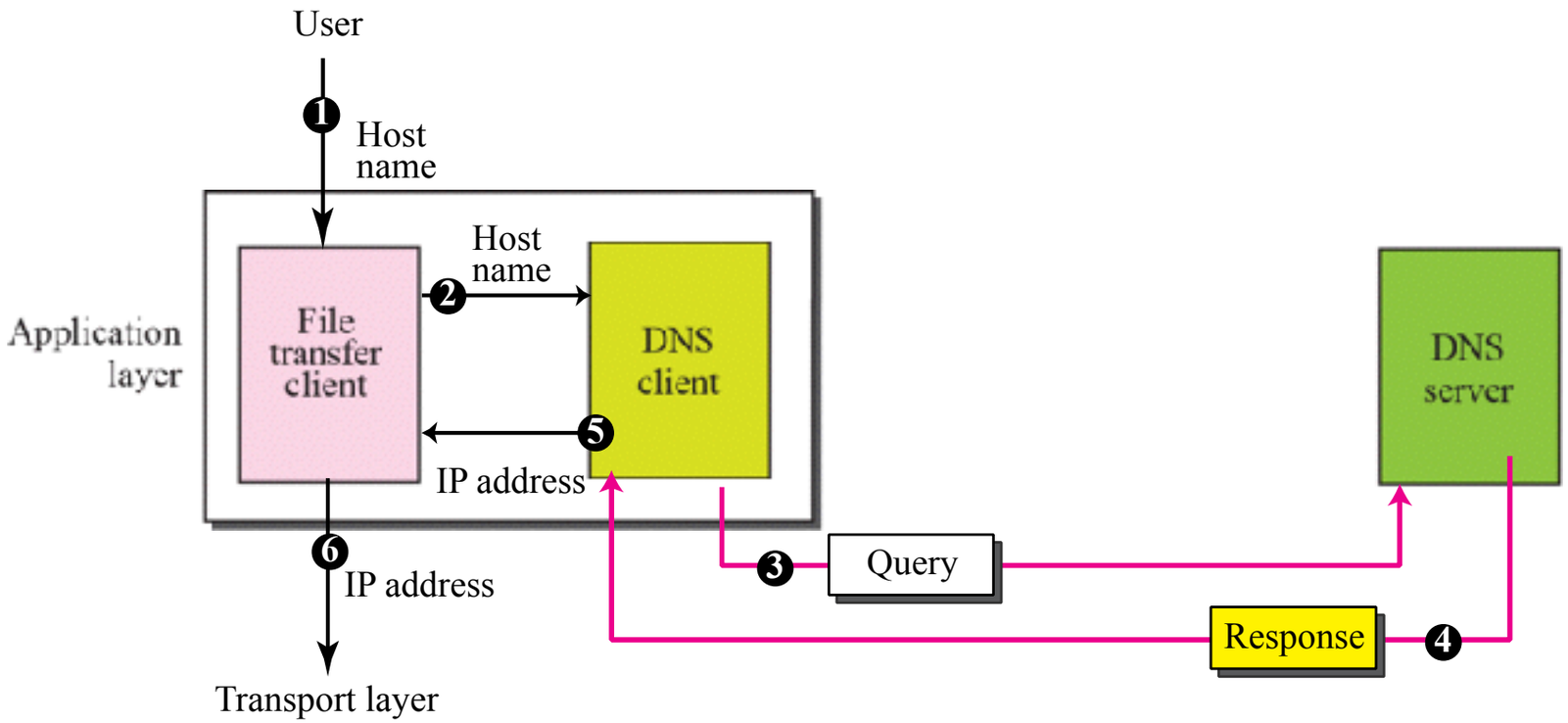
DNSCrypt

1.4 DNS 查询流程（递归查询模式）



- `dig +trace www.baidu.com`
- `tcpdump -n udp and port 53`

1.4 DNS 查询流程


User
1
Host name
Application layer
File transfer client
2
Host name
DNS client
5
IP address
6
IP address
Transport layer
DNS server
3
Query
Response
4

1.4 DNS 查询流程

$ dig +trace @8.8.8.8 www.baidu.com

```
; <<>> DiG 9.10.6 <<>> @8.8.8.8 www.baidu.com
; (1 server found)
;; global options: +cmd
;; Got answer:
;; ->>HEADER<<- opcode: QUERY, status: NOERROR, id: 21761
;; flags: qr rd ra; QUERY: 1, ANSWER: 3, AUTHORITY: 0, ADDITIONAL: 1

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

;; ANSWER SECTION:
www.baidu.com.          806     IN      CNAME   www.a.shifen.com.
www.a.shifen.com.       283     IN      A       58.217.200.15
www.a.shifen.com.       283     IN      A       58.217.200.13

;; Query time: 48 msec
;; SERVER: 8.8.8.8#53(8.8.8.8)
;; WHEN: Mon Oct 22 16:56:58 CST 2018
;; MSG SIZE  rcvd: 101
```

$ dig SOA www.baidu.com

```
; <<>> DiG 9.10.6 <<>> SOA www.baidu.com
;; global options: +cmd
;; Got answer:
;; ->>HEADER<<- opcode: QUERY, status: NOERROR, id: 43292
;; flags: qr rd ra; QUERY: 1, ANSWER: 1, AUTHORITY: 1, ADDITIONAL: 2

;; OPT PSEUDOSECTION:
; EDNS: version: 0, flags:; udp: 4000
;; QUESTION SECTION:
;www.baidu.com.                 IN      SOA

;; ANSWER SECTION:
www.baidu.com.          1010    IN      CNAME www.a.shifen.com.

;; AUTHORITY SECTION:
a.shifen.com.           39      IN      SOA     ns1.a.shifen.com. baidu_dns_master.baidu.com.
                                                1810220022  ; serial number of zone
                                                5           ; refresh check replica server
                                                5           ; retry     master server fails to answer after refresh
                                                2592000     ; expire    too long to answer client for this data.
                                                3600        ; neg ttl

;; ADDITIONAL SECTION:
ns1.a.shifen.com.300    IN      A       61.135.165.224

;; Query time: 23 msec
;; SERVER: 192.168.65.101#53(192.168.65.101)
;; WHEN: Mon Oct 22 17:05:40 CST 2018
;; MSG SIZE  rcvd: 142
```

2. Kubernetes DNS

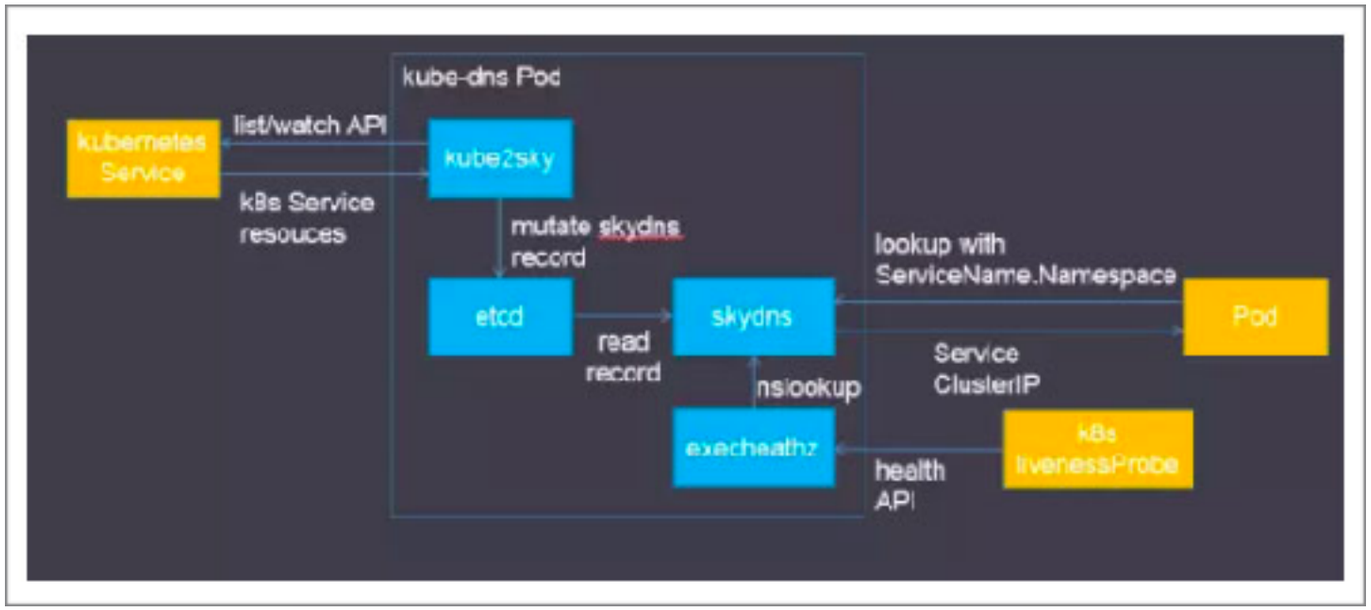
2.1 Kubernetes DNS 规范

- **Services**
  - A 记录
    - Normal my-svc.my-namespace.svc.cluster.loca -> cluster_ip
    - Headless my-svc.my-namespace.svc.cluster.local -> endpoints ip
  - SRV 记录： 命名端口 _my-port-name._my-port-protocol.my-svc.my-namespace.svc.cluster.local
- **Pods**
  - A 记录 **pod-ip-address.my-namespace.pod.cluster.local**

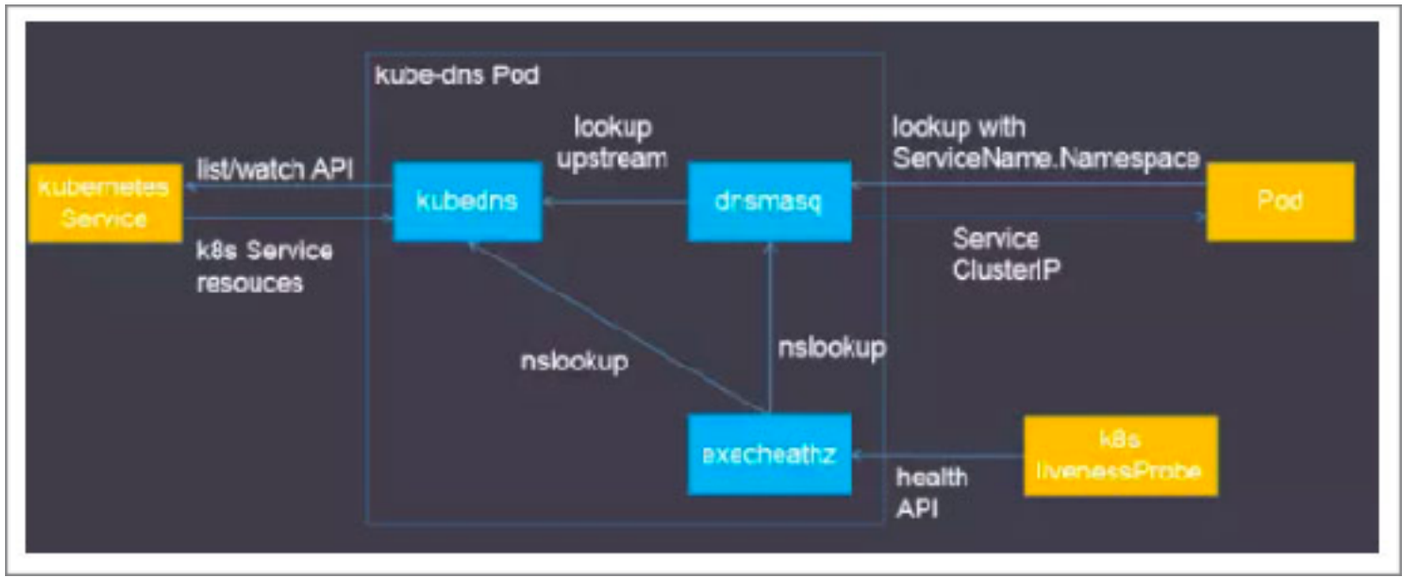
https://kubernetes.io/docs/concepts/services-networking/dns-pod-service/

2.2 Kubernetes DNS 迁移

kube-dns



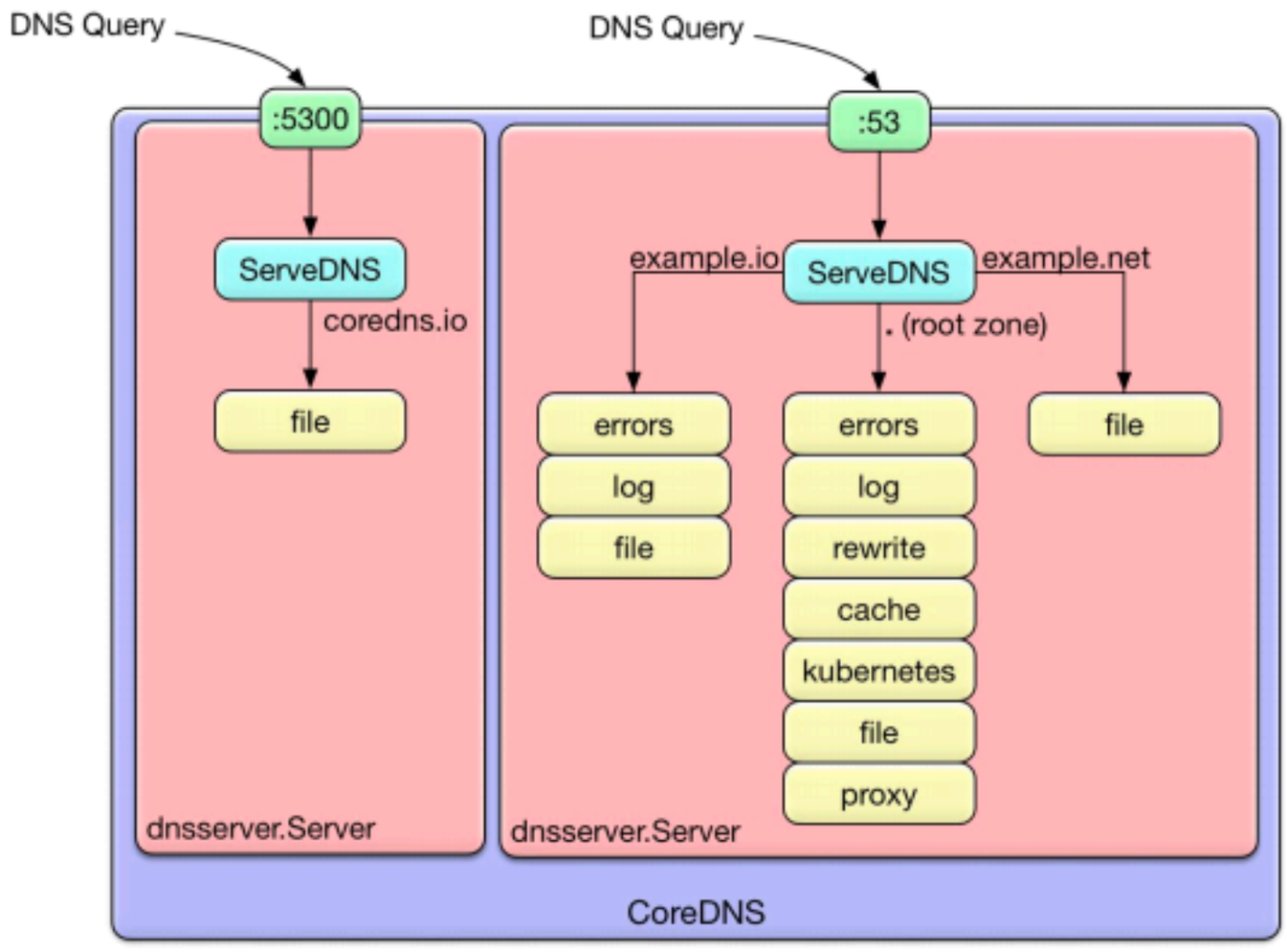
x < 1.4



1.4 < x < 1.11

Kubernetes DNS 迁移

CoreDNS



CoreDNS 是 CNCF 项目， 功能支持大多通过插件机制实现，包括基于 k8s 的服务发现，各种不同的插件通过链式来实现，在 k8s 1.11 版本版本中 GA，kubeadm 安装时候作为默认选项，目标是成为云原生的 DNS 服务器和服务发现机制。

2.3 Pod DNS

全域名 svc-name.default.svc.cluster.local

短域名 svc-name 或 svc-name.namespace

2.3 Pod DNS

- ## Default

  表示 Pod 里面的 DNS 配置继承了宿主机上的 DNS 配置。简单来说，就是该 Pod 的 DNS 配置会跟宿主机完全一致。

- ## ClusterFirst

  相对于上述的 Default，`ClusterFirst` 是完全相反的操作，它会预先把 `kube-dns`（或 CoreDNS）的信息当作预设参数写入到该 Pod 内的 DNS 配置。ClusterFirst 是预设的行为，若没有在 Pod 内特别描述 PodPolicy, 则会将 dnsPolicy 预设为 ClusterFirst。

  此外，ClusterFirst 还有一个冲突，如果你的 Pod 设置了 `HostNetwork=true`，则 ClusterFirst 就会被强制转换成 `Default`。

- ## ClusterFirstHostNetwork

  满足使用 HostNetwork 同时使用 `k8s DNS` 作为 Pod 预设 DNS 的配置。

- ## None

  清除 Pod 预设的 DNS 配置，当 dnsPolicy 设置成这个值之后，Kubernetes 不会为 Pod 预先载入任何自身逻辑判断得到的 DNS 配置。可通过 dnsConfig 来描述自定义的 DNS 参数， k8s 1.10 beta api-server 和 kubelet 需要启用 **—feature-gates=CustomPodDNS=true**

2.3 Pod DNS

- ClusterFirst 外部域名访问?

Pod ClusterFirst resolv.conf

```
nameserver 10.128.0.2
search default.svc.cluster.local svc.cluster.local cluster.local
options ndots:5
```

nslookup hello 10.128.0.2

Pod ClusterFirst resolv.conf

nameserver 10.128.0.2
search default.svc.cluster.local svc.cluster.local cluster.local
options ndots:5

```
[root@qdb-qtt-backend-skeleton-php-11 ~] eth0 = 10.0.50.237
# tcpdump port 53 -i flannel.1 -nn
tcpdump: verbose output suppressed, use -v or -vv for full protocol decode
listening on flannel.1, link-type EN10MB (Ethernet), capture size 262144 bytes
09:50:10.729310 IP 10.64.48.0.54251 > 10.64.3.10.53: 5106+ PTR? 2.0.128.10.in-addr.arpa. (41)
09:50:10.729765 IP 10.64.3.10.53 > 10.64.48.0.54251: 5106* 1/0/0 PTR coredns1.kube-system.svc.cluster.local.
09:50:10.730222 IP 10.64.48.0.33333 > 10.64.3.10.53: 23975+ A? hello.default.svc.cluster.local. (49)
09:50:10.730284 IP 10.64.48.0.33333 > 10.64.3.10.53: 24363+ AAAA? hello.default.svc.cluster.local. (49)
09:50:10.730511 IP 10.64.3.10.53 > 10.64.48.0.33333: 24363 NXDomain* 0/1/0 (142)
09:50:10.730616 IP 10.64.3.10.53 > 10.64.48.0.33333: 23975 NXDomain* 0/1/0 (142)
09:50:10.730850 IP 10.64.48.0.52835 > 10.64.3.10.53: 36407+ A? hello.svc.cluster.local. (41)
09:50:10.730869 IP 10.64.48.0.52835 > 10.64.3.10.53: 36808+ AAAA? hello.svc.cluster.local. (41)
09:50:10.731039 IP 10.64.3.10.53 > 10.64.48.0.52835: 36407 NXDomain* 0/1/0 (134)
09:50:10.731109 IP 10.64.3.10.53 > 10.64.48.0.52835: 36808 NXDomain* 0/1/0 (134)
09:50:10.731343 IP 10.64.48.0.36886 > 10.64.3.10.53: 5684+ A? hello.cluster.local. (37)
09:50:10.731362 IP 10.64.48.0.36886 > 10.64.3.10.53: 5908+ AAAA? hello.cluster.local. (37)
09:50:10.731545 IP 10.64.3.10.53 > 10.64.48.0.36886: 5908 NXDomain* 0/1/0 (130)
09:50:10.731649 IP 10.64.3.10.53 > 10.64.48.0.36886: 5684 NXDomain* 0/1/0 (130)
09:50:10.731882 IP 10.64.48.0.48411 > 10.64.3.10.53: 22989+ A? hello. (23)
09:50:10.731901 IP 10.64.48.0.48411 > 10.64.3.10.53: 23211+ AAAA? hello. (23)
09:50:10.732582 IP 10.64.3.10.53 > 10.64.48.0.48411: 22989 NXDomain 0/1/0 (98)
```

0 1 2 3 4

nslookup hello.1.2.3.4.5 10.128.0.2

```
09:59:56.476589 IP 10.64.48.0.45762 > 10.64.3.10.53: 62778+ PTR? 2.0.128.10.in-addr.arpa. (41)
09:59:56.476971 IP 10.64.3.10.53 > 10.64.48.0.45762: 62778* 1/0/0 PTR coredns1.kube-system.svc.cluster.local. (116)
09:59:56.477423 IP 10.64.48.0.47455 > 10.64.3.10.53: 55009+ A? hello.1.2.3.4.5. (33)
09:59:56.477443 IP 10.64.48.0.47455 > 10.64.3.10.53: 55409+ AAAA? hello.1.2.3.4.5. (33)
09:59:56.493805 IP 10.64.3.10.53 > 10.64.48.0.47455: 55409 NXDomain 0/1/0 (108)
09:59:56.503722 IP 10.64.3.10.53 > 10.64.48.0.47455: 55009 NXDomain 0/1/0 (108)
```

1

nslookup hello. 10.128.0.2

Pod ClusterFirst resolv.conf

nameserver 10.128.0.2
search default.svc.cluster.local svc.cluster.local cluster.local
options ndots:5

```
10:04:18.412147 IP 10.64.48.0.42117 > 10.64.3.10.53: 16908+ PTR? 2.0.128.10.in-addr.arpa. (41)
10:04:18.412608 IP 10.64.3.10.53 > 10.64.48.0.42117: 16908* 1/0/0 PTR coredns1.kube-system.svc.cluster.local. (116)
10:04:18.413081 IP 10.64.48.0.36333 > 10.64.3.10.53: 15224+ A? hello. (23)
10:04:18.413139 IP 10.64.48.0.36333 > 10.64.3.10.53: 15544+ AAAA? hello. (23)
10:04:18.431659 IP 10.64.3.10.53 > 10.64.48.0.36333: 15544 NXDomain 0/1/0 (98)
10:04:18.441347 IP 10.64.3.10.53 > 10.64.48.0.36333: 15224 NXDomain 0/1/0 (98)
```

1

总结：

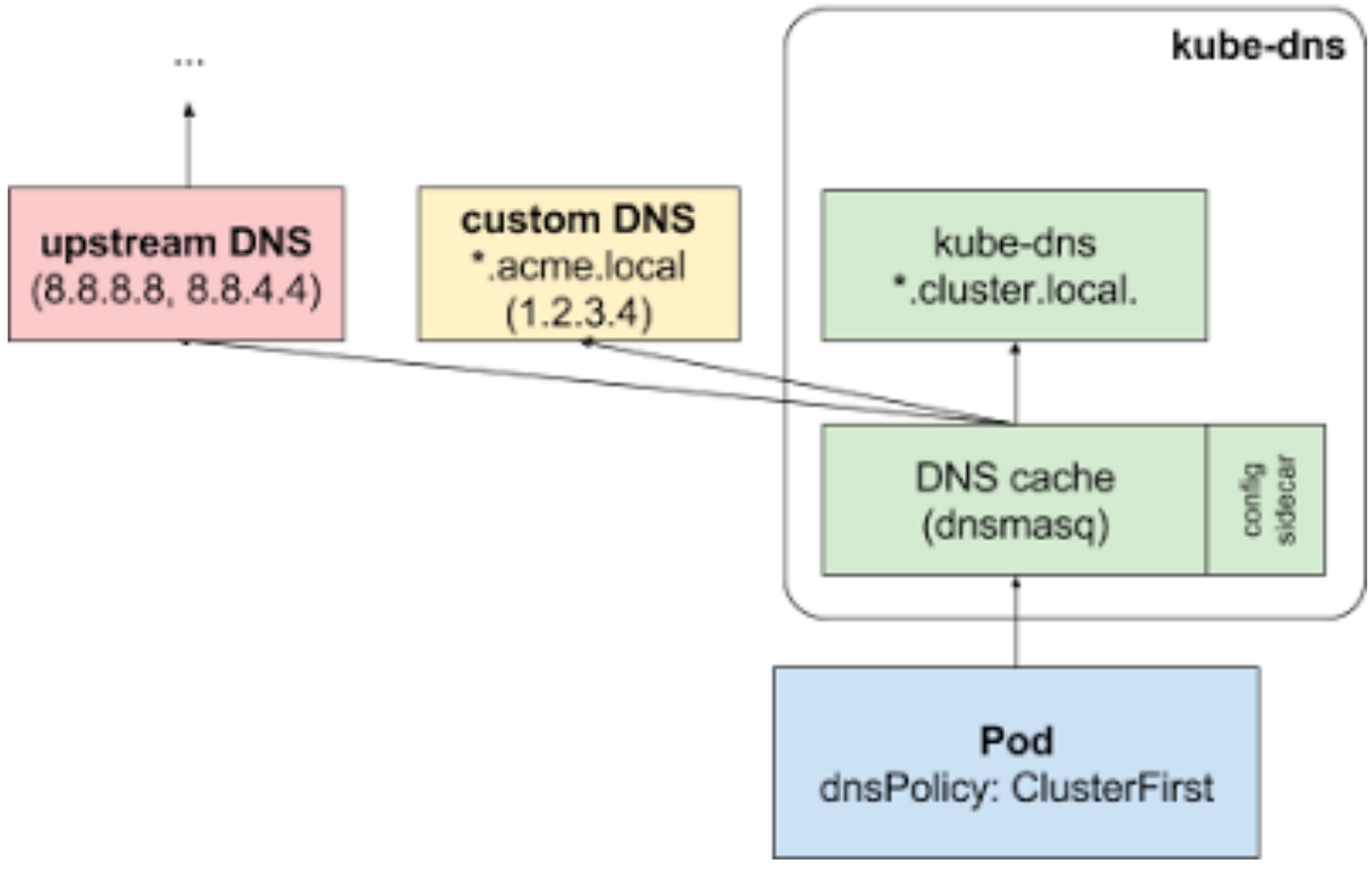
1. 对于 FQDN . 不通过 search 进行搜索
2. options ndots:5，域名包含大于等于 5，不进行 search
3. 其他情况使用 search 循环

```yaml
apiVersion: v1
kind: Pod
metadata:
  namespace: default
  name: dns-example
spec:
  containers:
    - name: test
      image: nginx
  dnsConfig:
    options:
      - name: ndots
        value: "1"
```

备注：如果使用 “curl -v http://www.baidu.com.”，会在 http header 中将 “Host: www.baidu.com. ” 导致不能访问

FQDN = Fully Qualified Domain Name = host-name + domain-name + “.”

2.3 Pod DNS


...
kube-dns
upstream DNS
(8.8.8.8, 8.8.4.4)
custom DNS
*.acme.local
(1.2.3.4)
kube-dns
*.cluster.local.
DNS cache
(dnsmasq)
config
sidecar
Pod
dnsPolicy: ClusterFirst

2.3 Pod DNS

- IPv6 查询如无必要可以禁用
- 外部域名减少试错

Linux 的 libc 不可思议的卡住（查看该2005年起暴出来的bug）限制只能有 3 个 DNS `nameserver` 记录和 6 个 DNS `search` 记录。Kubernetes 需要消耗 1 个 `nameserver` 记录和 3 个 `search` 记录。这意味着如果本地安装已经使用 3 个 `nameserver` 或使用 3 个以上的 `search` 记录，那么其中一些设置将会丢失。有个部分解决该问题的方法，就是节点可以运行 `dnsmasq`，它将提供更多的 `nameserver` 条目，但不会有更多的 `search` 条目。您也可以使用 kubelet 的 `--resolv-conf` 标志。

CoreDNS 配置

```
apiVersion: v1
data:
  Corefile: |
    .:53 {
        errors
        health
        kubernetes cluster.local. in-addr.arpa ip6.arpa {
          pods insecure
          upstream   # CNAME 的解析
          fallthrough in-addr.arpa ip6.arpa  # 透传
        }

        prometheus :9153
        proxy . /etc/resolv.conf
        cache 30
        loop
        reload
        loadbalance
    }
kind: ConfigMap
```

https://github.com/coredns/coredns/tree/master/plugin/kubernetes

配置顺序?

autopath @kubernets

2.4 DNS 排查过程

1. DNS Pod 是否运行正常
2. Pod 是否有报错
3. **查看 dns /etc/resolv.conf ***
4. Pod 内部解析是否成功
5. 使用 FQDN 解析是否成功

Thank You!