Rethinking the K8s DNS for the Modern Enterprise

KubeCon NA 2019

Deepa Kalani
Staff Engineer 2
NSX Service Mesh
VMware
dkalani@vmware.com

Venil Noronha
Member of Technical Staff
NSX Service Mesh
VMware
veniln@vmware.com

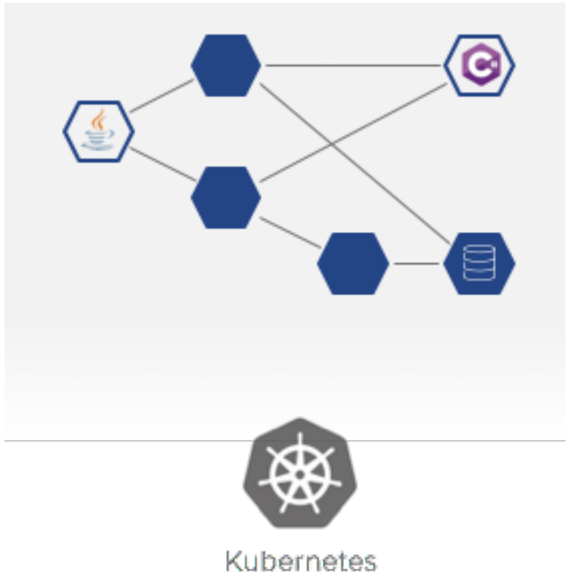
Service Mesh Capabilities

Identity

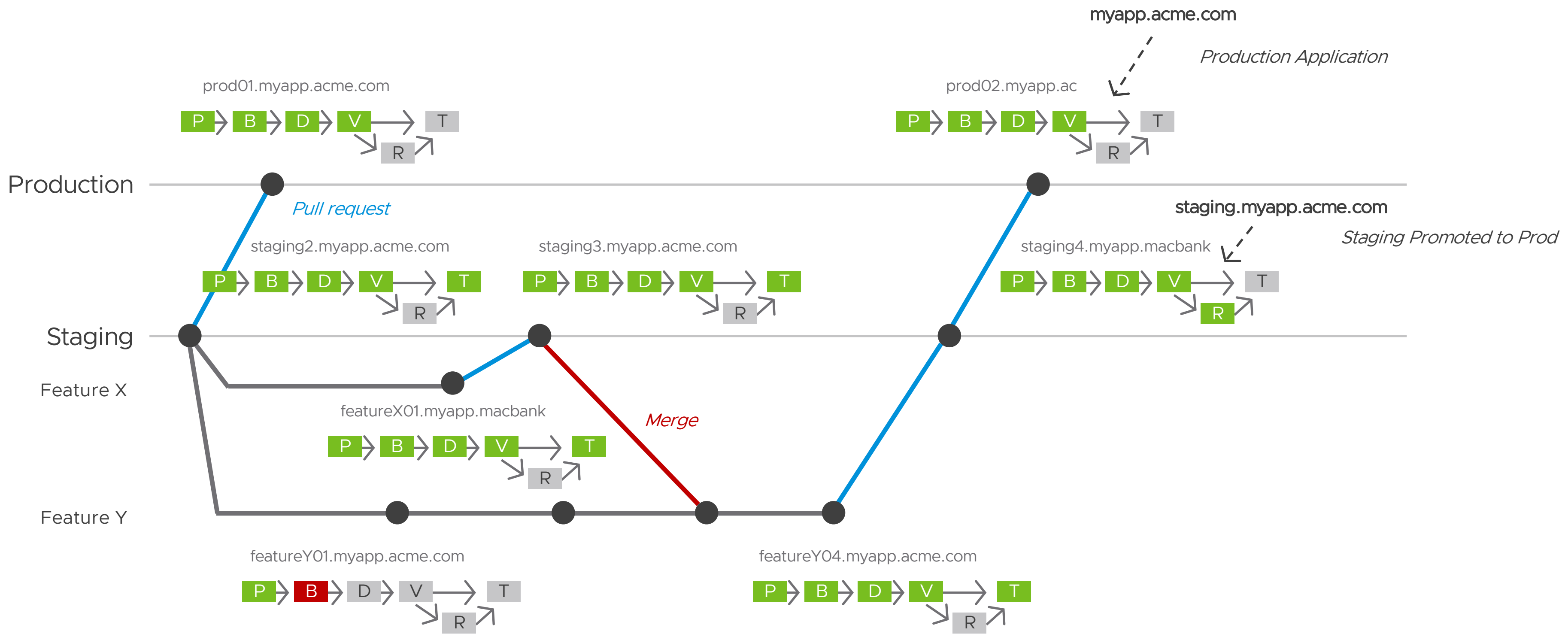
Service Discovery/DNS

Routing

Policy

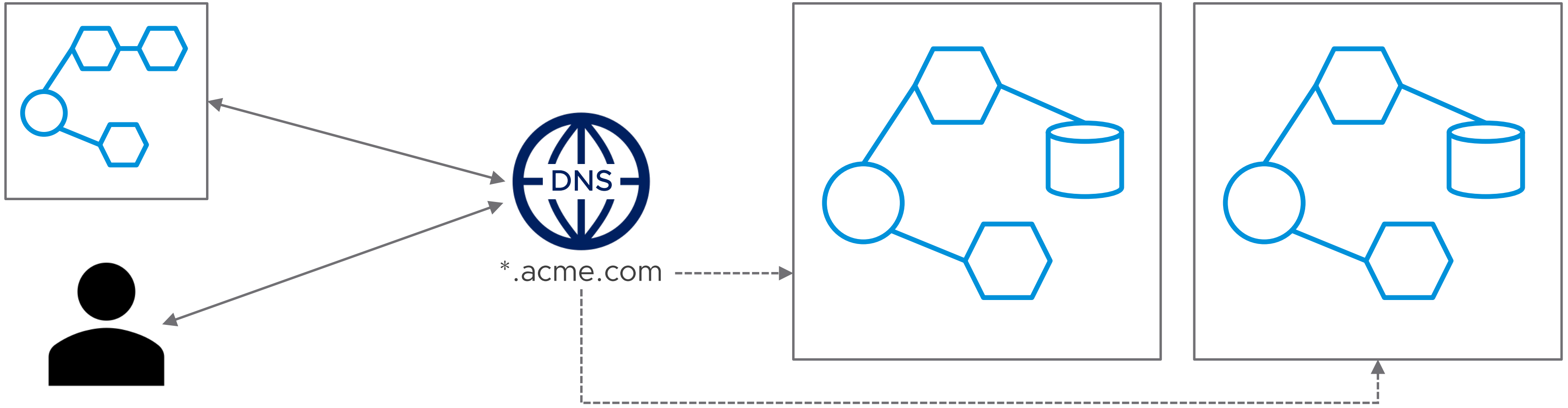


Names are complicated…

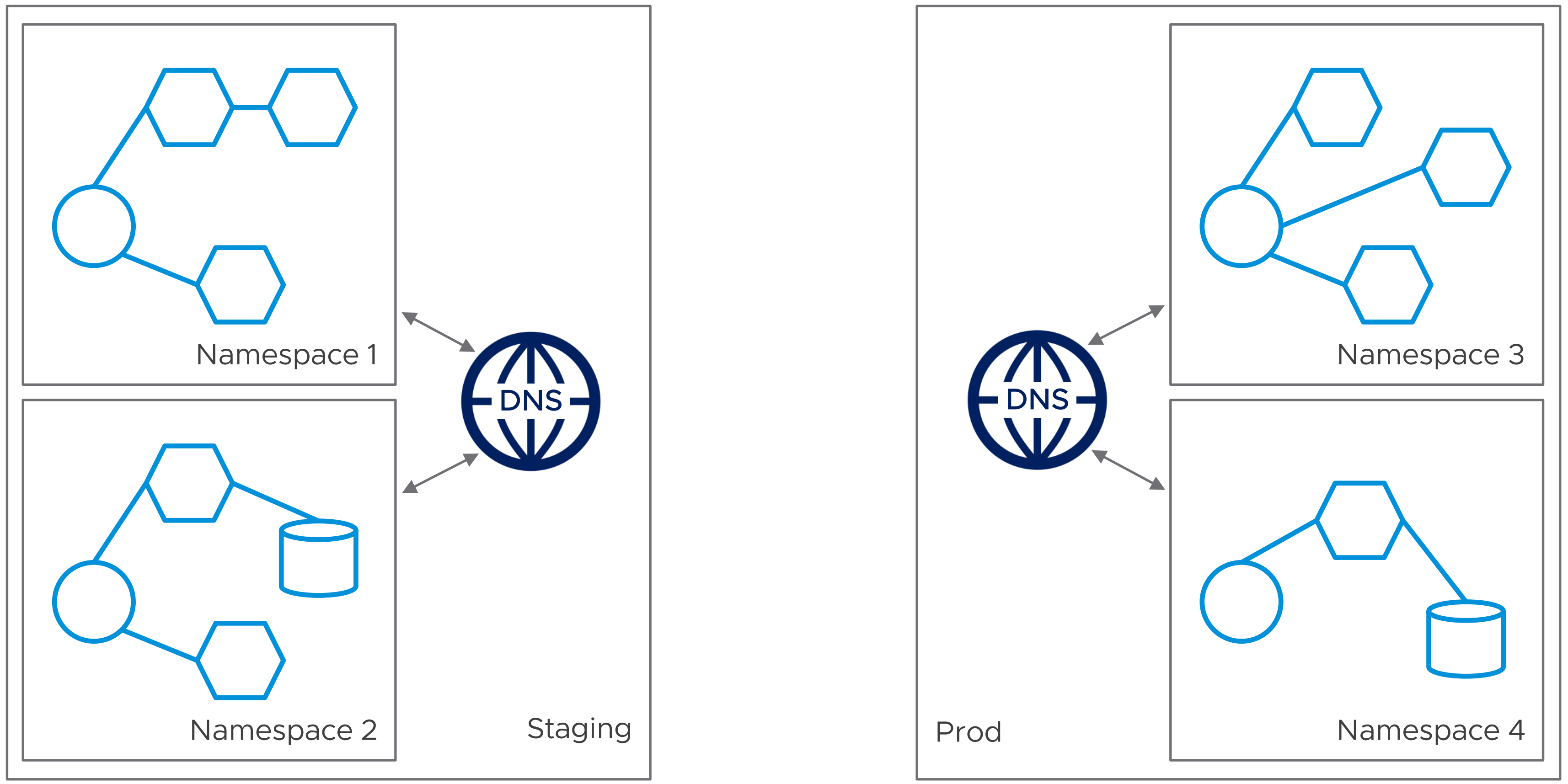


Application Migration – Simplification through Naming

- Multi-cloud and hybrid-cloud systems
- In a multi-cloud world, applications may be deployed on prem and in the cloud
- Developers should be able to deploy and migrate applications across any cloud provider without changing their native workloads

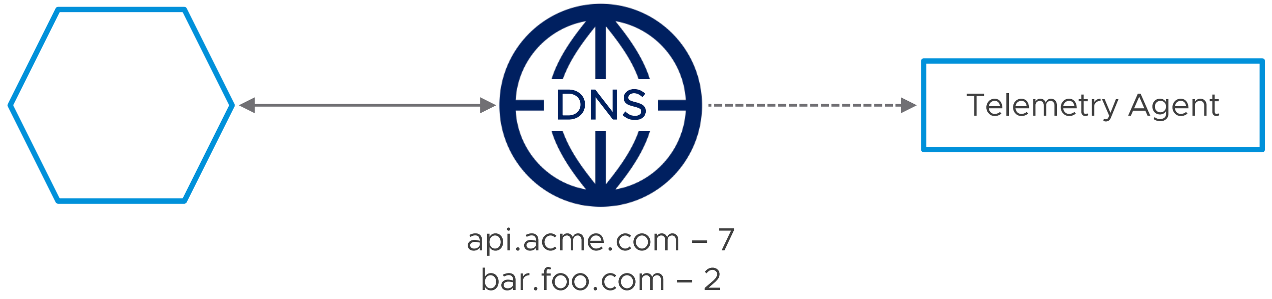


DNS Isolation – Enabler for Multi-tenant Clusters



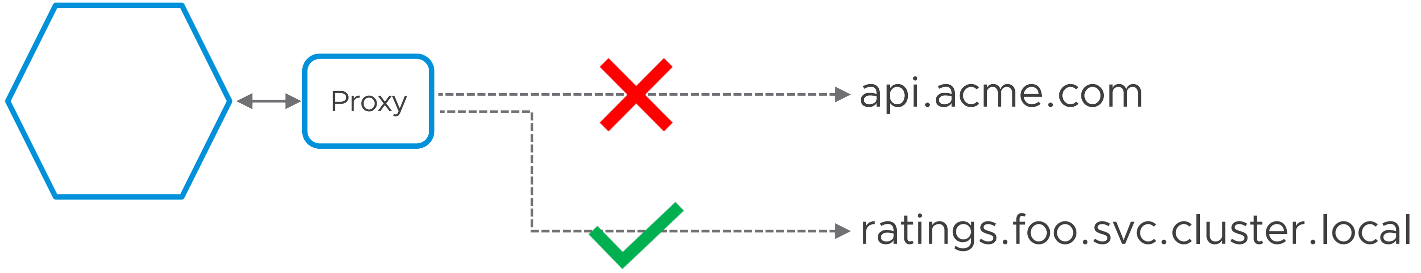
DNS Observability and Security

- Rich telemetry for DNS queries and responses
- Telemetry per tenant
- Open the door to behavioral analysis based on telemetry data.



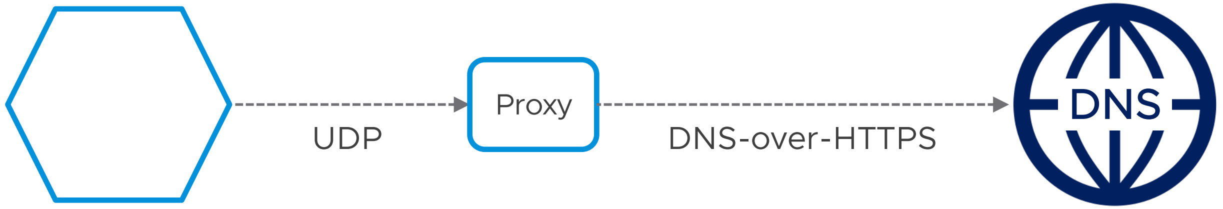
DNS Filtering

- Operators need a way to specify filtering the DNS layer
- DNS policies allow for access control and logging
- Example:
  - *Deny the frontend service from discovering *.com and log such requests*
  - *TenantA services should not discover tenantb.services*
- Treat DNS just as another entity in the Kubernetes cluster
- Apply L4/L7 policies based on DNS queries/responses



DNS Evolution

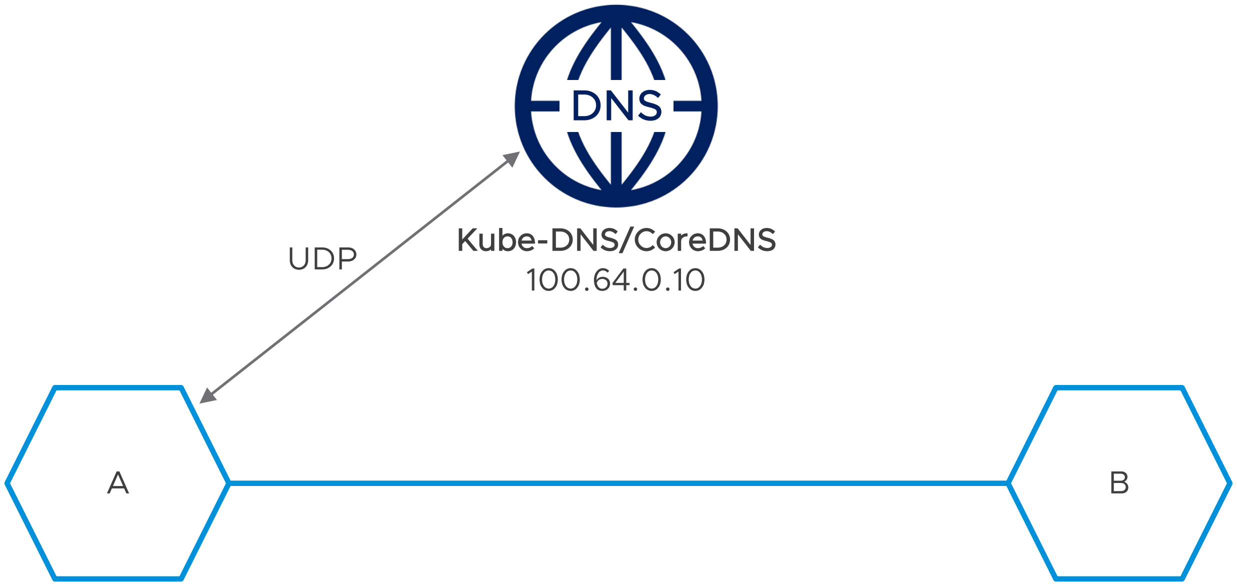
- Some tenants might want to encrypt DNS queries to maintain privacy
- Imperative in a multi-tenant environment
- Upgrade UDP/TCP DNS queries to DoT/DoH



Current State of Kubernetes DNS

- No tenant isolation for DNS
- No dynamic configuration of DNS
  - Can’t configure search domains dynamically
  - Can’t configure nameservers dynamically
- Policies cannot be enforced at the DNS layer
- Doesn’t provide first-class support for secure DNS
  - DNS-over-TLS (DoT)
  - DNS-over-HTTPS (DoH)

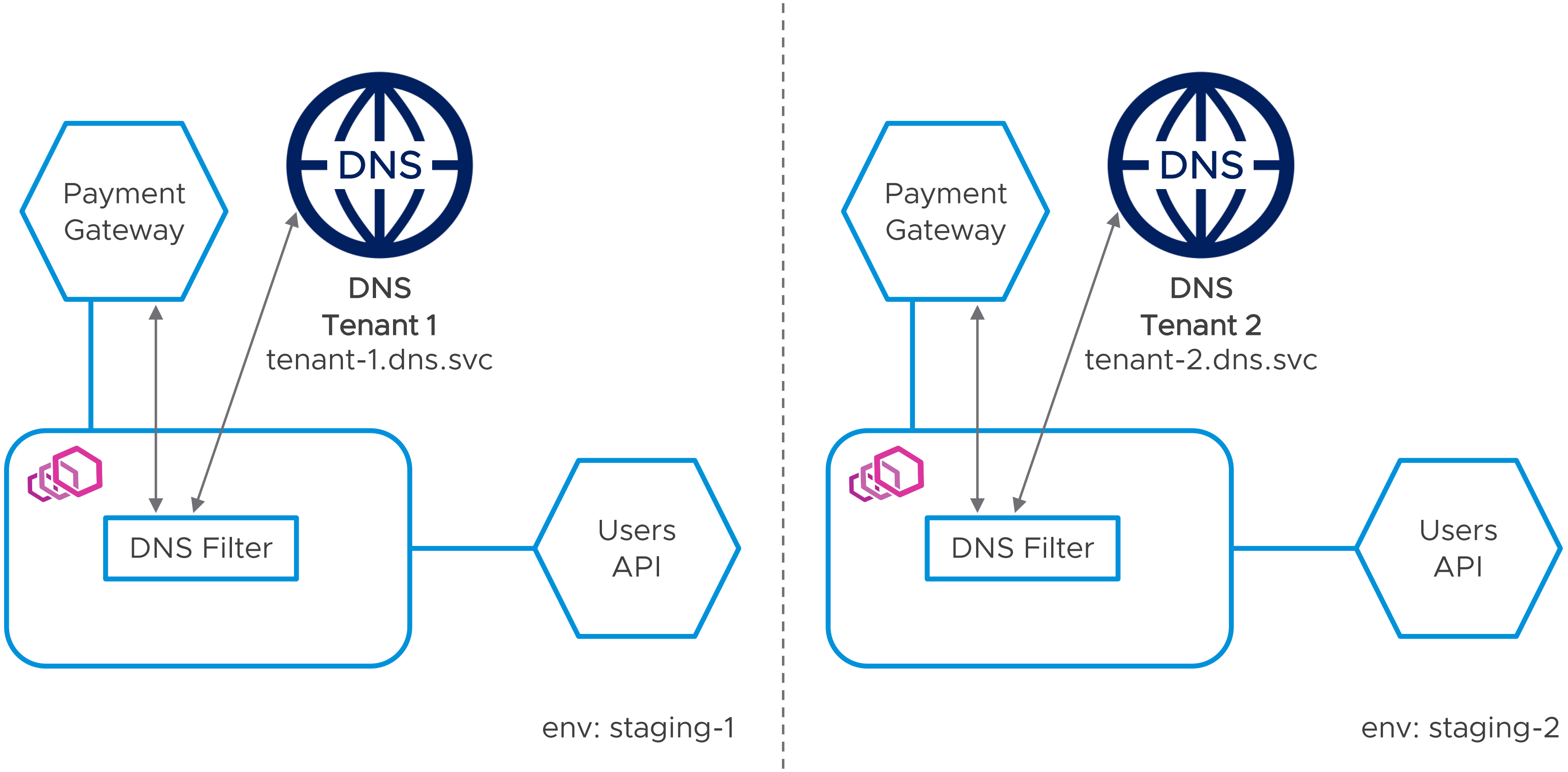
Plain Old Kubernetes DNS



DNS Isolation – Enabler for Multi-tenant Clusters

```yaml
apiVersion: "networking.example.com/v1alpha1"
kind: DnsPolicy
metadata:
  name: dns-policy
spec:
  namespaceSelector:
    matchLabels:
      env: staging-1
  server:
    # The DNS server address for the tenant.
    address: tenant-1.dns.svc
```

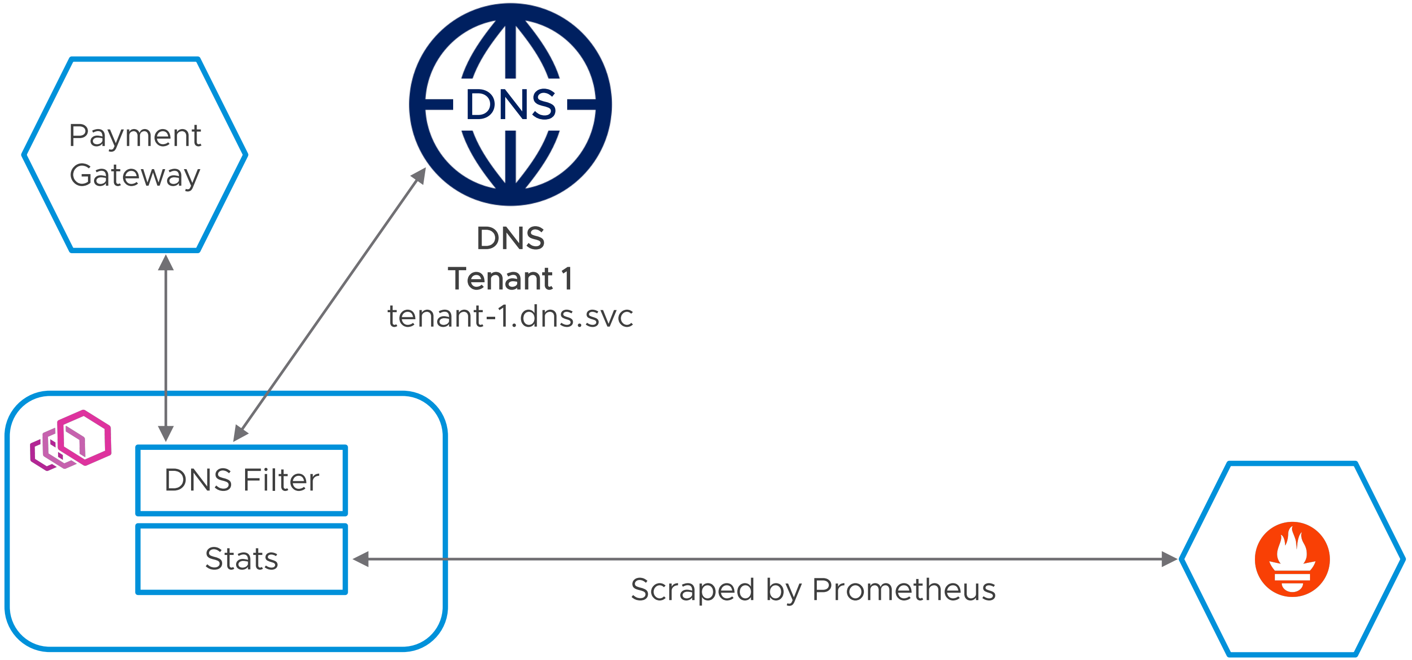
DNS Isolation – Enabler for Multi-tenant Clusters



DNS Observability and Security

```yaml
apiVersion: monitoring.coreos.com/v1
kind: ServiceMonitor
metadata:
  name: tenant-monitor
spec:
  selector:
    matchLabels:
      app: payment-gateway # The service label.
  namespaceSelector:
    matchNames:
      - payments # The service namespace.
  endpoints:
  - targetPort: 8000 # The Envoy stats port.
    path: /stats/prometheus # The Envoy stats endpoint.
```

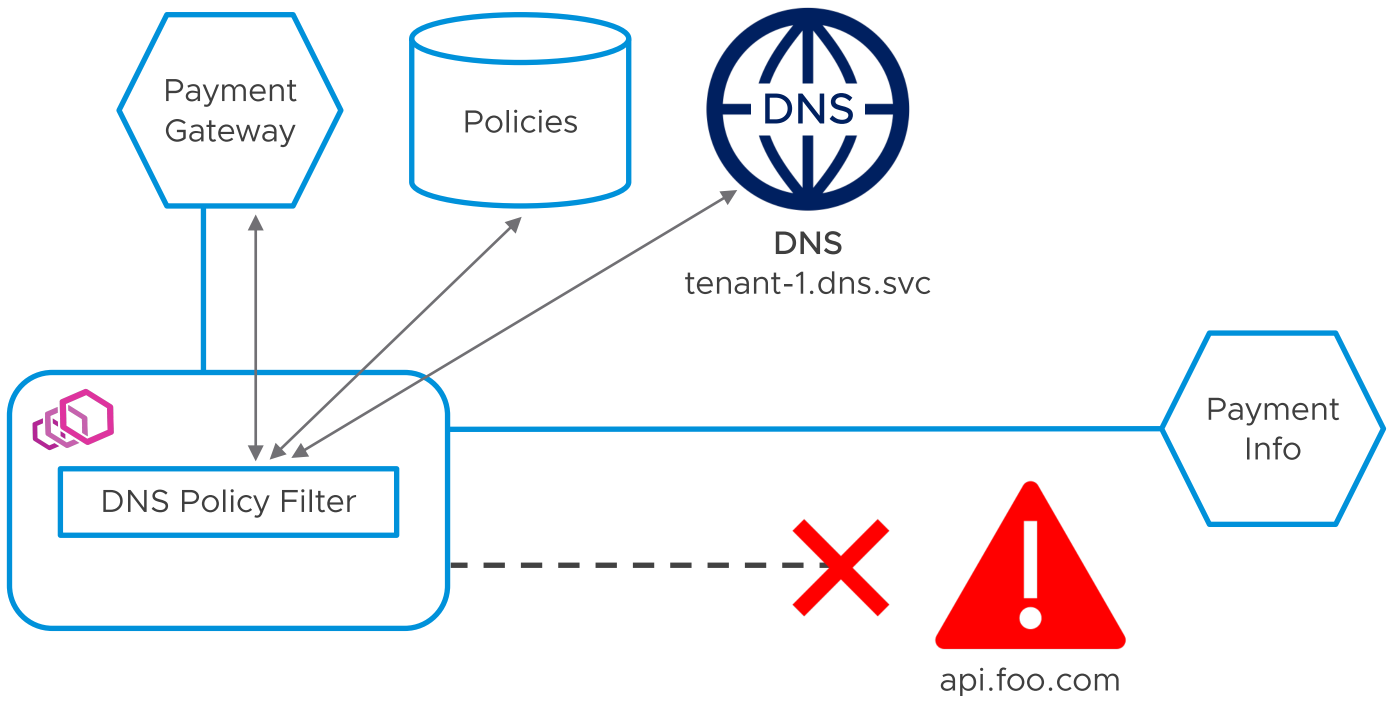
DNS Observability and Security



DNS Filtering

```yaml
apiVersion: "networking.example.com/v1alpha1"
kind: DnsPolicy
metadata:
  name: dns-policy
spec:
  namespaceSelector:
    matchLabels:
      workload: payments
  server:
    address: tenant-1.dns.svc
  policy:
    whitelist:
    - *.payments.svc.cluster.local # Can list local services.
    blacklist:
    - *.foo.com # Can list external domains.
    defaultAction: WARN # Or DENY or ALLOW.
```

DNS Filtering



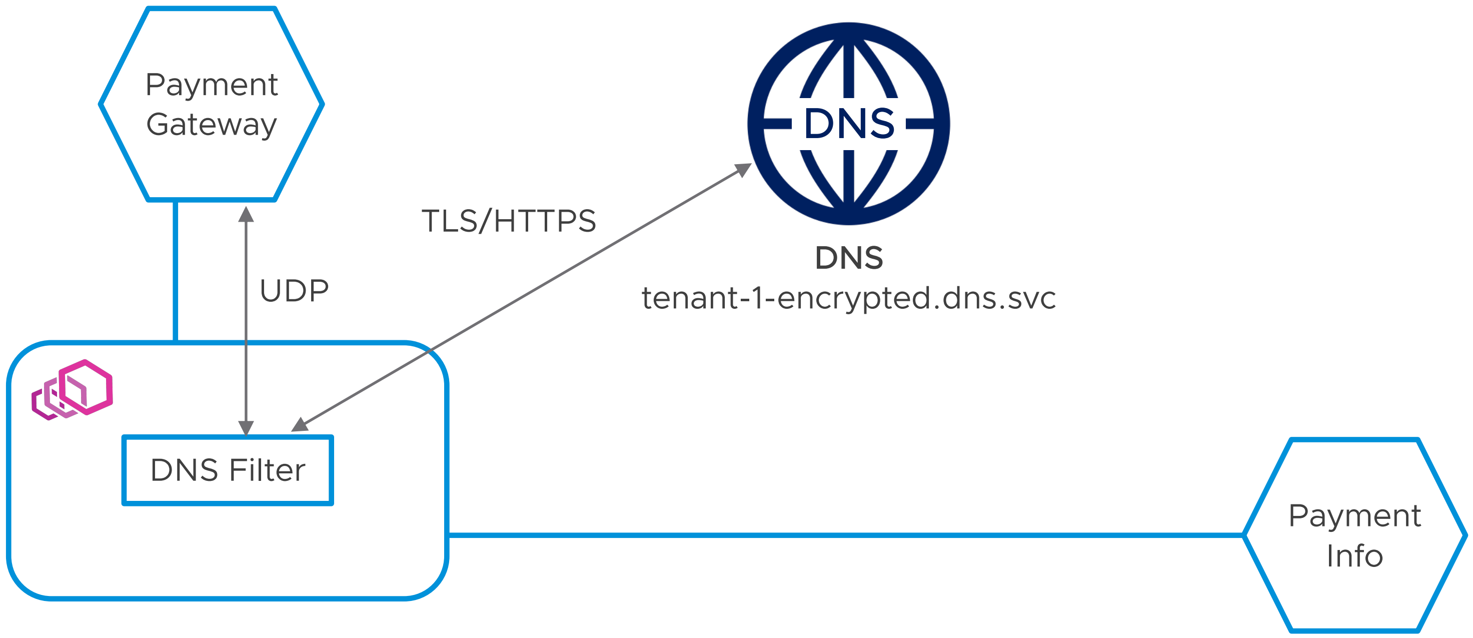
DNS Filtering

```
  spec:
    namespaceSelector:
      matchLabels:
        workload: payments
    selector:
      matchLabels:
        app: payment-gateway # Matcher for app level configuration.
    server:
      address: tenant-1.dns.svc # The DNS server.
    policy:
      defaultAction: DENY
      server:
        address: policy-server.acme.com # The DNS policy server.
        protocol: grpc # The DNS policy server protocol.
```

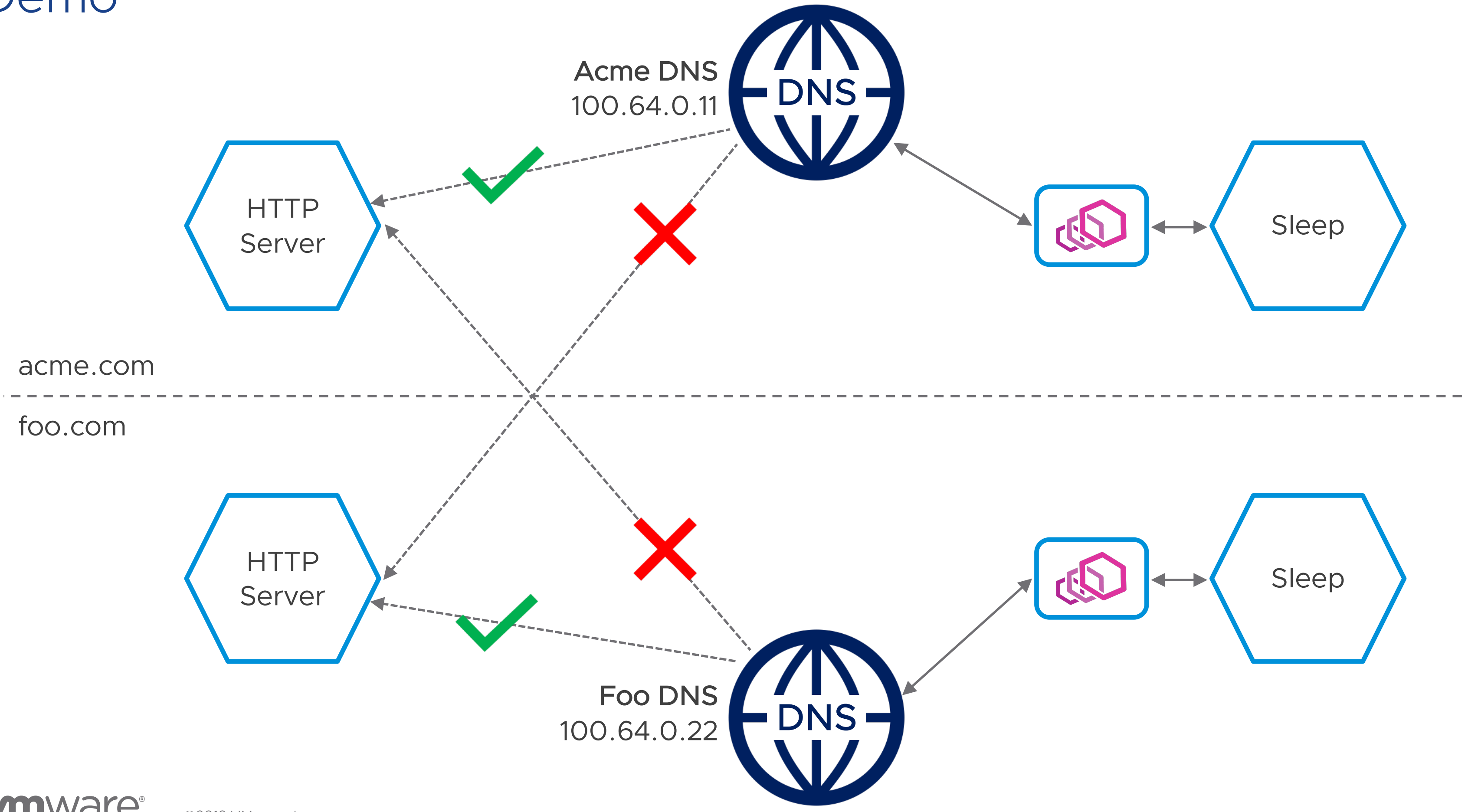
DNS Evolution

```yaml
apiVersion: "networking.example.com/v1alpha1"
kind: DnsPolicy
metadata:
  name: dns-policy
spec:
  namespaceSelector:
    matchLabels:
      workload: payments
  server:
    # A DNS server that supports encryption.
    address: tenant-1-encrypted.dns.svc
  protocol:
    upgrade: true # Upgrade from cleartext to HTTPS or TLS.
    type: dns-over-https # or dns-over-tls
```

DNS Evolution



Demo


Acme DNS
100.64.0.11
DNS
HTTP Server
Sleep
acme.com
foo.com
HTTP Server
Sleep
Foo DNS
100.64.0.22
DNS

```
$ kubectl get pods -n acme
NAME                       READY   STATUS    RESTARTS   AGE
coredns-77c65cbbc5-bhwzz   1/1     Running   0          5h34m
httpbin-5fc7cf895d-j2gll   1/1     Running   0          5h34m
sleep-5ffdbd896d-hfmb9     2/2     Running   0          5h34m
$
$
$ kubectl get pods -n foo
NAME                       READY   STATUS    RESTARTS   AGE
coredns-77c65cbbc5-t4zn4   1/1     Running   0          5h34m
httpbin-5fc7cf895d-x8t2s   1/1     Running   0          5h34m
sleep-5ffdbd896d-6rxc8     2/2     Running   0          5h34m
$
```

```
----------------------------------------------------------------
[acme:sleep] --> [acme:httpbin]

kubectl exec sleep-5ffdbd896d-hfmb9 \
                -n acme \
                -c sleep -- nslookup httpbin.acme.com

nslookup: can't resolve '(null)': Name does not resolve
Name:      httpbin.acme.com
Address 1: 100.66.185.144
Address 2: 100.66.185.144
----------------------------------------------------------------
[foo:sleep] --> [foo:httpbin]

kubectl exec sleep-5ffdbd896d-6rxc8 \
                -n foo \
                -c sleep -- nslookup httpbin.foo.com
nslookup: can't resolve '(null)': Name does not resolve
```

```
Address 1: 100.66.185.144
Address 2: 100.66.185.144
----------------------------------------------------------------
[foo:sleep] --> [foo:httpbin]

kubectl exec sleep-5ffdbd896d-6rxc8 \
                -n foo \
                -c sleep -- nslookup httpbin.foo.com
nslookup: can't resolve '(null)': Name does not resolve

Name:      httpbin.foo.com
Address 1: 100.68.169.65
Address 2: 100.68.169.65
----------------------------------------------------------------
[acme:sleep] --> [foo:httpbin]

kubectl exec sleep-5ffdbd896d-hfmb9 \
                -n acme \
                -c sleep -- nslookup httpbin.foo.com
```

```
Address 1: 100.68.169.65
Address 2: 100.68.169.65
----------------------------------------------------------------
[acme:sleep] --> [foo:httpbin]

kubectl exec sleep-5ffdbd896d-hfmb9 \
                -n acme \
                -c sleep -- nslookup httpbin.foo.com
nslookup: can't resolve '(null)': Name does not resolve

nslookup: can't resolve 'httpbin.foo.com': Try again
command terminated with exit code 1
make: [demo] Error 1 (ignored)
----------------------------------------------------------------
[foo:sleep] --> [acme:httpbin]

kubectl exec sleep-5ffdbd896d-6rxc8 \
                -n foo \
                -c sleep -- nslookup httpbin.acme.com
                -n acme \
                -c sleep -- nslookup httpbin.foo.com
nslookup: can't resolve '(null)': Name does not resolve

nslookup: can't resolve 'httpbin.foo.com': Try again
command terminated with exit code 1
make: [demo] Error 1 (ignored)
----------------------------------------------------------------
[foo:sleep] --> [acme:httpbin]

kubectl exec sleep-5ffdbd896d-6rxc8 \
                -n foo \
                -c sleep -- nslookup httpbin.acme.com
nslookup: can't resolve '(null)': Name does not resolve

nslookup: can't resolve 'httpbin.acme.com': Try again
command terminated with exit code 1
make: [demo] Error 1 (ignored)
----------------------------------------------------------------
$
```

Summary

- DNS plays a key role for service discovery and application migration
- Multi-tenancy at the DNS layer is very critical for enterprise systems
- Envoy proxy can solve some interesting challenges with DNS
- Envoy proxy’s xDS APIs let us dynamically configure DNS filters
- The DNS filters can also be integrated with third-party systems to provide richer observability, security, and filtering
- Next: Contribute the work to existing open source projects!

Thank You

@deepakalani | **Deepa Kalani**

@venilnoronha | **Venil Noronha**

istio.io
envoyproxy.io
venilnoronha.io