

Scaling container policy management with kernel features

Joe Stringer

Cilium.io

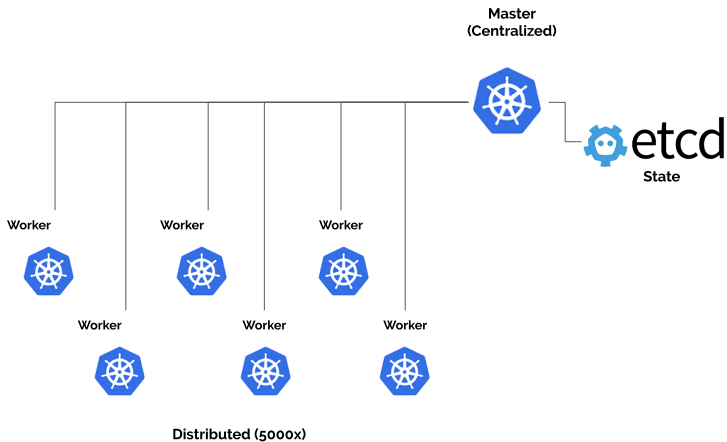
Linux Plumbers 2019, Lisbon, Portugal



Overview

- 1 Background
- 2 Deploying fast datapaths fast
- 3 Identity-based security
- 4 Layer 7 security

Kubernetes Architecture 101



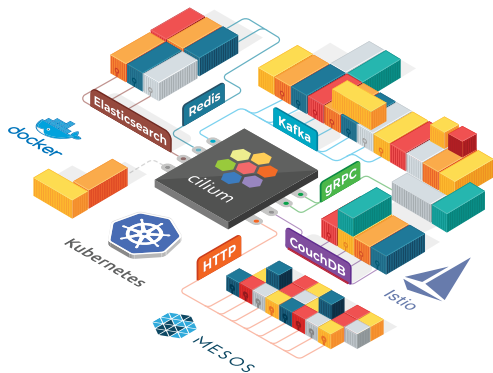
Kubernetes networking plugins

- Plumb local connectivity (CNI)
- Connect remote nodes
- Services / loadbalancing
- Network policy



Cilium

- Agent runs on each node
- Native eBPF dataplane
- Identity-based security
- Scalable

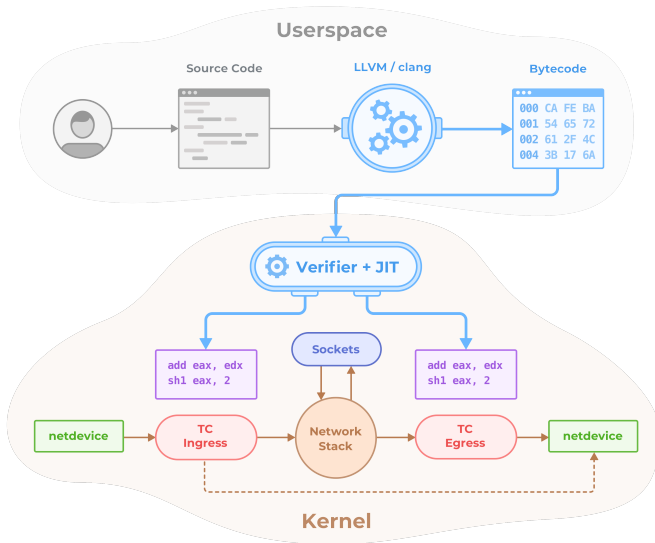


What does it mean to scale?

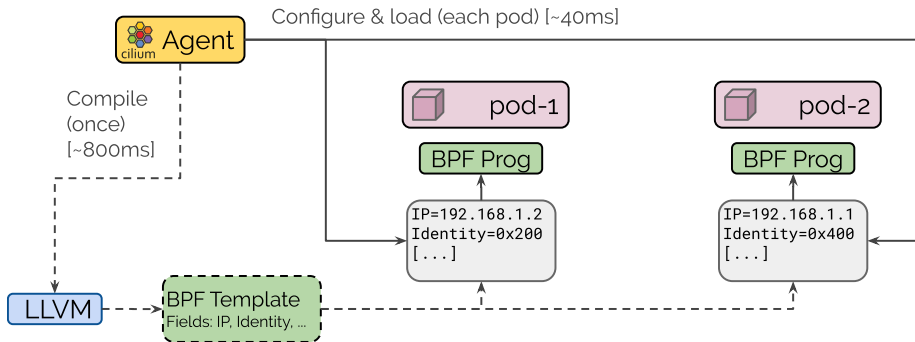
- Manage cluster interactions
 - Minimize unnecessary events
 - Reduce event sizes
 - ...
- Optimize work within the node
 - Apply datapath changes efficiently

Deploying fast datapaths fast

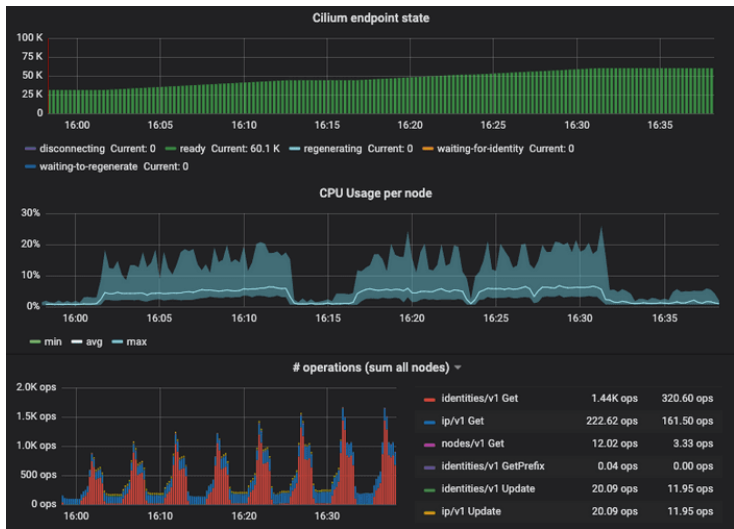
BPF plumbing



ELF Templating



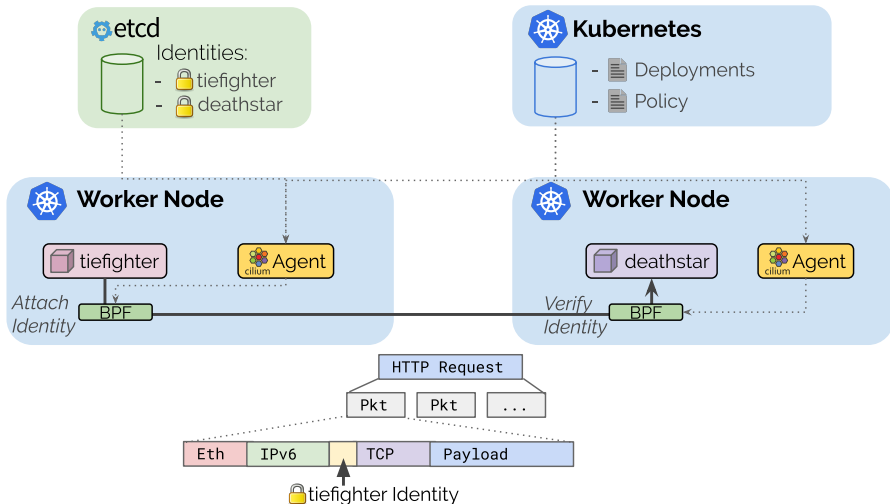
1K nodes: Scaling to 60k pods



Future directions

- Optimize verifier execution: $O(n) \rightarrow O(1)$
- Support code path templatzation

Identity-based security



Policy example

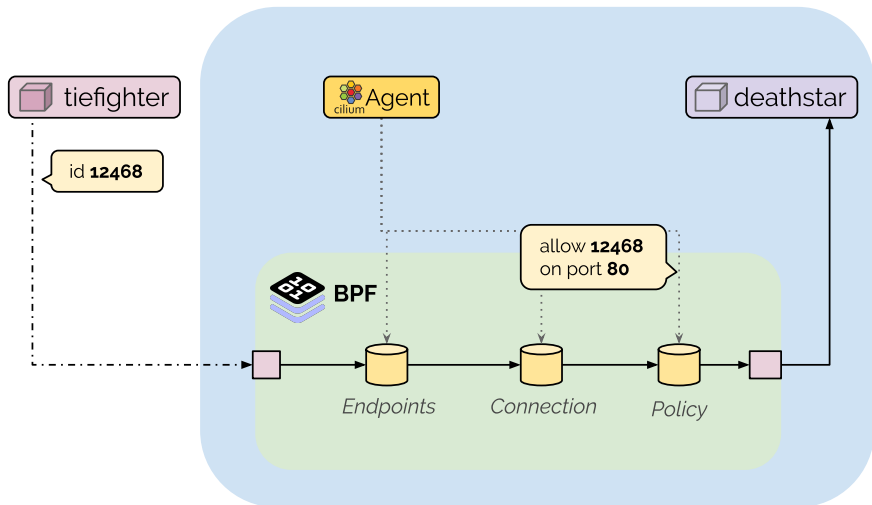
```
apiVersion: "cilium.io/v2"
kind: CiliumNetworkPolicy
description: "Restrict deathstar access to empire ships"
metadata:
  name: "deathstar-ingress"
spec:
  endpointSelector:
    matchLabels:
      org: empire
      class: deathstar
  ingress:
    - fromEndpoints:
        - matchLabels:
            org: empire
      toPorts:
        - ports:
            - port: "80"
              protocol: TCP
```

Label selectors

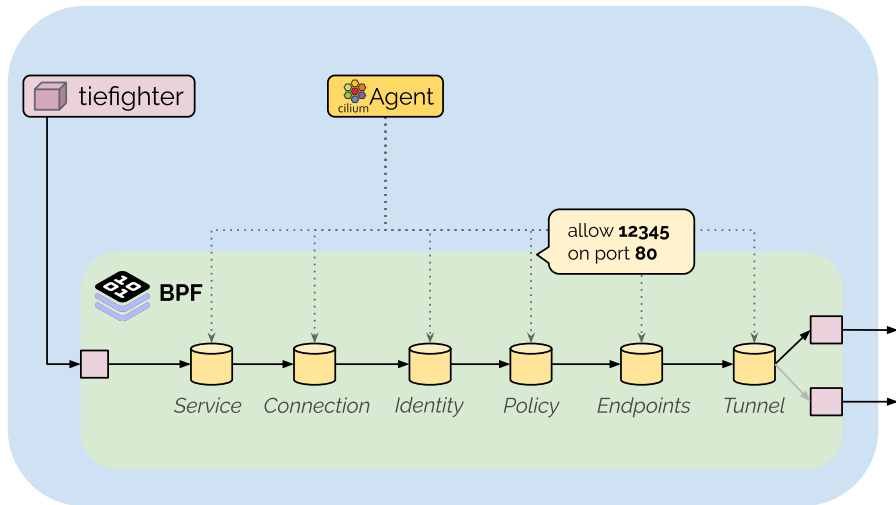
- 12345 {org:empire, class:deathstar}
- 12468 {org:empire, class:tiefighter}
- 12465 {org:alliance, class:xwing}

<code>matchLabels: org: empire</code>	<code>12345 {org:empire, class:deathstar} 12468 {org:empire, class:tiefighter}</code>
<code>matchLabels: org: empire class: deathstar</code>	<code>12345 {org:empire, class:deathstar}</code>

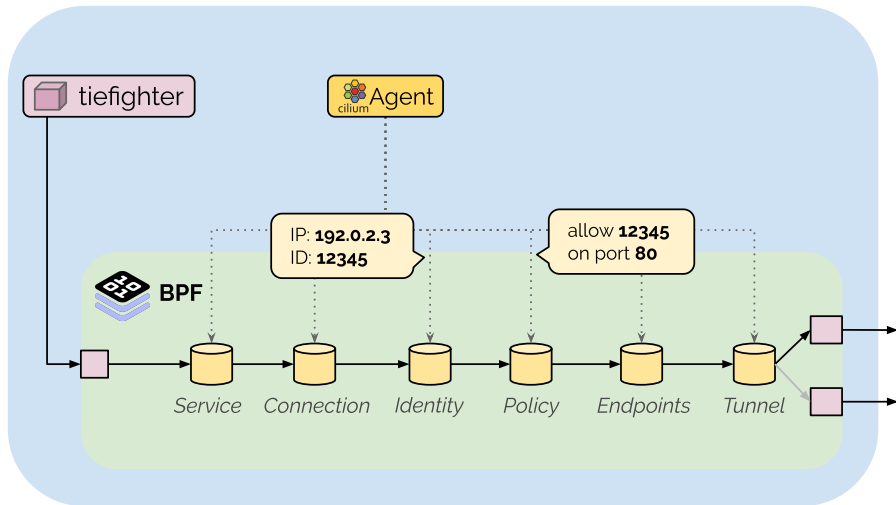
Datapath Configuration: Ingress



Datapath Configuration: Egress

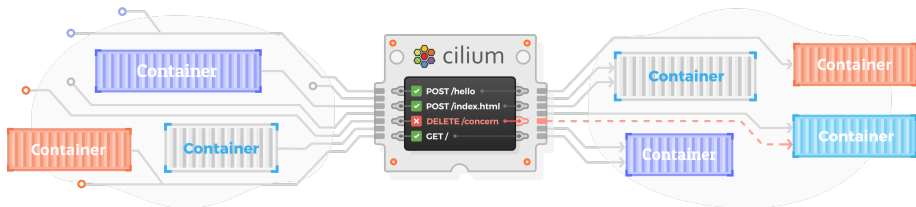


Datapath Configuration: Egress



Layer 7 security

L7 is the new L4



Rejecting traffic in a protocol-aware manner

```
$ kubectl exec tiefighter -- \  
> curl -s -XPOST deathstar.default.svc.cluster.local/v 1 /request-landing
```

Rejecting traffic in a protocol-aware manner

```
$$ kubectl exec tiefighter -- \
> curl -s -XPOST deathstar.default.svc.cluster.local/v 1 /request-landing
Ship landed
$$ █
```

Rejecting traffic in a protocol-aware manner

```
$ kubectl exec tiefighter -- \
> curl -s -XPOST deathstar.default.svc.cluster.local/v 1 /request-landing
Ship landed
$
$
$ kubectl exec tiefighter -- \
> curl -s XPUT deathstar.default.svc.cluster.local/v 1 /exhaust-port
```

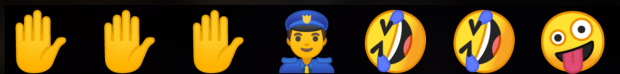
Rejecting traffic in a protocol-aware manner

```
$ kubectl exec tiefighter -- \
> curl -s -XPOST deathstar.default.svc.cluster.local/v 1 /request-landing
Ship landed
$
$
$ kubectl exec tiefighter -- \
> curl -s XPUT deathstar.default.svc.cluster.local/v 1 /exhaust-port
👋👋👋👮🤖🤖😬
```

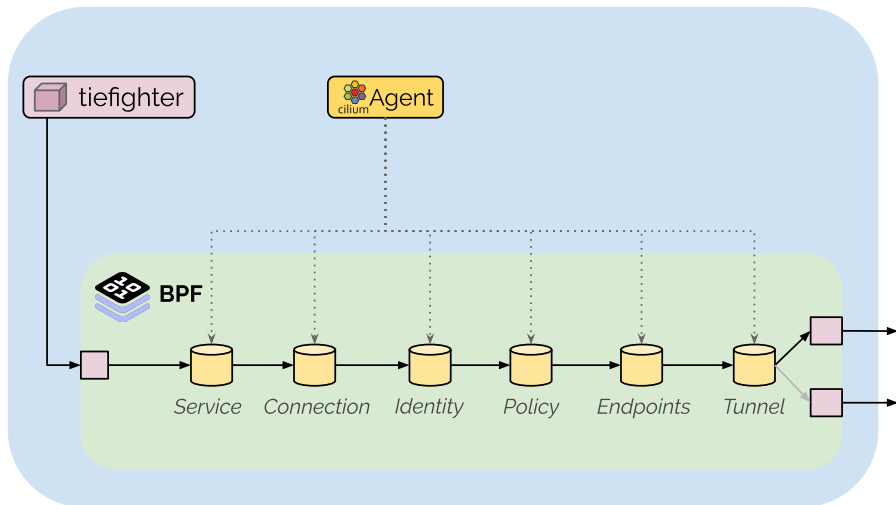
Rejecting traffic in a protocol-aware manner

```
$ kubectl exec tiefighter -- \
> curl -s -XPOST deathstar.default.svc.cluster.local/v 1 /request-landing
Ship landed
```

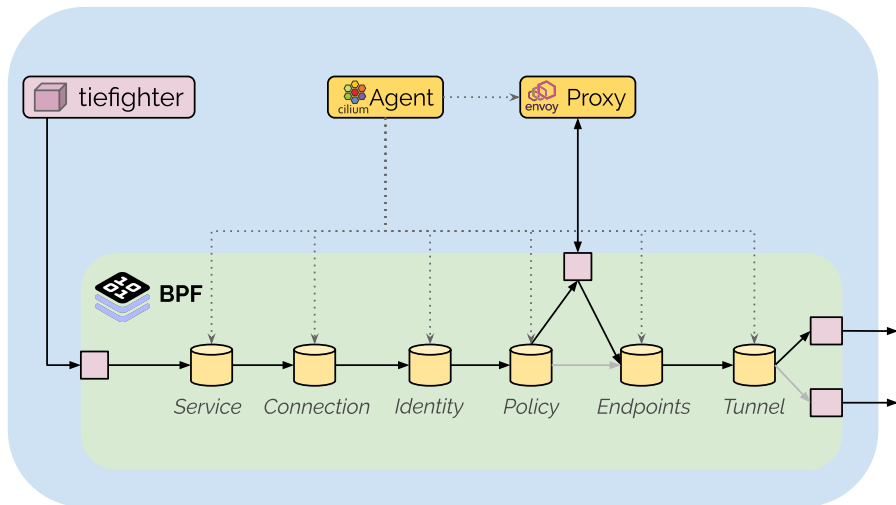
```
$ kubectl exec tiefighter -- \
> curl -s XPUT deathstar.default.svc.cluster.local/v 1 /exhaust-port
```



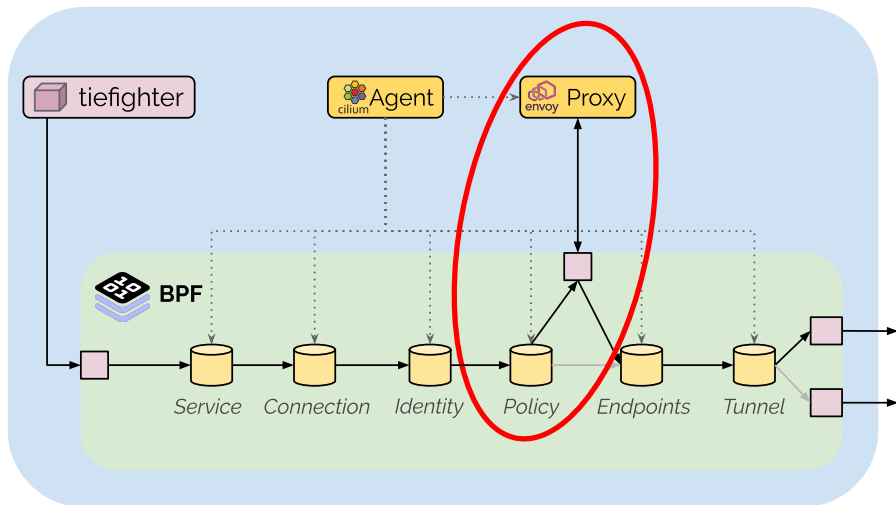
Datapath Configuration: L3 flow



Datapath Configuration: L7 flow

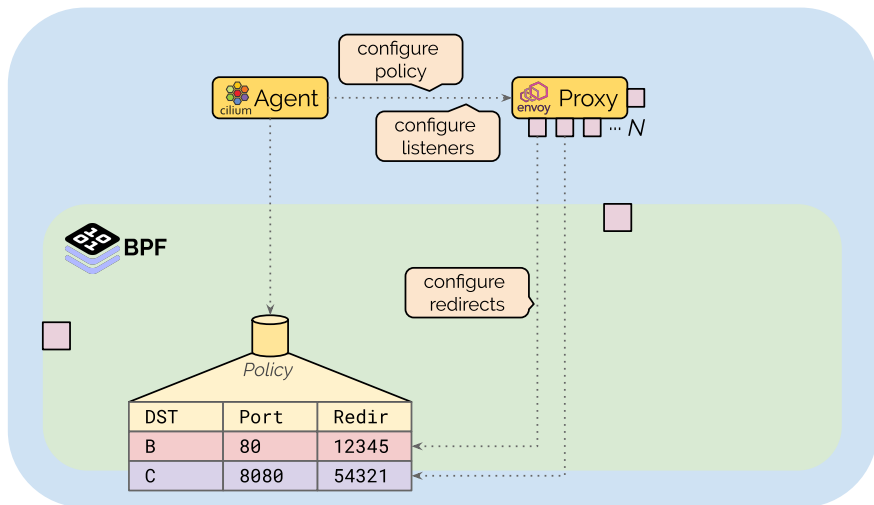


Datapath Configuration: L7 flow



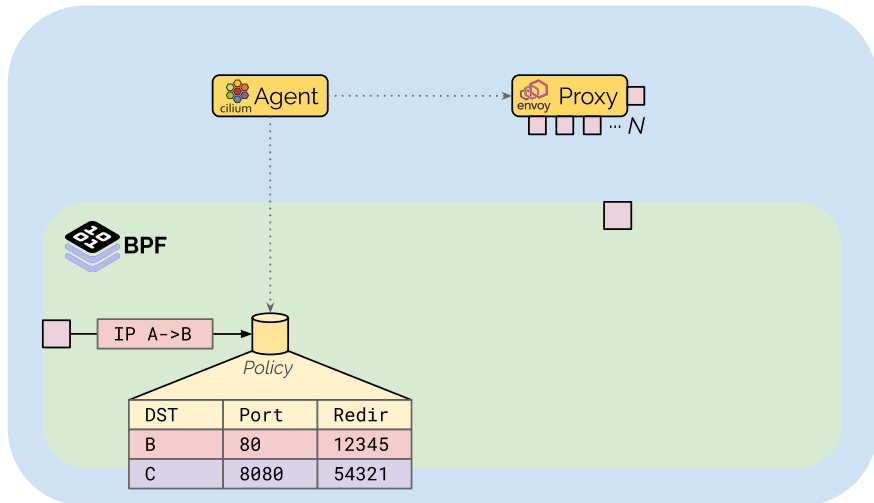
L7 Configuration: Past

Per-endpoint configuration



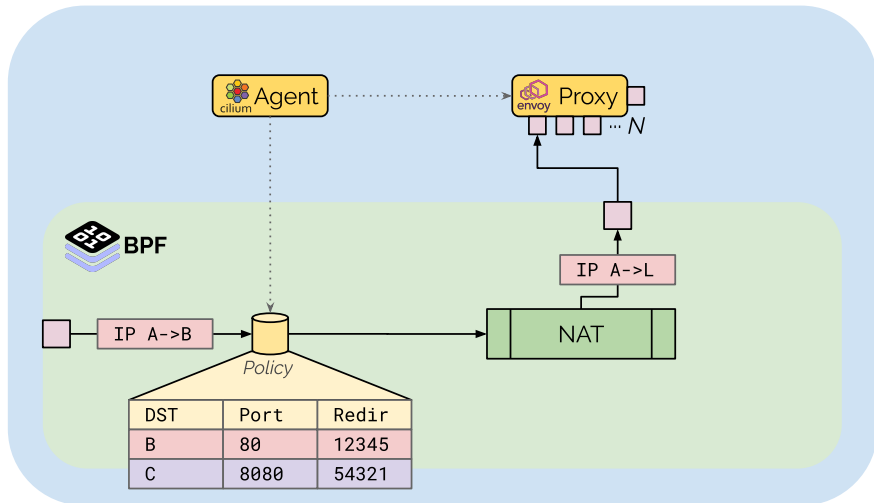
A: 192.0.2.3:33333 ; B: 192.0.2.4:80 ; L: 192.0.2.1:12345

L7 Configuration: Past



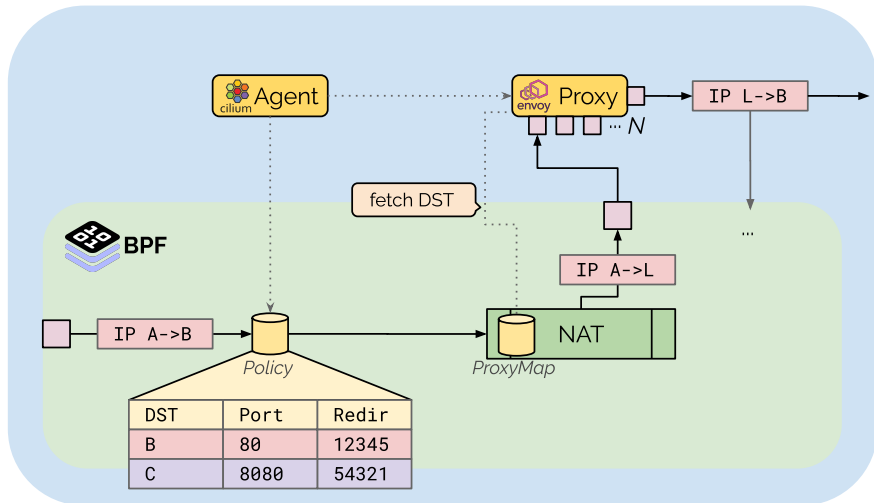
A: 192.0.2.3:33333 ; B: 192.0.2.4:80 ; L: 192.0.2.1:12345

L7 Configuration: Past



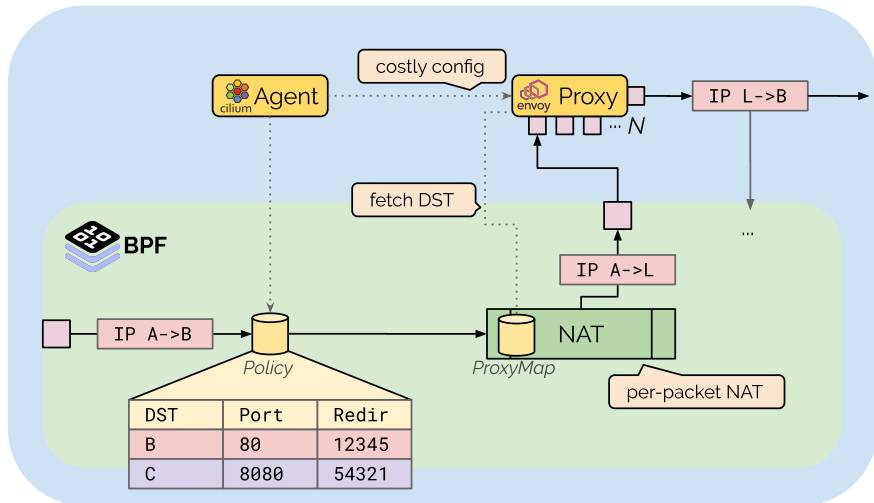
A: 192.0.2.3:33333 ; B: 192.0.2.4:80 ; L: 192.0.2.1:12345

L7 Configuration: Past



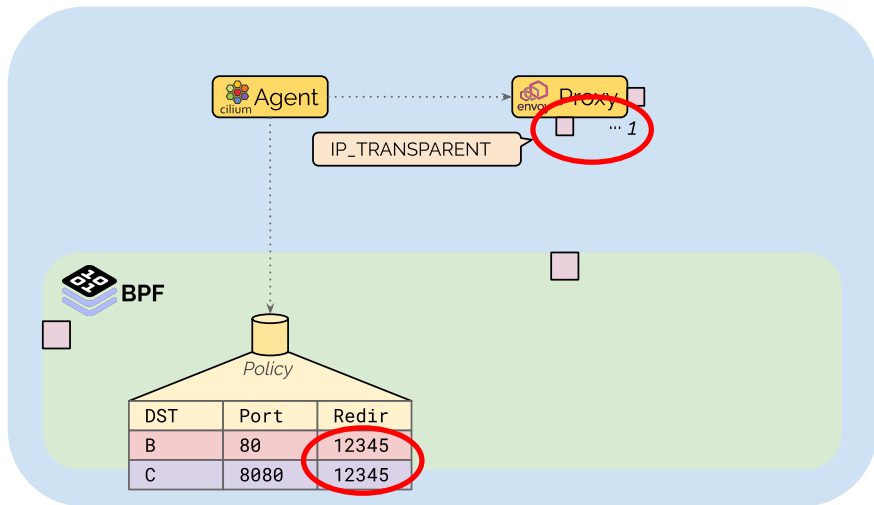
A: 192.0.2.3:33333 ; B: 192.0.2.4:80 ; L: 192.0.2.1:12345

L7 Configuration: Past

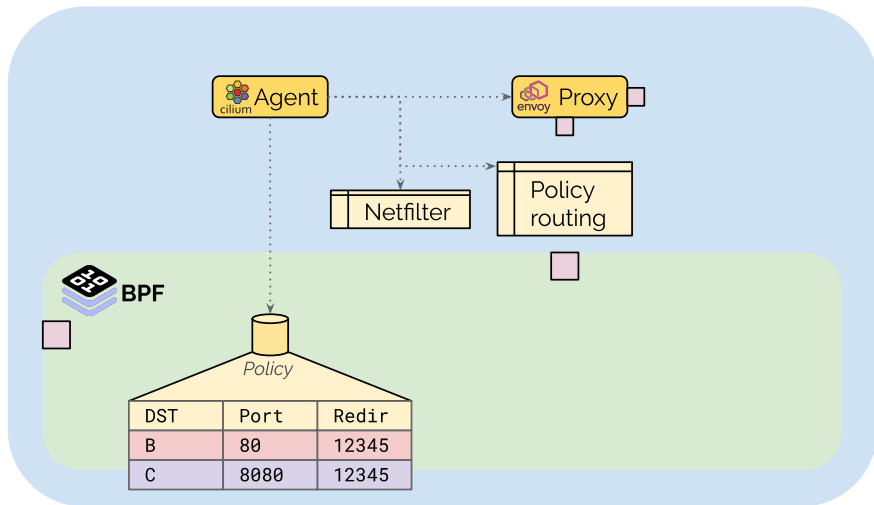


A: 192.0.2.3:33333 ; B: 192.0.2.4:80 ; L: 192.0.2.1:12345

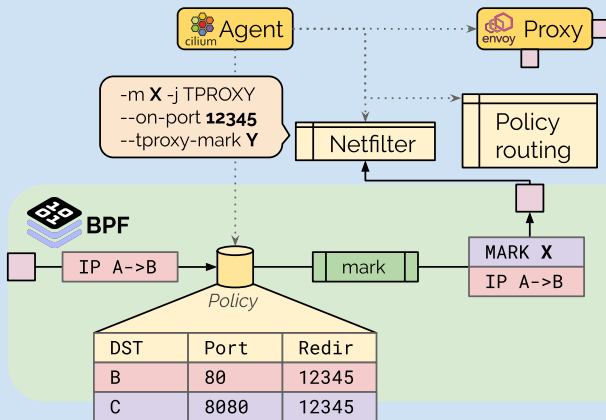
L7 Configuration: Present



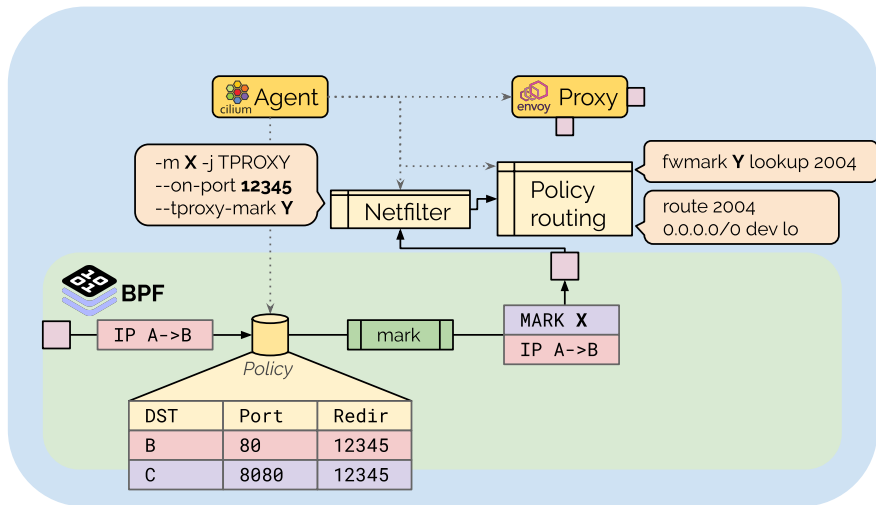
L7 Configuration: Present



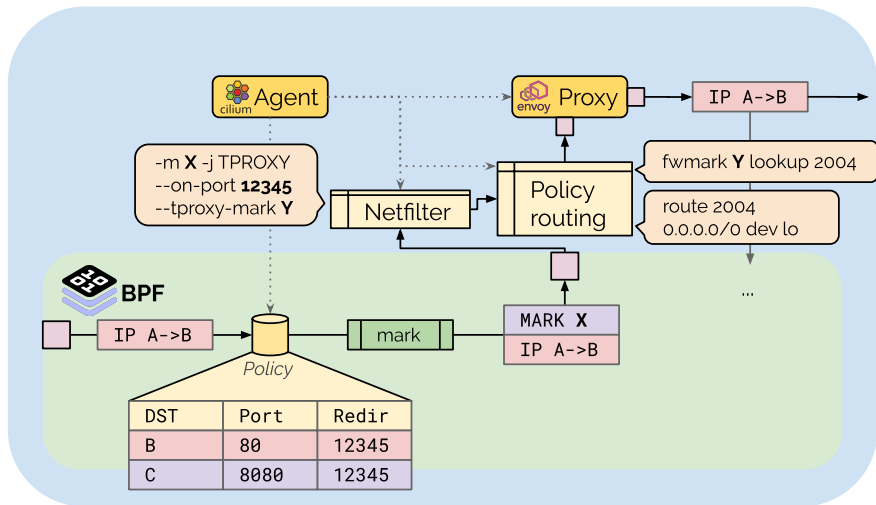
L7 Configuration: Present



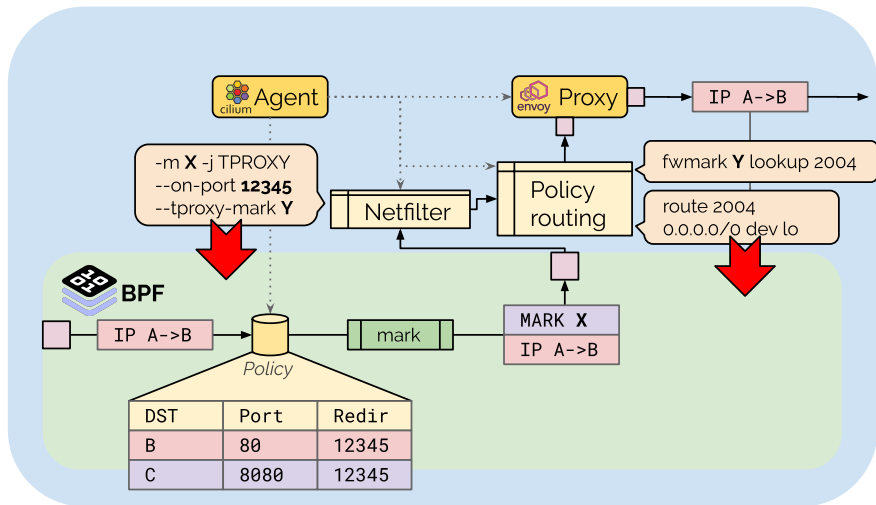
L7 Configuration: Present



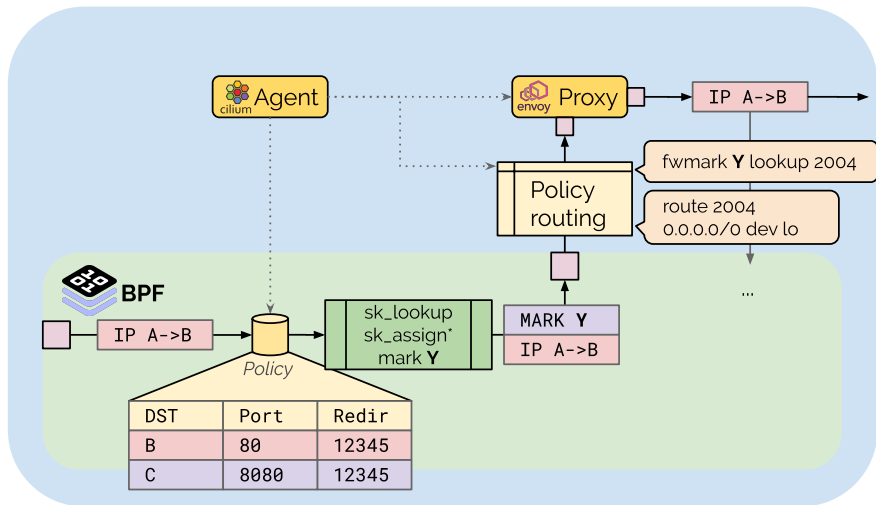
L7 Configuration: Present



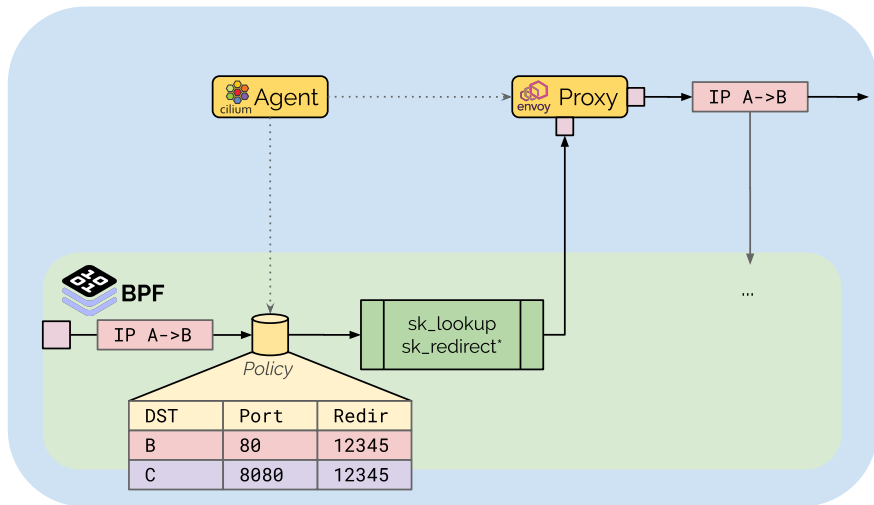
L7 Configuration: Proposal



L7 Configuration: Proposal



L7 Configuration: Socket redirect



Socket assign: Hiccup

- BPF progs are attached at TC ingress
- `skb_orphan()` invoked directly before PREROUTING

¹<https://www.mail-archive.com/netdev@vger.kernel.org/msg303851.html>

²<https://www.mail-archive.com/netdev@vger.kernel.org/msg304057.html>

Socket assign: Hiccup

- BPF progs are attached at TC ingress
- `skb_orphan()` invoked directly before PREROUTING
- TC folks are already carrying hacks for this¹

¹<https://www.mail-archive.com/netdev@vger.kernel.org/msg303851.html>

²<https://www.mail-archive.com/netdev@vger.kernel.org/msg304057.html>

Socket assign: Hiccup

- BPF progs are attached at TC ingress
- `skb_orphan()` invoked directly before PREROUTING
- TC folks are already carrying hacks for this¹
- Just move to `____dev_forward_skb()`²?

¹<https://www.mail-archive.com/netdev@vger.kernel.org/msg303851.html>

²<https://www.mail-archive.com/netdev@vger.kernel.org/msg304057.html>

Summary

- Minimize processing cost
 - Number of events
 - Cost for each event
- Separate concerns: Policy vs addressing
- Frontload expensive operations
- ... while keeping runtime costs low

Thank you

More information

-  <https://cilium.io>
-  <https://cilium.io/slack>
-  <https://github.com/cilium/cilium>
-  <https://twitter.com/ciliumproject>



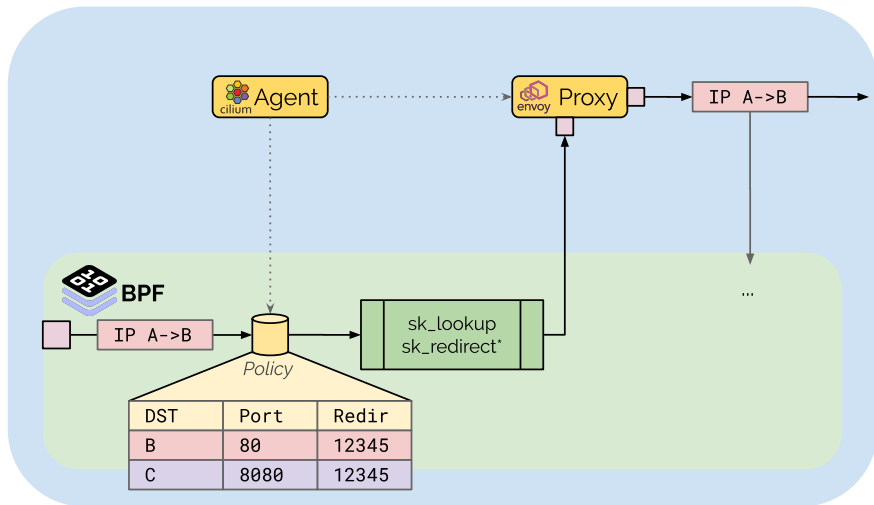
Thank you

More information

-  <https://cilium.io>
-  <https://cilium.io/slack>
-  <https://github.com/cilium/cilium>
-  <https://twitter.com/ciliumproject>



Socket redirect



Early demux

```
int tcp_v4_early_demux(struct sk_buff *skb) {  
    // ... snipped: validate packet type, tcphdr offsets  
  
    struct sock *sk = __inet_lookup_established(dev_net(skb->dev),  
                                                &tcp_hashinfo, iph->saddr, th->source,  
                                                iph->daddr, ntohs(th->dest),  
                                                skb->skb_iif, inet_sdif(skb));  
  
    if (sk) {  
        skb->sk = sk;  
        skb->destructor = sock_edemux;  
        if (sk_fullsock(sk)) {  
            struct dst_entry *dst = READ_ONCE(sk->sk_rx_dst);  
  
            if (dst)  
                dst = dst_check(dst, 0);  
            if (dst &&  
                inet_sk(sk)->rx_dst_ifindex == skb->skb_iif)  
                skb_dst_set_noref(skb, dst);  
        }  
    }  
    return 0;  
}
```

Socket assign: RFC

```
/* int bpf_sk_assign(struct sk_buff *skb, struct bpf_sock *sk,
 *                  u64 flags)
 *
 * Description
 *   Assign the *sk* to the *skb*.
 *
 *   This operation is only valid from TC ingress path.
 *
 *   The *flags* argument must be zero.
 *
 * Return
 *   0 on success, or a negative errno in case of failure.
 *
 *   * **--EINVAL**      Unsupported flags specified.
 *   * **--EOPNOTSUPP**:: Unsupported operation, for example
 *                       a call from outside of TC ingress.
 *   * **--ENOENT**::    The socket cannot be assigned.
 */
```

Socket assign: API quirks

- Nit: Need to use `sock_common` socket lookups
- Add `bpf_skc_lookup_udp()`
- Optimization: Add lookup flags to `bpf_sk*_lookup_*`()

ELF Templating: Initial implementation

- Treat ELF symbol table as a data table
- Declare symbols for all static data (constants)
- (hack) Cast desired static data symbol pointer to data
- Copy ELF, replacing values in symbol table with desired constants
- Loader relocation copies the value from the symbol table into the instructions

ELF Templating: Upstream libbpf

- Create maps for `.bss`, `.data`, `.rodata`
- Relocate loads from static symbols to be map loads
- Kernel-side, fix up to be direct load