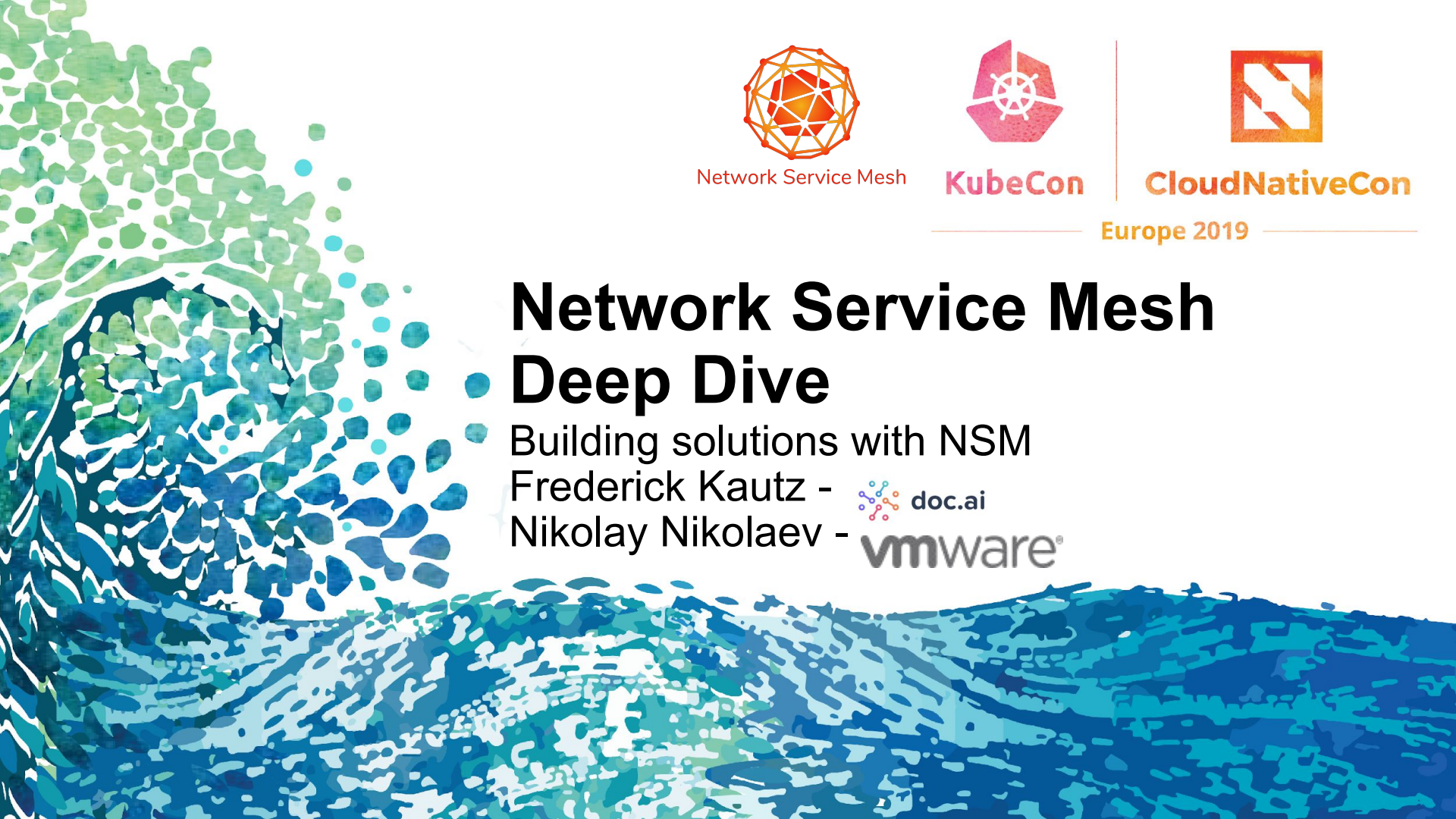


Network Service Mesh Deep Dive

Building solutions with NSM

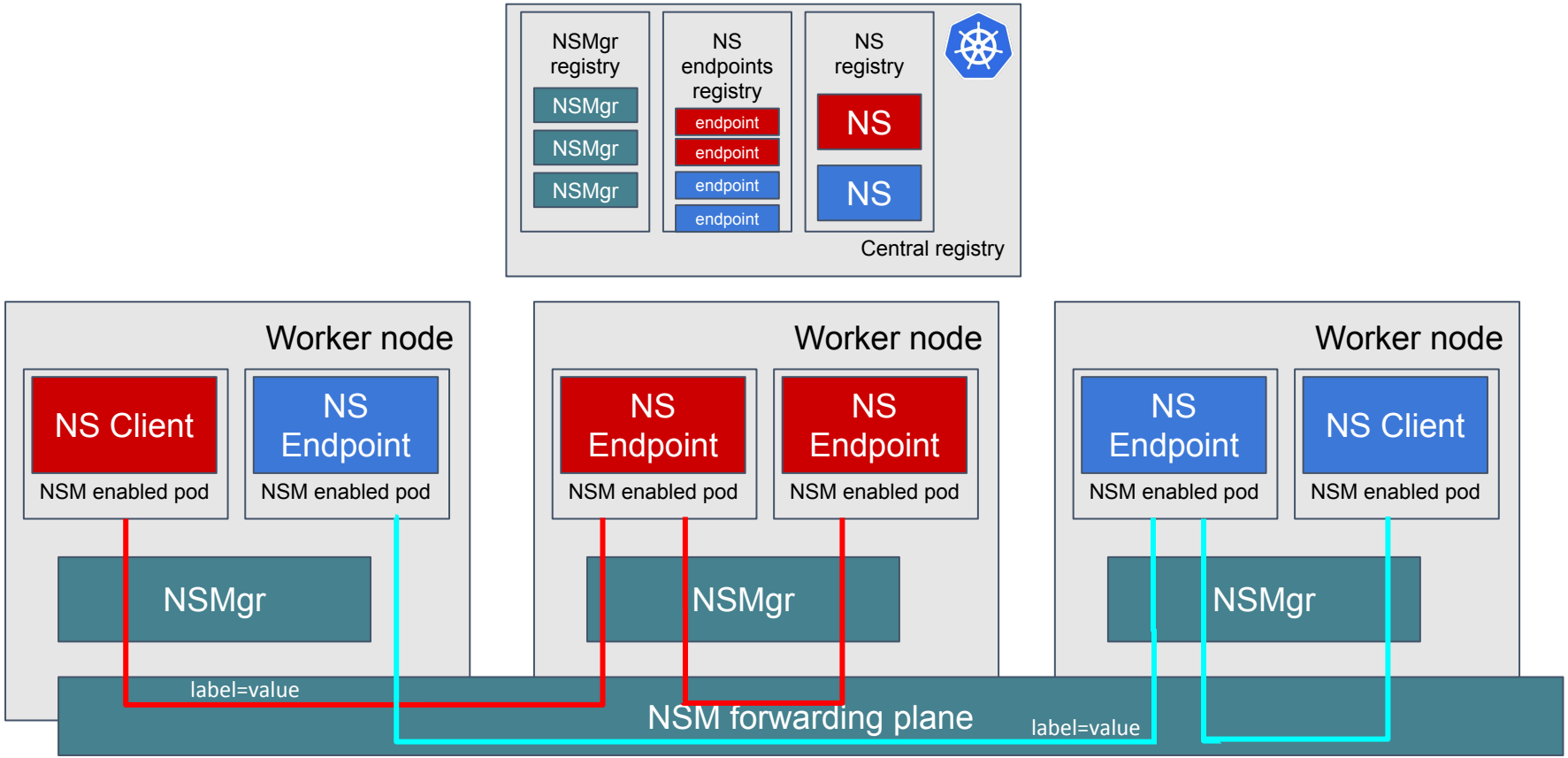
Frederick Kautz - doc.ai
Nikolay Nikolaev - vmware

Giant QR Code to this Presentation

Network Service Mesh

- A Network Service definition
- A gRPC API to describe, publish and consume Network Service(s)
- A distributed control plane with minimum shared state
- A concrete Kubernetes based implementation
    - Runtime interface injection/removal for Pods. Orthogonal to CNI
    - Leverage etcd as a central shared storage through CRDs
    - Use Kubernetes `DaemonSet` to provision local node agents
    - VPP as a base forwarding component

Network Service Mesh overview



Network Service Manifest - yaml overview


```
apiVersion: networkservicemesh.io/v1
kind: NetworkService
metadata:
 name: secure-intranet-connectivity
spec:
 payload: IP
 matches:
   - match:
     sourceSelector:
       app: firewall
     route:
       - destination:
         destinationSelector:
           app: vpn-gateway
   - match:
     route:
       - destination:
         destinationSelector:
           app: firewall
```

Describe the type `NetworkService`

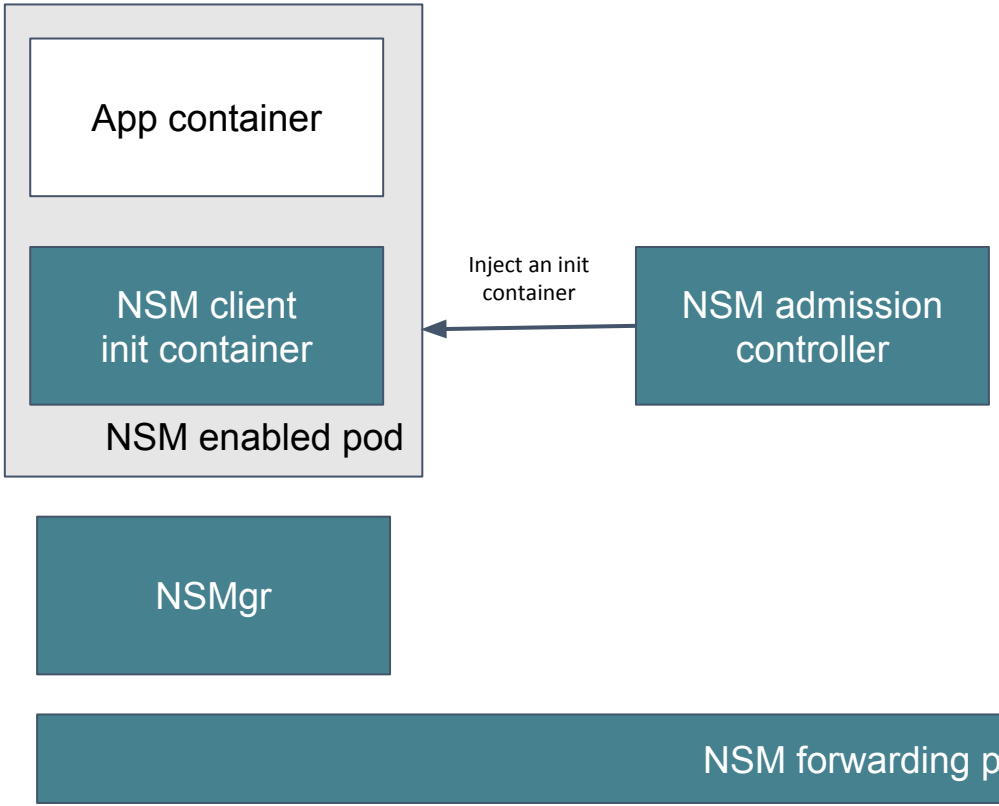
The name of the service is `secure-intranet-connectivity`

Match the service request labels for `app=firewall`

Find an endpoint that implements `secure-intranet-connectivity` and is labeled `app=vpn-gateway`

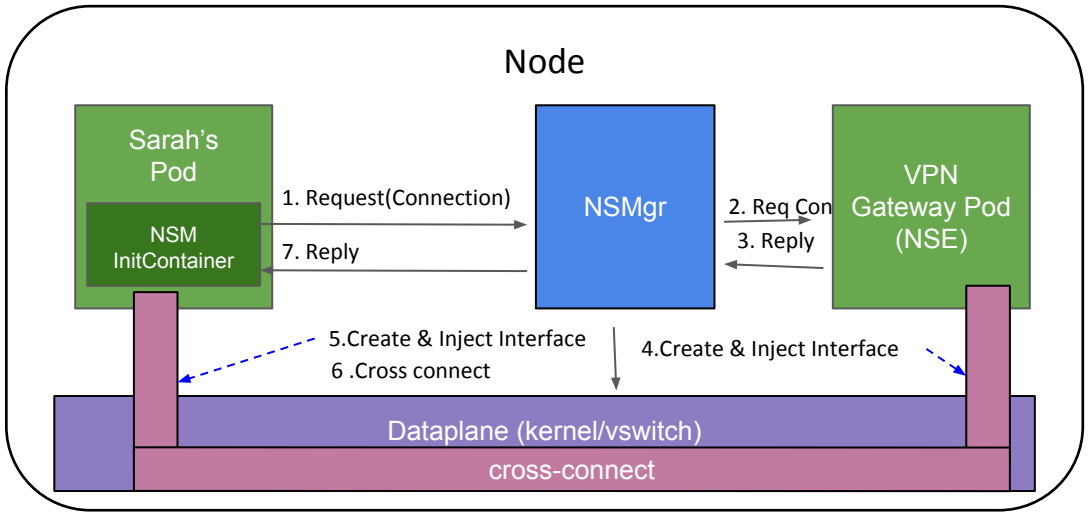
Wildcard `sourceSelector`

Intro to the SDK - NS Client with an init container

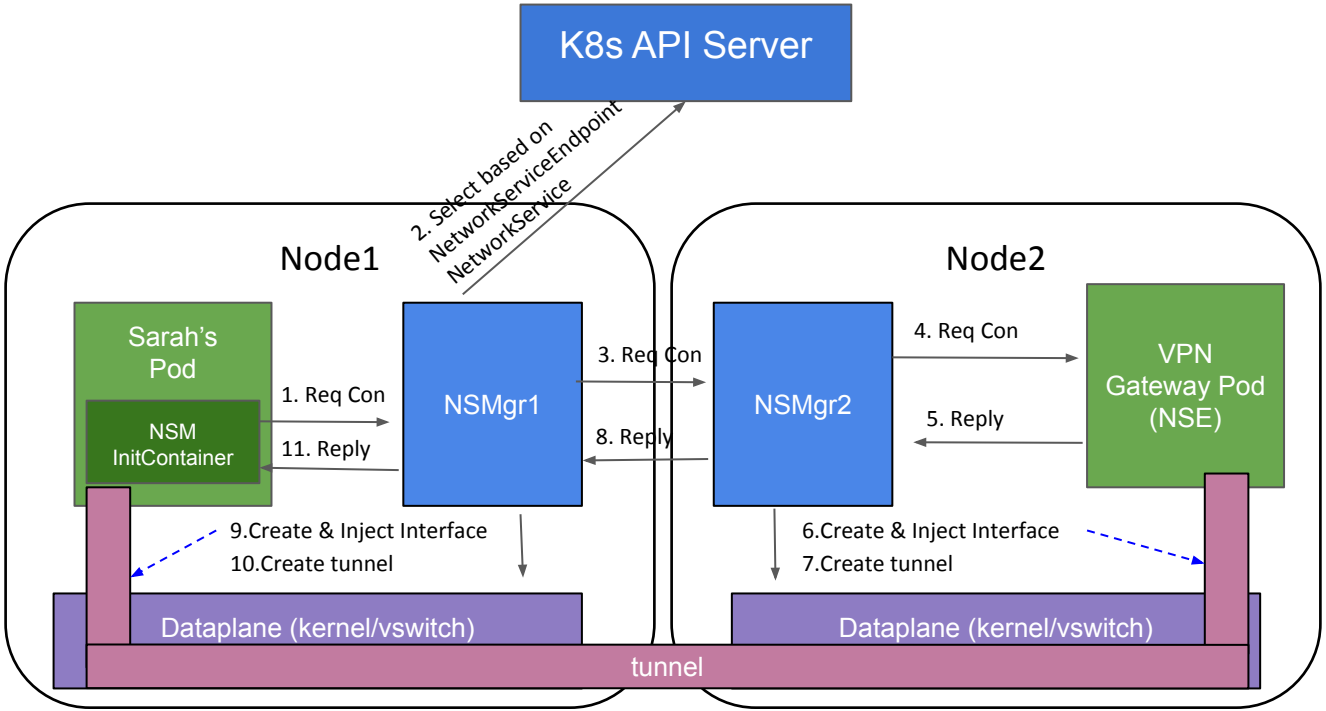



```
apiVersion: apps/v1
kind: Deployment
spec:
 template:
   spec:
     containers:
       - name: alpine-img
         image: alpine:latest
         command: ['tail', '-f', '/dev/null']
metadata:
 name: my-app
 annotations:
   ns.networkservicemesh.io: service?label=valu
```

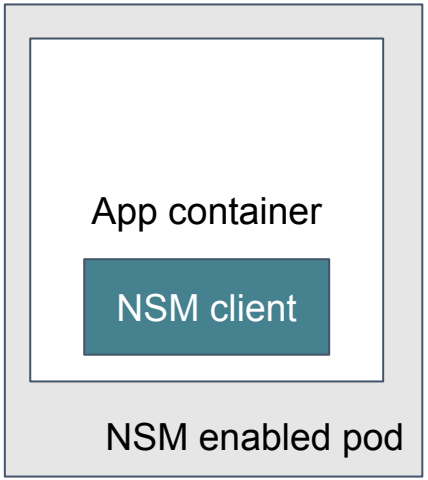
Network Service Manifest - yaml overview



Network Service Manifest - yaml overview



Intro to the SDK - writing a *smart* client



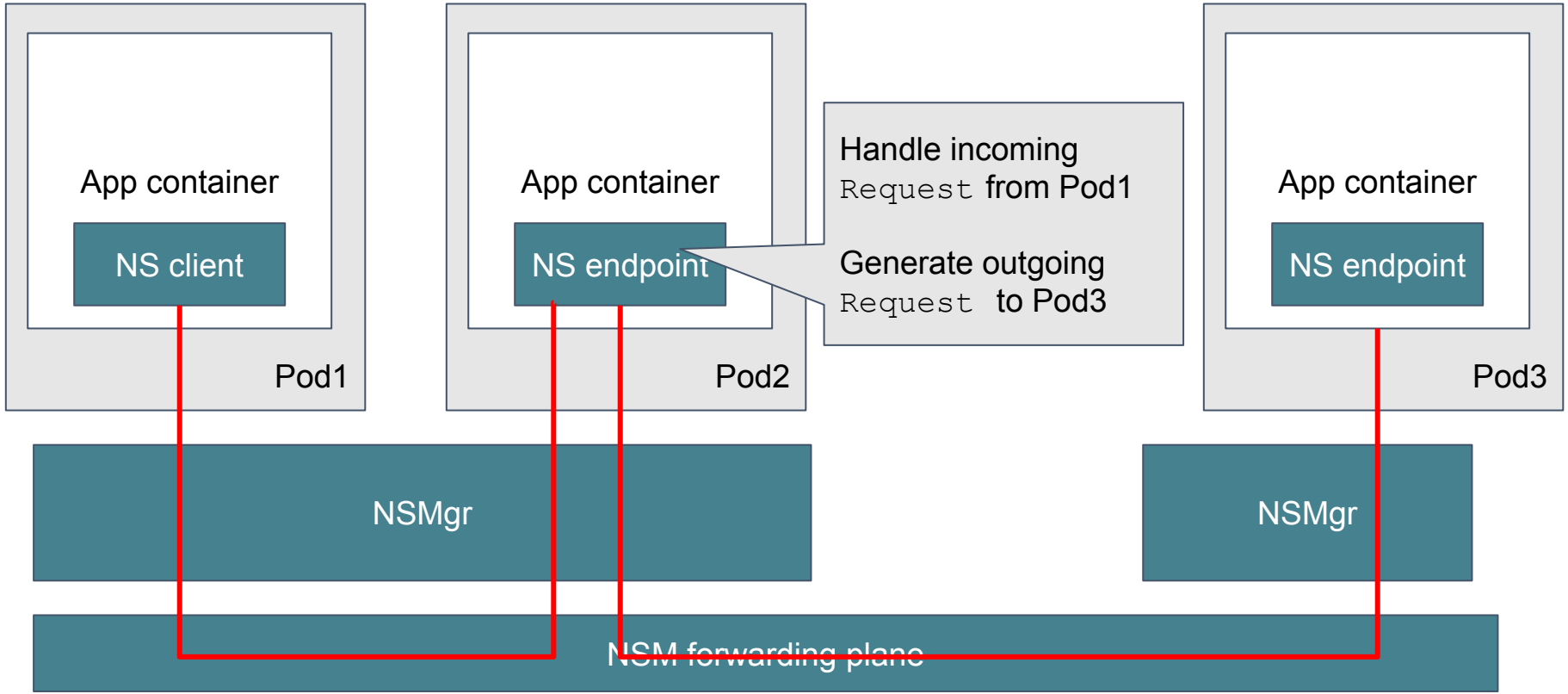

```
"id:"2"
network_service:"secure-intranet"
mechanism:
      < type:KERNEL_INTERFACE
        parameters: <key:"description" value:"NSM Endpoint" >
        parameters:<key:"name" value:"nsm1pmjvpppj" >
        parameters:<key:"netnsInode" value:"4026533759" >
        parameters:<key:"socketfile" value:"nsm1pmjvpppj/memif.sock" > >
context:
      < src_ip_addr:"10.60.1.1/30" dst_ip_addr:"10.60.1.2/30"
        src_ip_required:true dst_ip_required:true
        routes:<prefix:"8.8.8.8/30" >
excluded_prefixes:"10.244.0.0/16" excluded_prefixes:"10.96.0.0/12"
excluded_prefixes:"10.244.0.0/16" excluded_prefixes:"10.96.0.0/12"
ip_neighbors:< ip:"10.244.1.7" hardware_address:"02:f8:80:95:9b:83" >
ip_neighbors:< ip:"fe80::f8:80ff:fe95:9b83" hardware_address:"02:f8:80:95:9b:83" >
```

NSMgr

```
// Ensure the client is terminated at the end
client.Destroy()
```

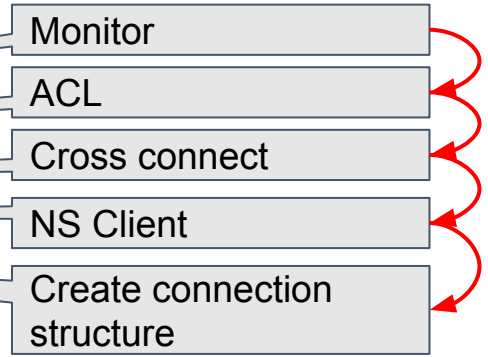
NSM forwarding plane

Intro to the SDK - Endpoint Composition

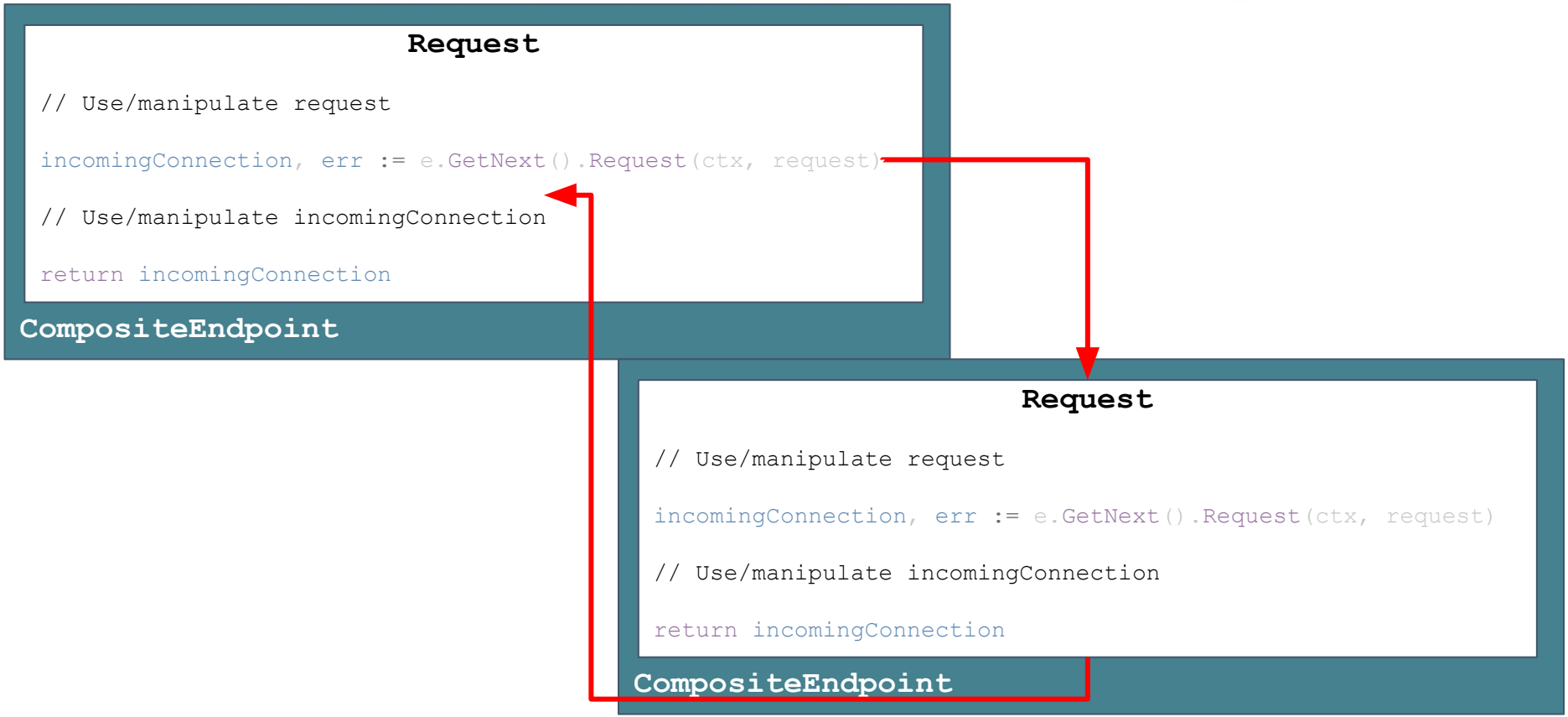


Intro to the SDK - Endpoint Composition


```
import     "github.com/networkservicemesh/networkservicemesh/pkg/tools"
import     "github.com/networkservicemesh/networkservicemesh/sdk/endpoint"

func main() {

    composite := endpoint.NewCompositeEndpoint(
        endpoint.NewMonitorEndpoint(nil),
        newVppAgentAclComposite(nil),
        newVppAgentXConnComposite(nil),
        endpoint.NewClientEndpoint(nil),
        endpoint.NewConnectionEndpoint(nil))

    nsmEndpoint, err := endpoint.NewNSMEndpoint(nil, nil, composite)


    nsmEndpoint.Start()


    // Capture signals to cleanup before exiting
    nsmEndpoint.Delete()
}
```



Intro to the SDK - Endpoint Composition



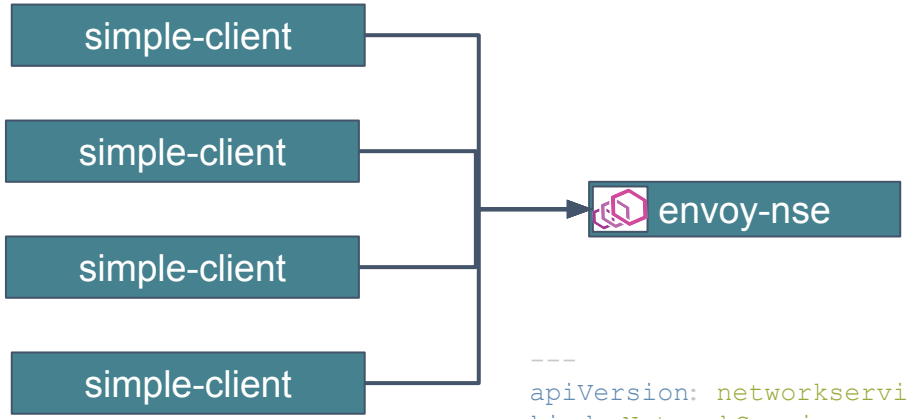
Intro to the SDK - Pre-defined composables

- `networkservicemesh/networkservicemesh/sdk` comes with a number of composables
  - **client** - a NS client. Useful for composing NS Endpoints
  - **connection** - base `connection` structure fill-in
  - **ipam** - Simple IP address management
  - **monitor** - connection monitoring mechanism binding

Network Service Mesh - examples

- The `networkservicemesh/examples` repo hosts 5 examples:
  - **Simple ICMP responder** - 4 Clients and 2 Endpoints over kernel interfaces
  - **VPP ICMP responder** - 4 clients and 2 endpoints over shared memory interfaces
  - **Envoy Interceptor** - Envoy proxy as Network Service; inspired by Istio
  - **Proxy** - a reverse proxy which can serve as an HTTP gateway to NSM defined service
  - **Secure intranet** - a slightly more complicated Sarah story, composed of 5 endpoints NS

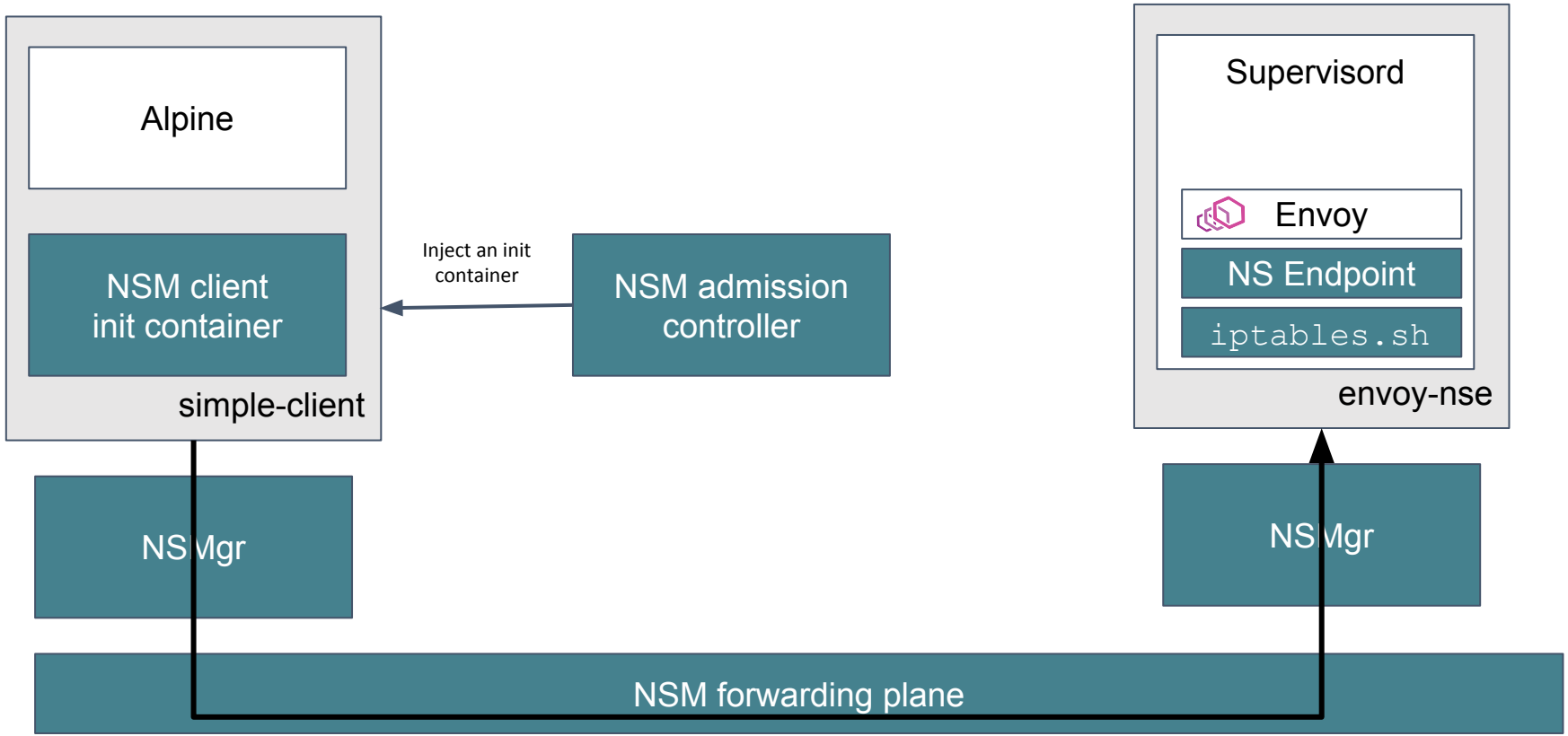
Network Service Mesh example - Envoy interceptor




```
---
apiVersion: networkservicemesh.io/v1
kind: NetworkService
metadata:
 name: web-service
spec:
 payload: IP
 matches:
   - match:
     route:
       - destination:
         destinationSelector:
           app: nginx-nse
```

```
---
admin:
 access_log_path: /dev/null
 address:
   socket_address:
     address: 0.0.0.0
     port_value: 8081
static_resources:
 clusters:
   name: cluster_0
   connect_timeout: "0.25s"
   load_assignment:
     cluster_name: cluster_0
     endpoints:
       - lb_endpoints:
           - endpoint:
               address:
                 socket_address:
                   address: 0.0.0.0
                   port_value: 8080
 listeners:
   name: listener_0
   address:
     socket_address:
       address: 0.0.0.0
       port_value: 8080
   filter_chains:
     - filters:
         - name: envoy.echo
           config:
```

Network Service Mesh example - Envoy interceptor



Thank You

https://networkservicemesh.io/community/ (slack, community calls)

Wednesday 14:30 - 15:30 - NSM Meetup at the Hub Lounge

Thursday 10:30 - 12:30 - Nikolay at booth D2

Frederick Kautz - fkautz@gmail.com

Nikolay Nikolaev - nnikolay@vmware.com


KubeCon
CloudNativeCon
Europe 2019
Network Service Mesh