
OpenCredo
A TRIFORK COMPANY

OpenCredo
A TRIFORK COMPANY

Upgrading your service mesh to Linkerd 2

Lessons learned

Tilen Faganel

Senior Consultant
@ OpenCredo

You can find me on WeChat via @tfaganel and
twitter via @TilenFaganel

Agenda

A brief intro

Evolution of service meshes and Linkerd

Redesign and upgrade

Lessons learned

“Service mesh is an approach and a dedicated infrastructure layer for operating a secure, fast and reliable microservices ecosystem.”

A new paradigm

Low-latency infrastructure layer

Layer 7 network exclusively for applications

A way to increase the observability, resilience and security

Extension to Kubernetes

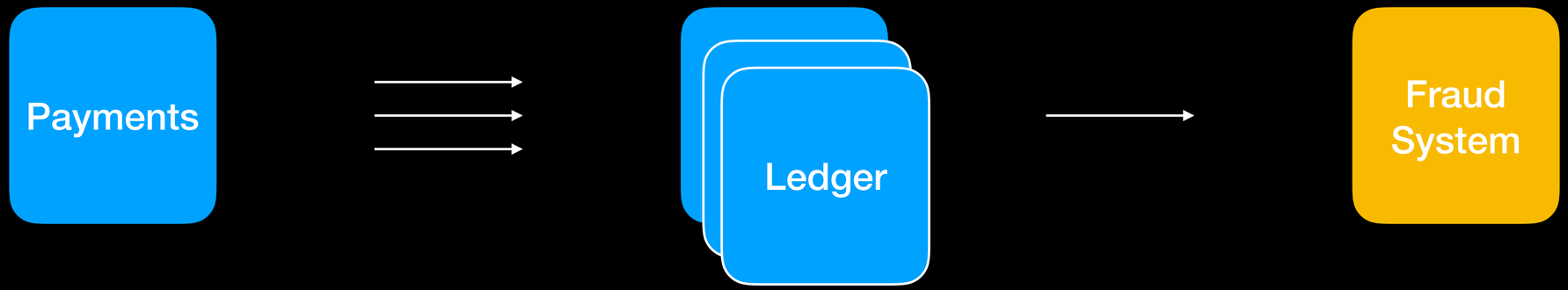
Magic?

Software defined networking

@tfaganel @TilenFaganel

An example

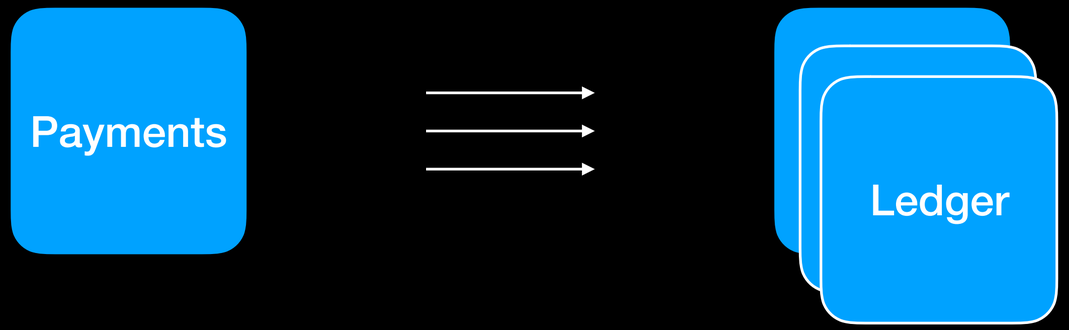
An example



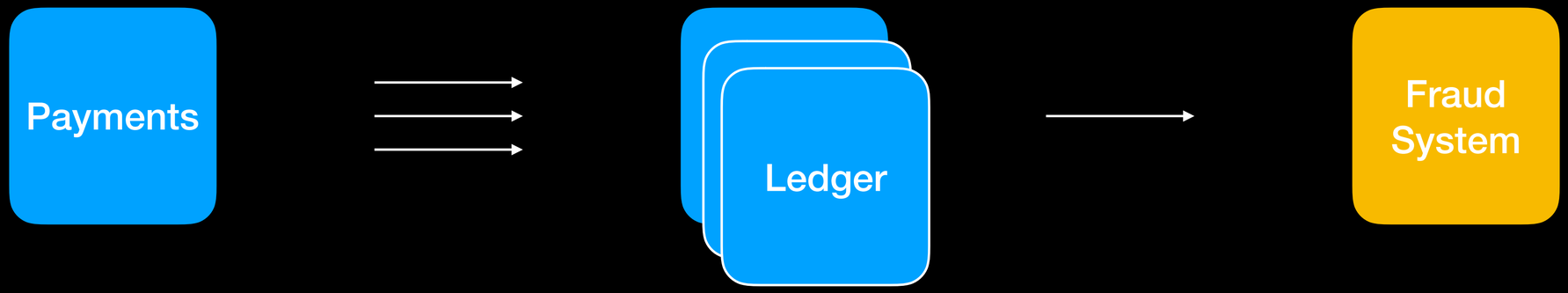
An example

An example

- Service discovery
- Load balancing
- Circuit Breaking
- Retries
- Authentication & Authorization
- Automatic

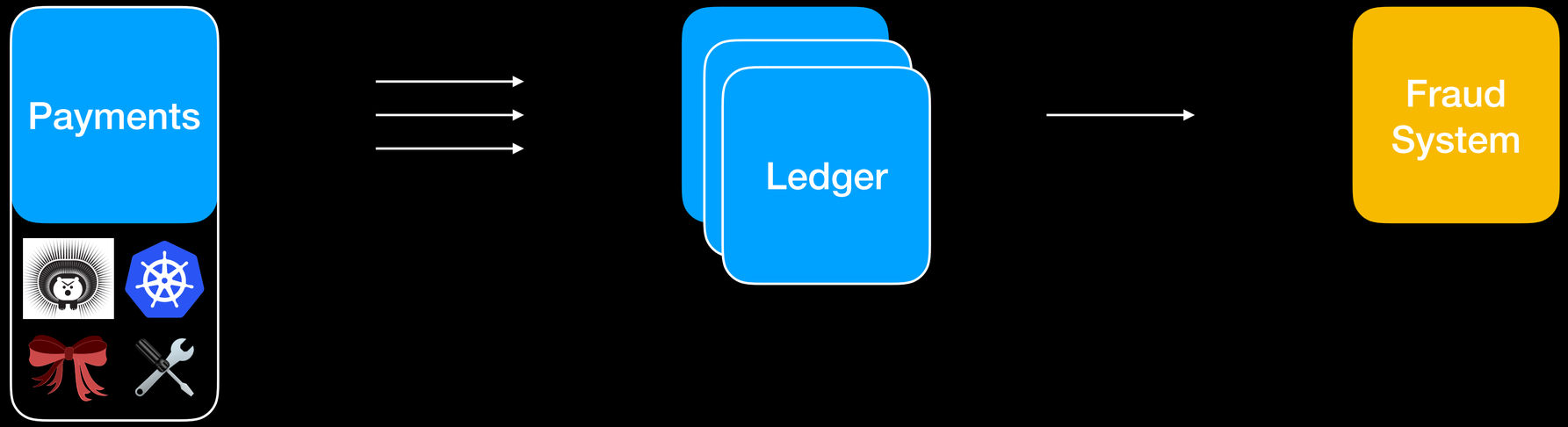


An example

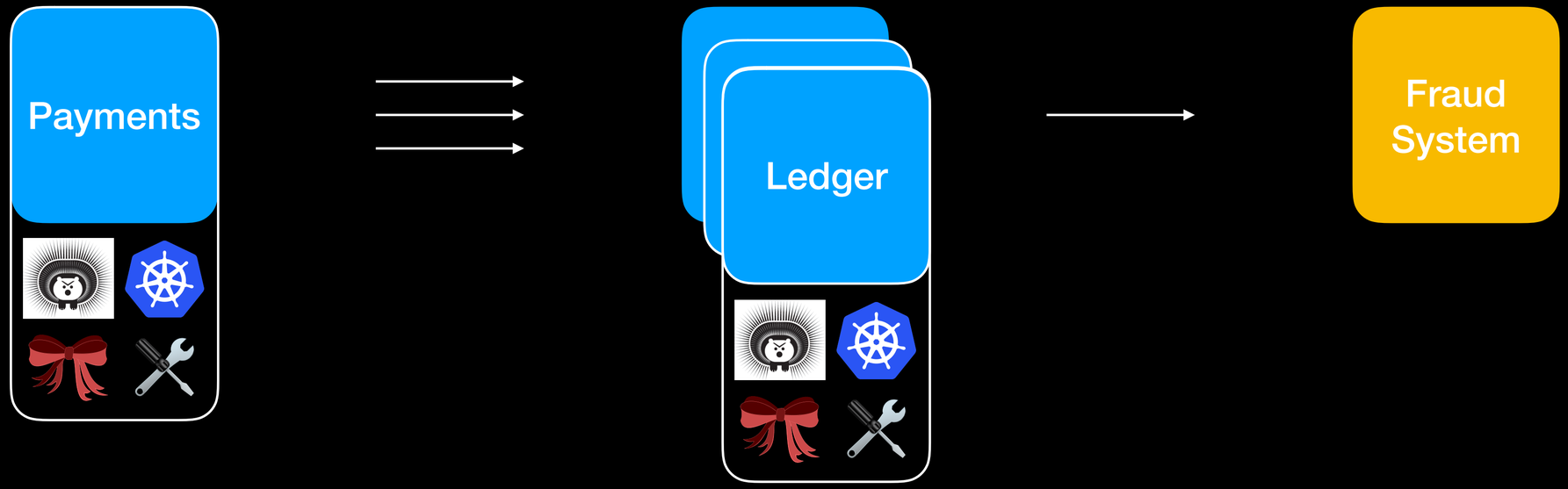


Resiliency

An example



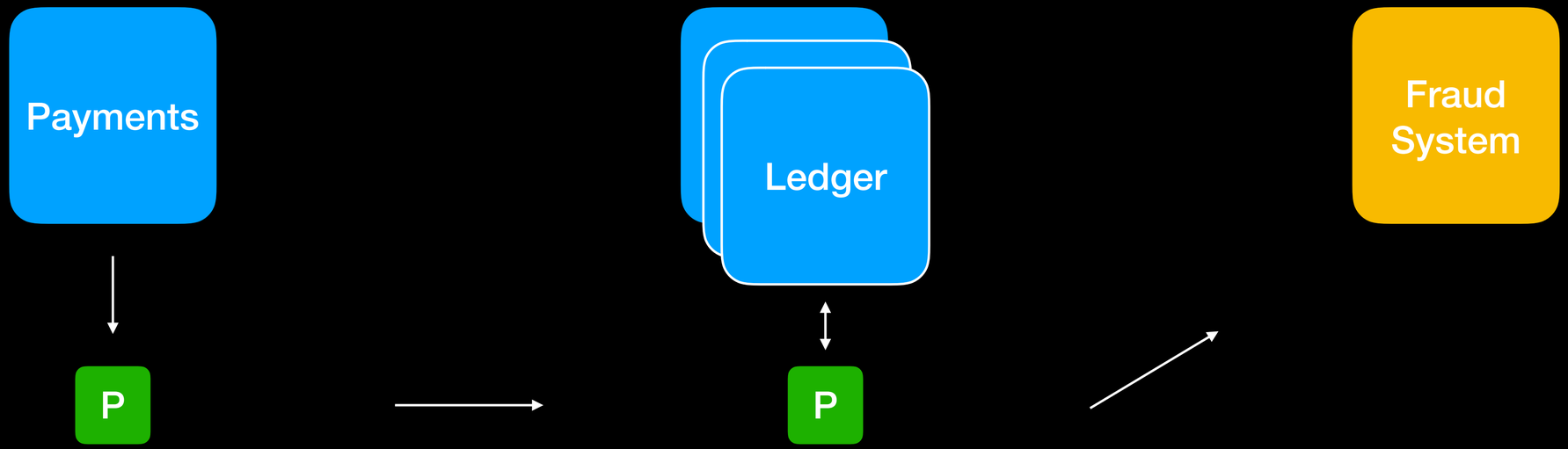
An example



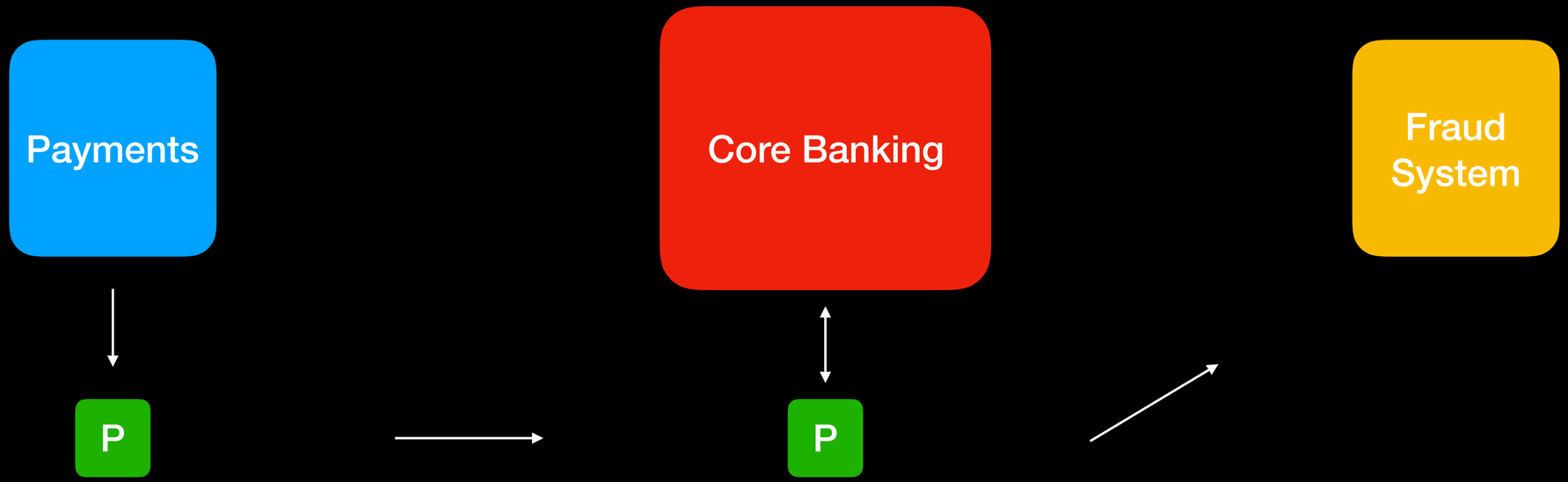
An example

An example

An example

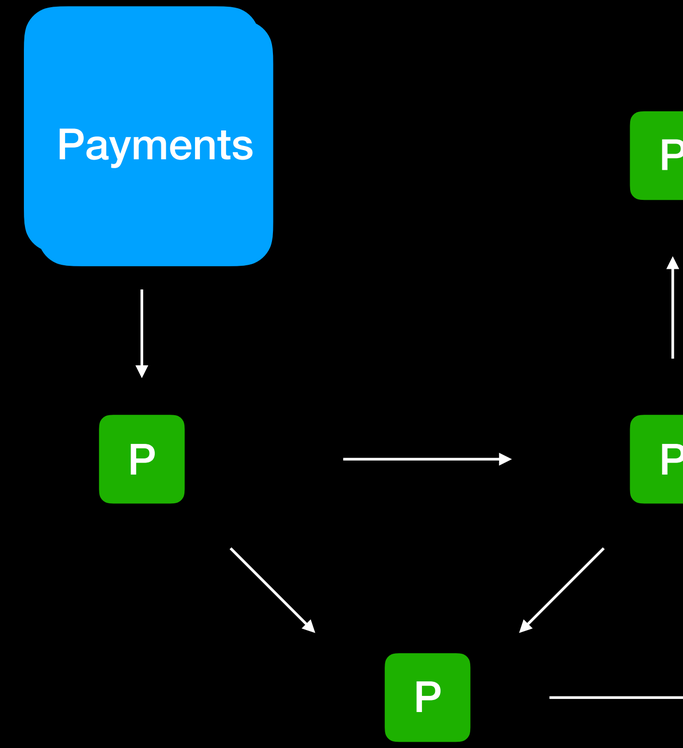


An example



Service mesh

- Service discovery
- Load balancing
- Circuit Breaking
- Retries
- Authentication & Authorization
- Automatic



A collective of smart configurable autonomous proxies

@tfaganel @TilenFaganel

Which proxy to use?

Linkerd

Istio

Consul

Envoy

App Mesh

Kong

Linkerd

Linkerd

- Single all-in-one network proxy - Finagle
- Runs in the JVM
- Routing policies
- Supports most resiliency requirements
- Generic pluggable design
- Single config

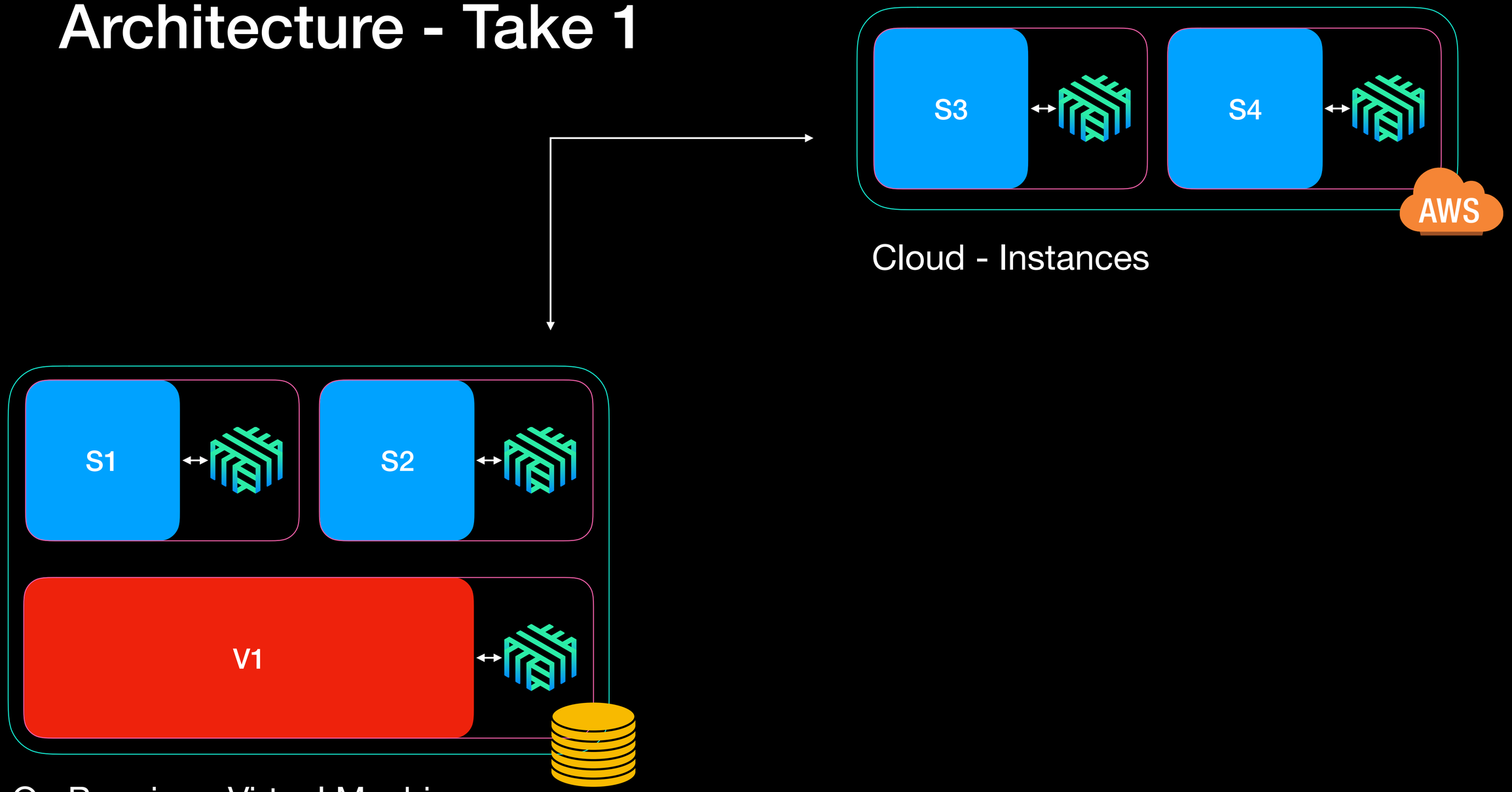
```
dtab: |
  /ph  => /$/io.buoyant.rinet ;
  /svc => /ph/443 ;
  /svc/169.254.169.254 => /ph/80/169.254.169.254 ;
  /svc => /$/io.buoyant.porthostPfx/ph ;
  /k8s => /#/k8sIncoming ;
  /portNsSvc => /#/portNsSvcToK8s ;

  /host => /$/io.buoyant.porthostPfx/portNsSvc ;
  /host => /portNsSvc/http/default ;
  /host => /portNsSvc/http ;

  /svc => /$/io.buoyant.hostportPfx/legacy ;
  /svc => /$/io.buoyant.http.domainToPathPfx/host ;
```

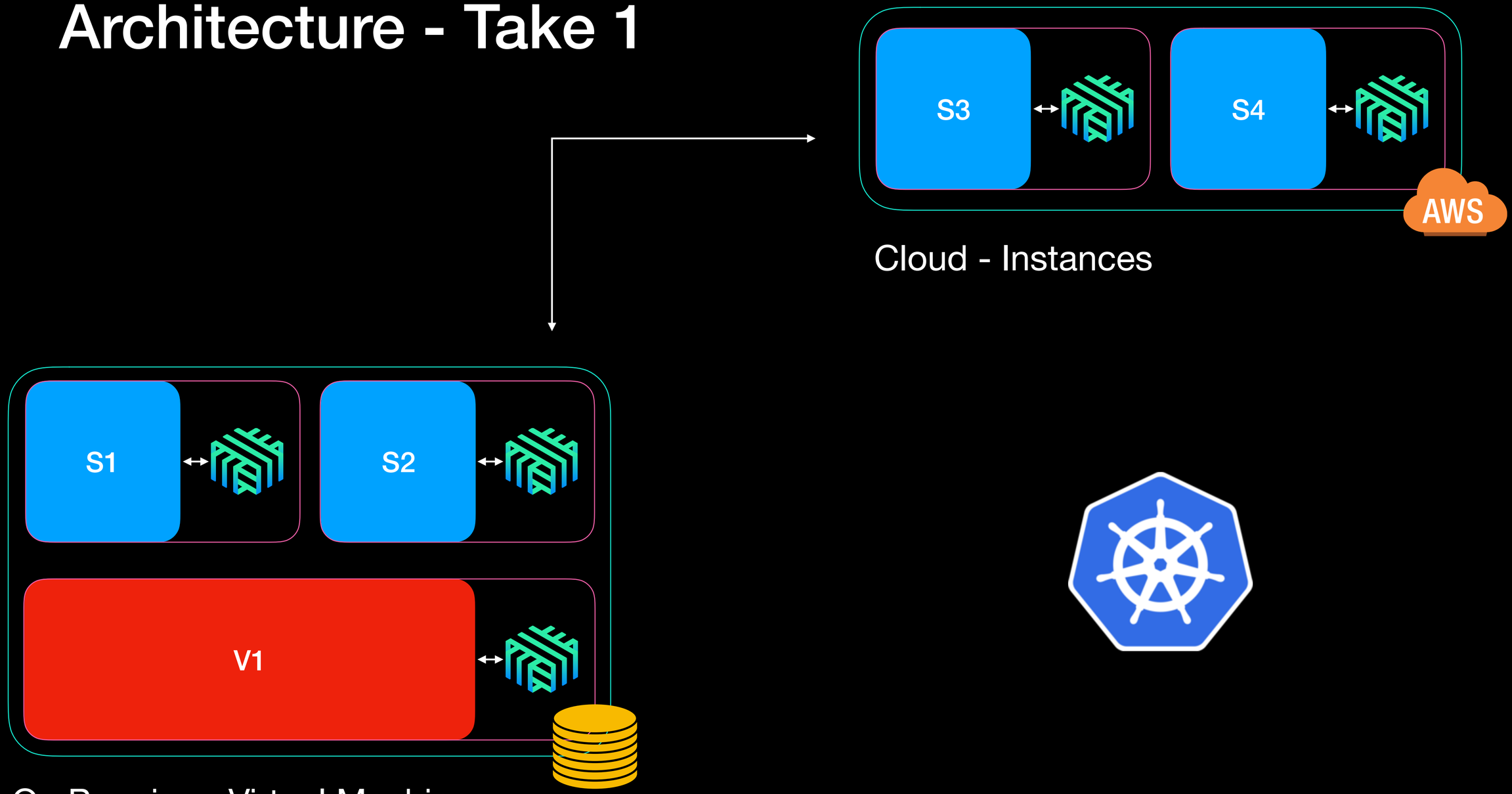
```
service:
  responseClassifier:
    kind: io.l5d.http.retryable5XX
  retries:
    budget:
      minRetriesPerSec: 10
      percentCanRetry: 0.5
      ttlSecs: 15
    backoff:
      kind: jittered
      minMs: 10
      maxMs: 2000
client:
  kind: io.l5d.static
  configs:
  # Use HTTPS if sending to port 443
  - prefix: "/$/io.buoyant.rinet/443/{service}"
    tls:
      commonName: "{service}"
      disableValidation: true
  # Use HTTPS if sending to remote cluster linkerd
  - prefix: "/%/io.l5d.k8s.daemonset"
    loadBalancer:
      kind: ewma
      maxEffort: 10
      decayTimeMs: 30000
```

Architecture - Take 1



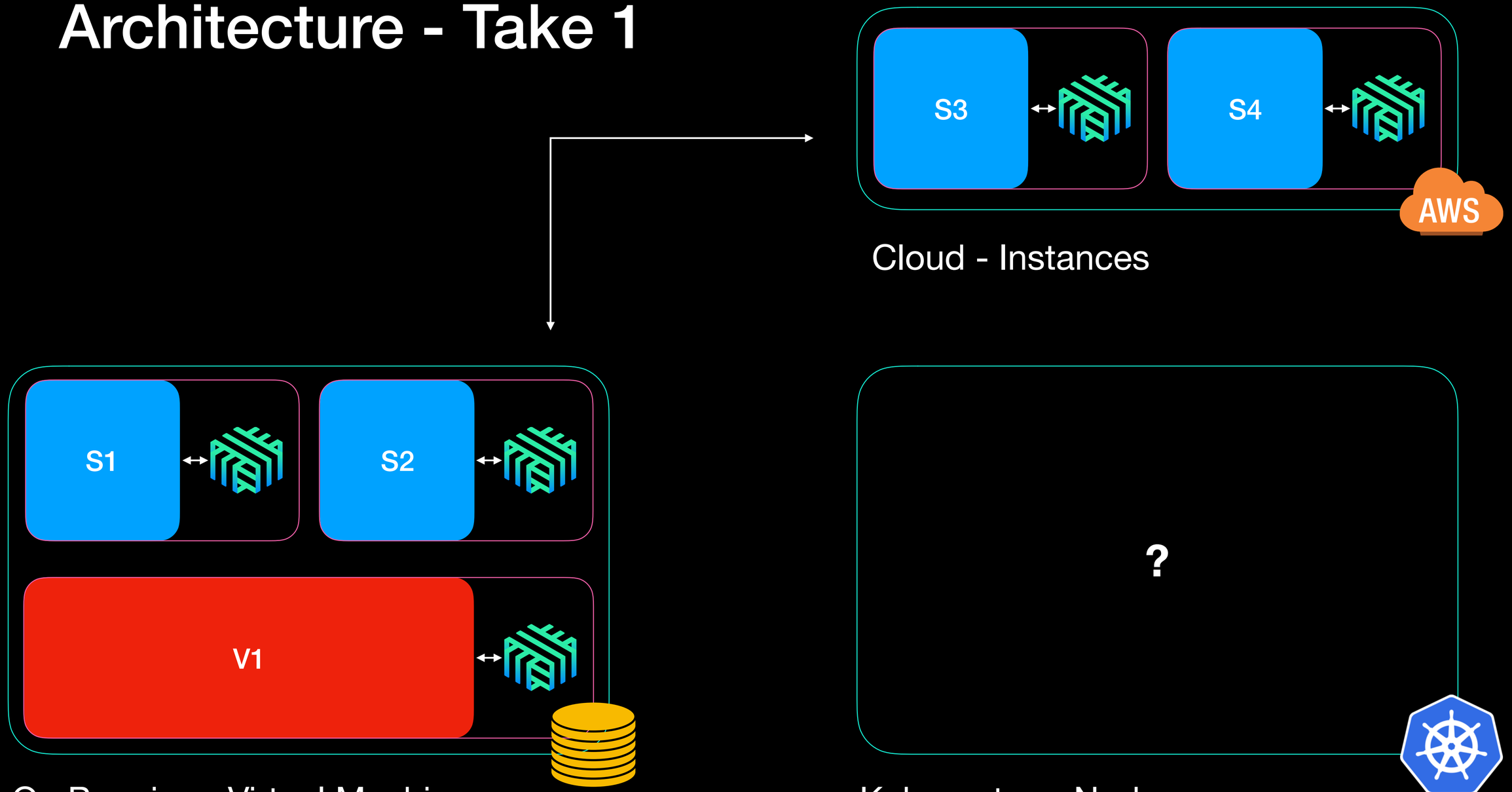
@tfaganel @TilenFaganel

Architecture - Take 1



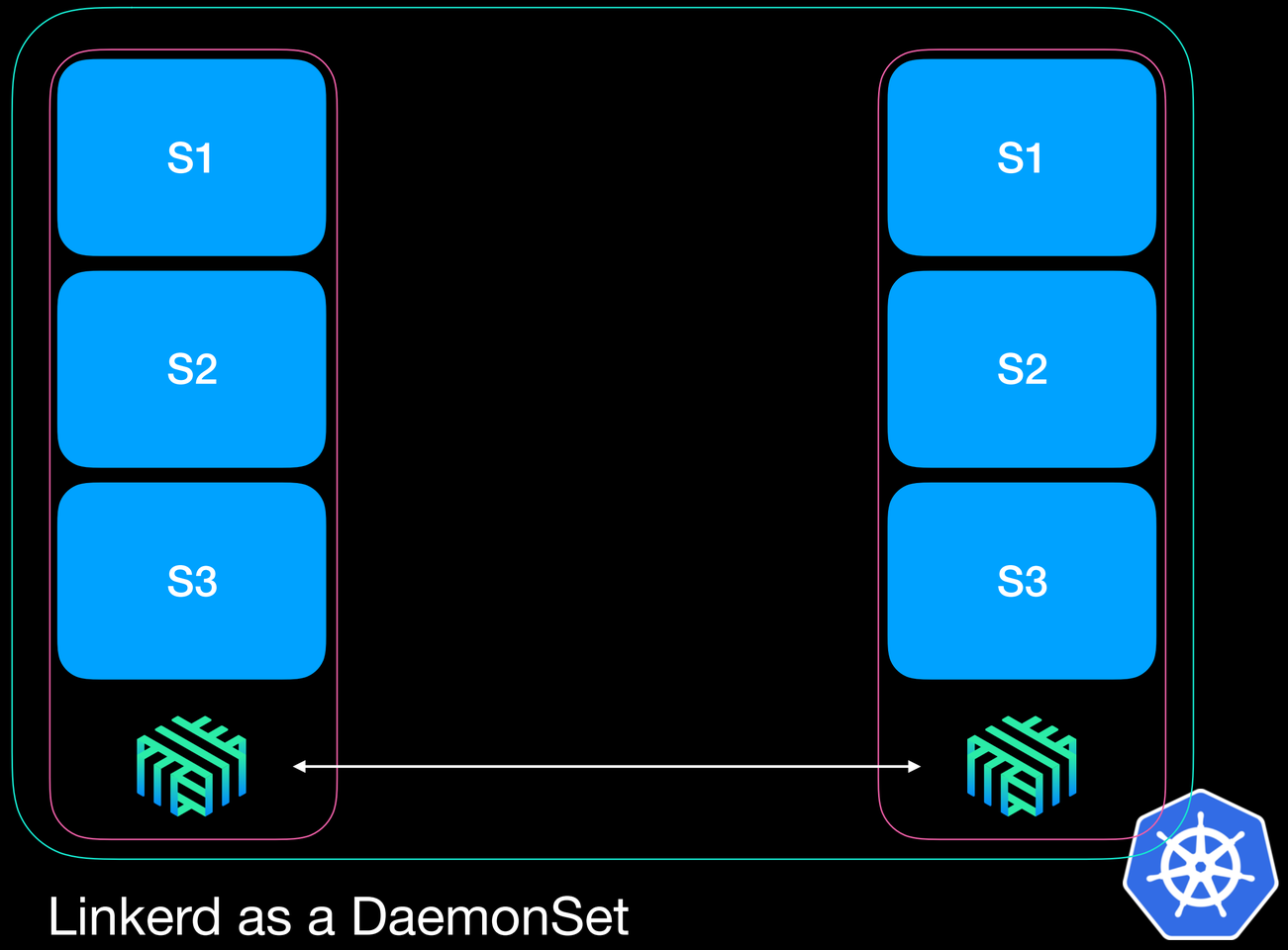
@tfaganel @TilenFaganel

Architecture - Take 1



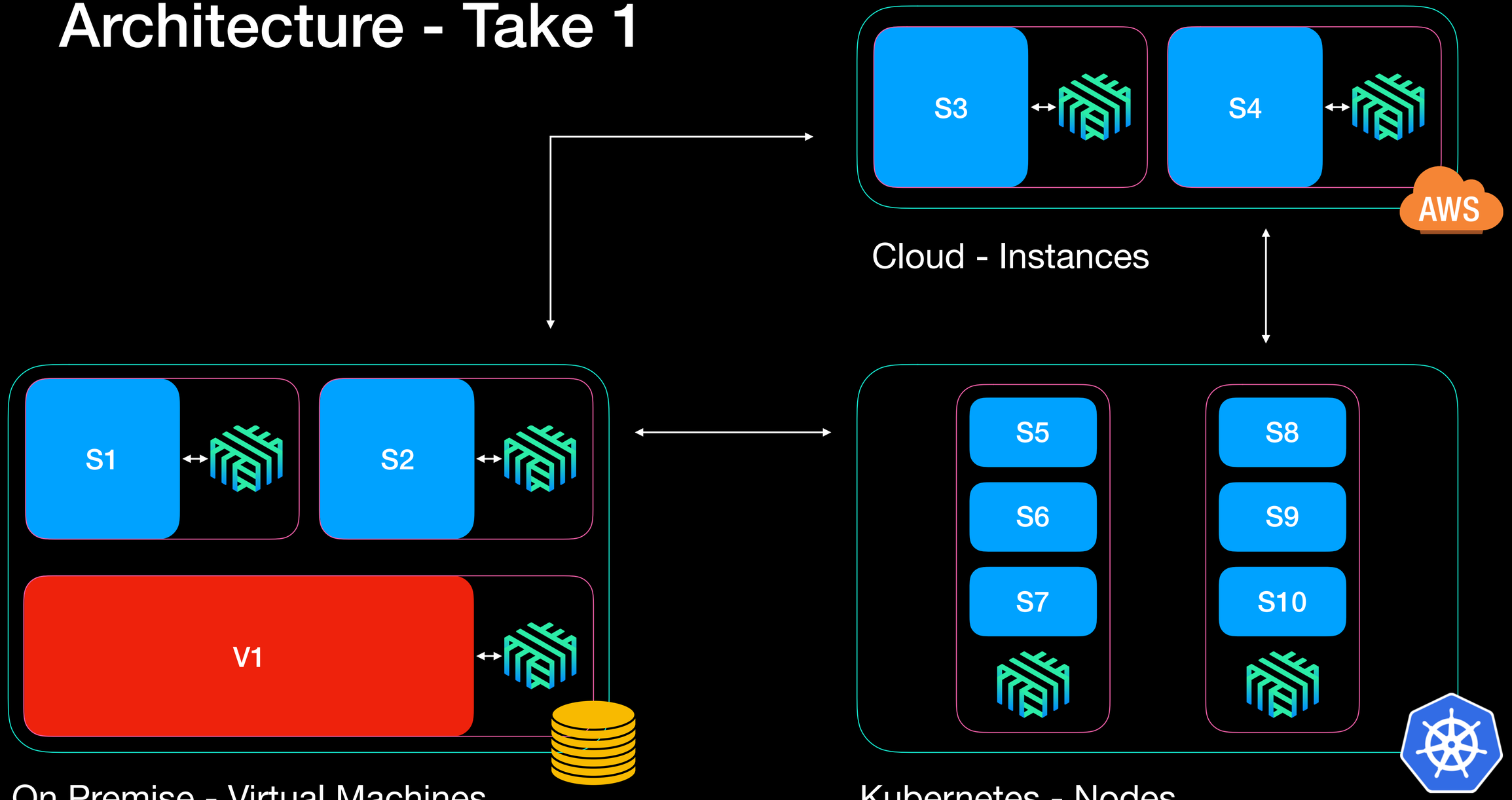
@tfaganel @TilenFaganel

Architecture - Take 1



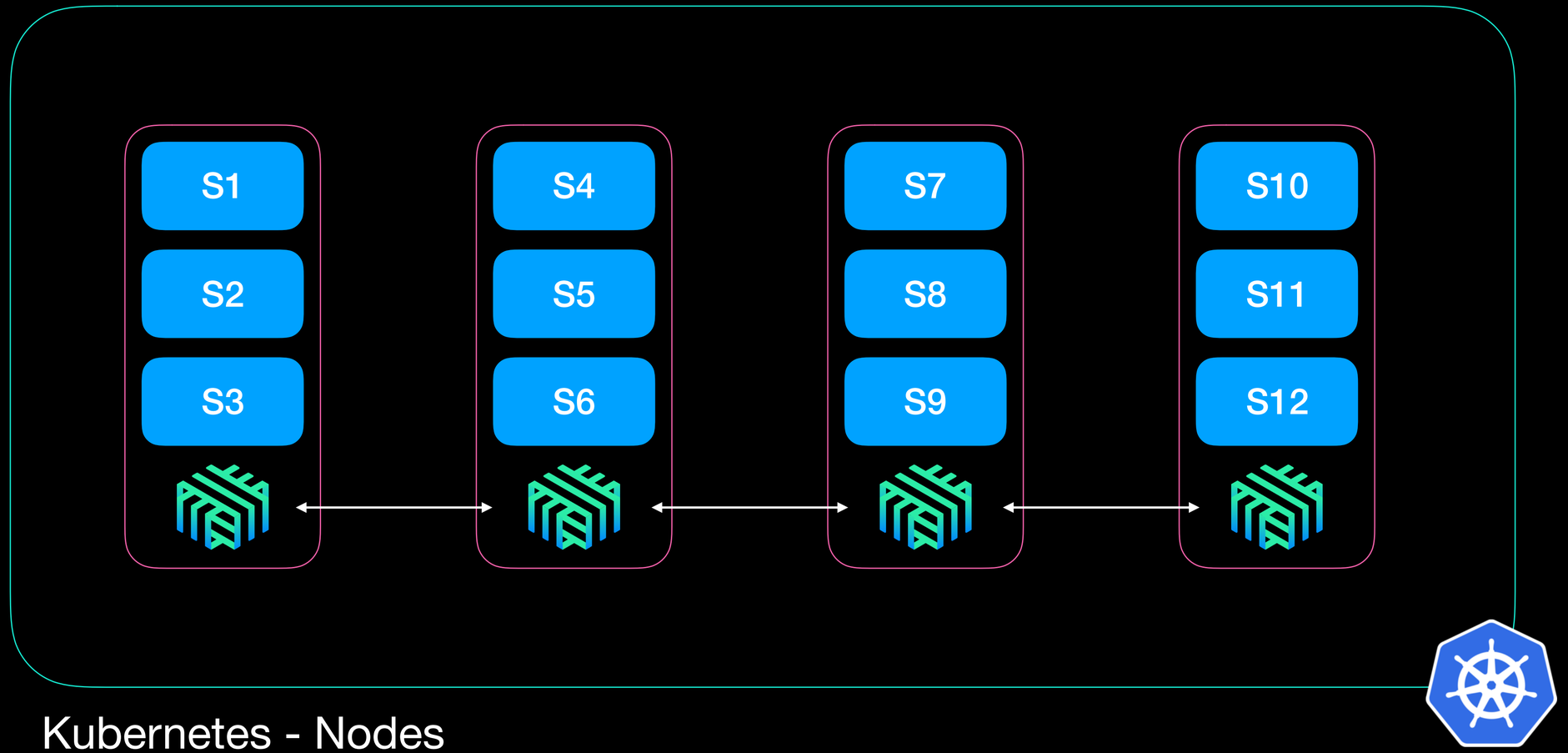
Linkerd as a DaemonSet

Architecture - Take 1



@tfaganel @TilenFaganel

Architecture - Take 2

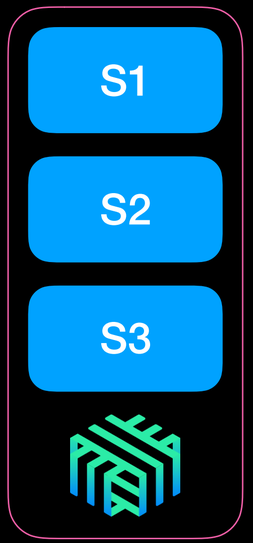



localhost
linkerd
dtab
logging
requests
help
http-incoming
version 1.6.1
ROUTER
http-incoming
Pending
0
Requests
7
Retries
0
Success rate
100.00%
Fail rate
0.00%
Requests per client
102
50
0
11:09:00
11:09:10
11:09:20
11:09:30
11:09:40
11:09:50
11:10:00
11:10:10
Retry budget available: 20% / 20%
CLIENTS
expand all · collapse all
/#/io.l5d.k8s.http/default/http/payments
collapse
Successes
7 100.00%
Failures
0
Requests
7
Connections
2
Latencies
max 14 ms
p9990 9 ms
p99 7 ms
p95 3 ms
p50 1 ms
Client success rate
100.00
99.99
11:09:30
11:09:40
11:09:50
11:10:00
11:10:10
Endpoints available: 1 / 1


localhost
Service Mesh Overview
Last 15 minutes
Global Success Rate
98.0%
Ingress Success Rate
100%
Internal Success Rate
96.5%
Global Request Volume
1.6 rps
Global Request Errors
0 rps
Ingress Request Volume
0.7 rps
Ingress Request Errors
0 rps
Internal Request Volume
0.9 rps
Internal Request Errors
0 rps
Request Volumes Per Service
150 rps
100 rps
50 rps
0 rps
10:58
10:59
11:00
11:01
11:02
11:03
11:04
11:05
11:06
11:07
11:08
11:09
11:10
11:11
11:12
Request Errors Per Service
0.15 rps
0.10 rps
0.05 rps

Problems

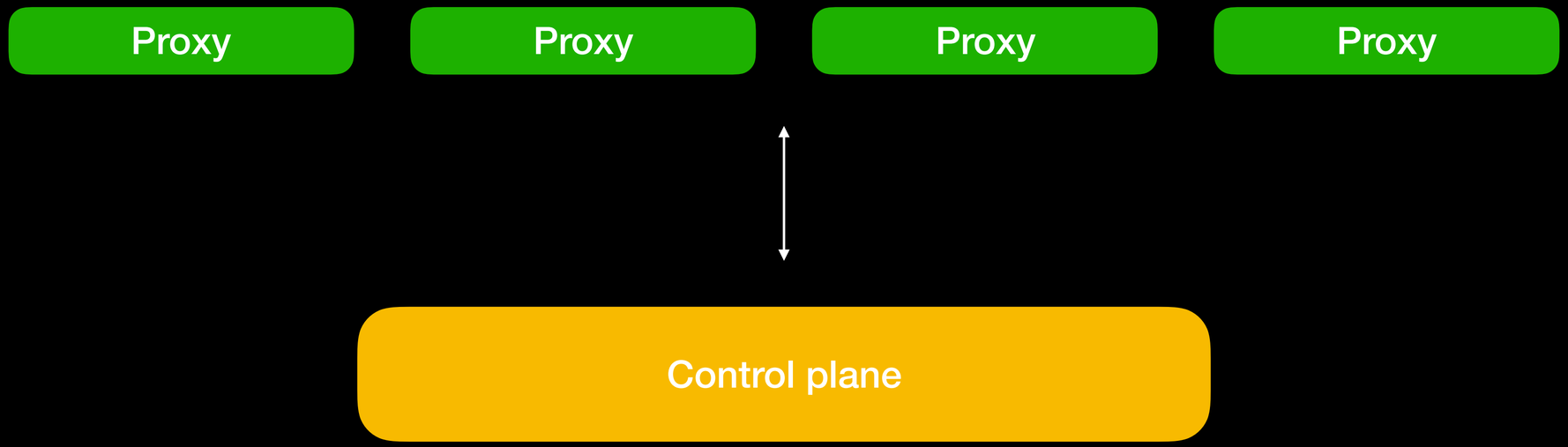
- Siloed instances on every node
- Large resource consumption
- Proxy per node instead of instance
- Complex configuration and its updates
- Disjointed monitoring
- No proper mTLS support
- High developer friction



What’s next?

Proxy

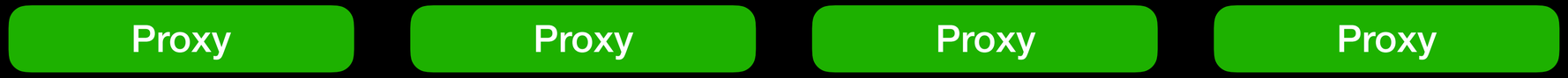
What’s next?



Control plane

- Manages and configures the proxies
- Standard stateless deployment
- Public API
- Collects & exposes telemetry
- Enforces policies
- Issues certificates
- Fully cloud-native

Data plane



1. **Proxy requests**

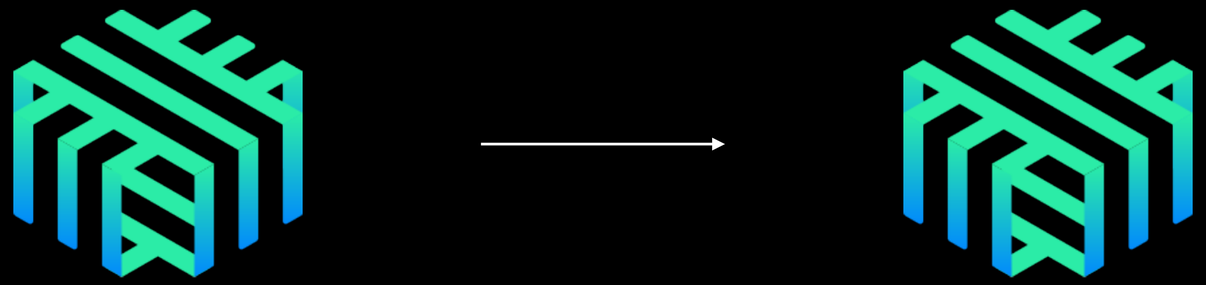
Linkerd 2

Istio

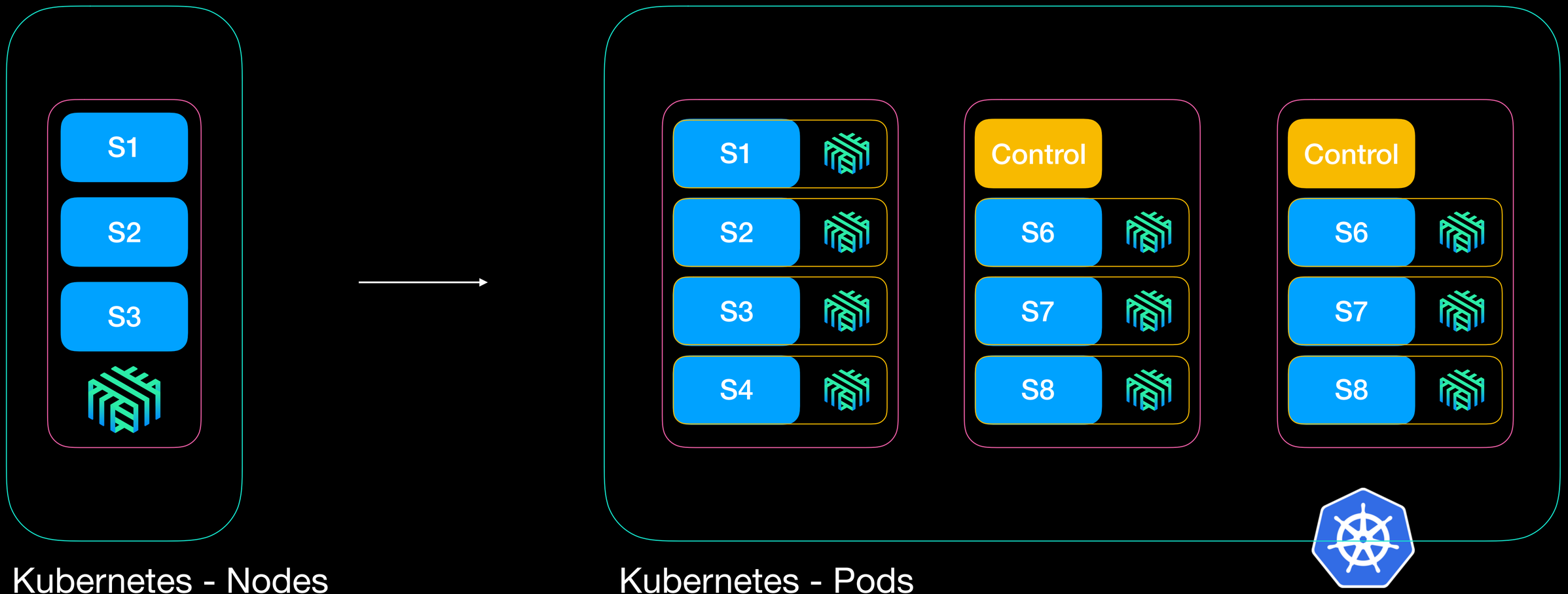
Linkerd 2

@tfaganel @TilenFaganel

Let’s plan a migration



Architecture - Take 3



```yaml
apiVersion: linkerd.io/v1alpha1
kind: ServiceProfile
metadata:
  name: ledger.default.svc.cluster.local
  namespace: default
spec:
  routes:
  - condition:
      method: GET
      pathRegex: /transactions
    name: GET /transactions
    isRetryable: true
  - condition:
      method: POST
      pathRegex: /transactions
    name: POST /transactions
    timeout: 100ms
  - condition:
      method: GET
      pathRegex: /transactions/[^/]*
    name: GET /transactions/{id}
```

@tfaganel @TilenFaganel

Goals

- No required developer involvement or friction
- No disruption
- Minimal changes

```
env:
- name: NODE_NAME
  valueFrom:
    fieldRef:
      fieldPath: spec.nodeName
- name: http_proxy
  value: $(NODE_NAME):4140
```

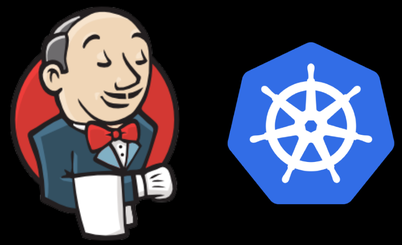
http://ledger.default

@tfaganel @TilenFaganel

Admission Webhooks

- Alter pod definitions before scheduling
- Used by Linkerd 2
- Run in Kubernetes as any deployment
- Completely transparent

The plan



Trigger

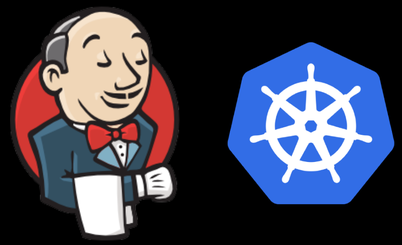
Deploy control plane → Stop old AW → Start new AW → Rolling redeploy → Cleanup

Begin

k8s-infrastructure

@tfaganel @TilenFaganel

The plan



Trigger

Deploy control plane → Stop old AW → Start new AW → Rolling redeploy → Cleanup

Begin

Rollback if needed

k8s-infrastructure

@tfaganel @TilenFaganel

Fully automate your infrastructure from the start

@tfaganel @TilenFaganel


Let the platform do the heavy lifting

“Service mesh provides a transparent, reliable and autonomous network hub for any service.”


OpenCredo
A TRIFORK COMPANY

Demo