Performance tuning and best practices in a Knative based, large-scale serverless platform with Istio

张龚, Gong Zhang, IBM China Development Lab
庄宇, Yu Zhuang, IBM China Development Lab

IstioCon

#IstioCon

Speakers

Gong (Grace) Zhang, zhanggbj@cn.ibm.com, twitter.com/gracezhang1110, www.linkedin.com/in/gong-zhang-75560670/

Advisory Software Engineer of IBM Cloud Code Engine team focusing on Knative Serving and Istio, contributor of the Knative and Cloud Foundry community, maintainer of a Knative benchmarking tool called kperf, speaker of Open Source Summit China 2019 about Istio integration with containerized Cloud Foundry

Yu Zhuang, yuzcdl@cn.ibm.com, https://www.linkedin.com/in/yu-zhuang-51915287/
Architect and Senior Software Engineer in IBM Cloud. Working on IBM Cloud Code Engine (Serverless platform), focusing on Knative, Istio, and Tekton, community, leading team to develop and offer serverless capabilities in IBM Cloud, which based on these Opensource technologies. Before he was architect for Cloud Foundry on Kubernetes in IBM Cloud.

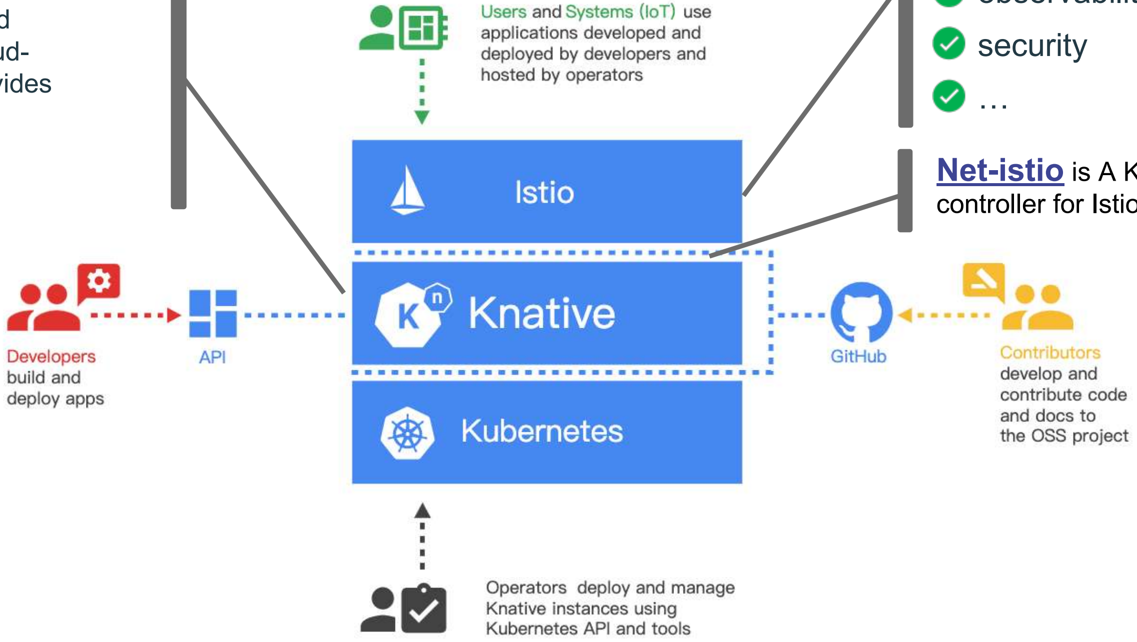
Agenda

- Knative and Istio
- How Istio is leveraged in a Knative based platform
- Performance bottleneck analysis and tuning
    - Istio scalability optimization during Knative Service provisioning
    - Unleash maximum scalability by fully leveraging Istio features in Knative with service mesh enabled
- Reference

Knative and Istio

Knative is an open source project which provides a set of components (Serving and Eventing) that introduce **event-driven** and **serverless** capabilities for Kubernetes clusters for deploying, running, and managing serverless, cloud-native applications. It provides benefits:

- ✓ Focus on code
- ✓ Scale to zero
- ✓ Quick entry to serverless computing
- ✓ … …



Users and Systems (IoT) use applications developed and deployed by developers and hosted by operators
Istio
Developers build and deploy apps
API
Knative
GitHub
Contributors develop and contribute code and docs to the OSS project
Kubernetes
Operators deploy and manage Knative instances using Kubernetes API and tools

Knative design based on knative.dev

Istio is the default networking layer solution of Knative. It is leveraged for

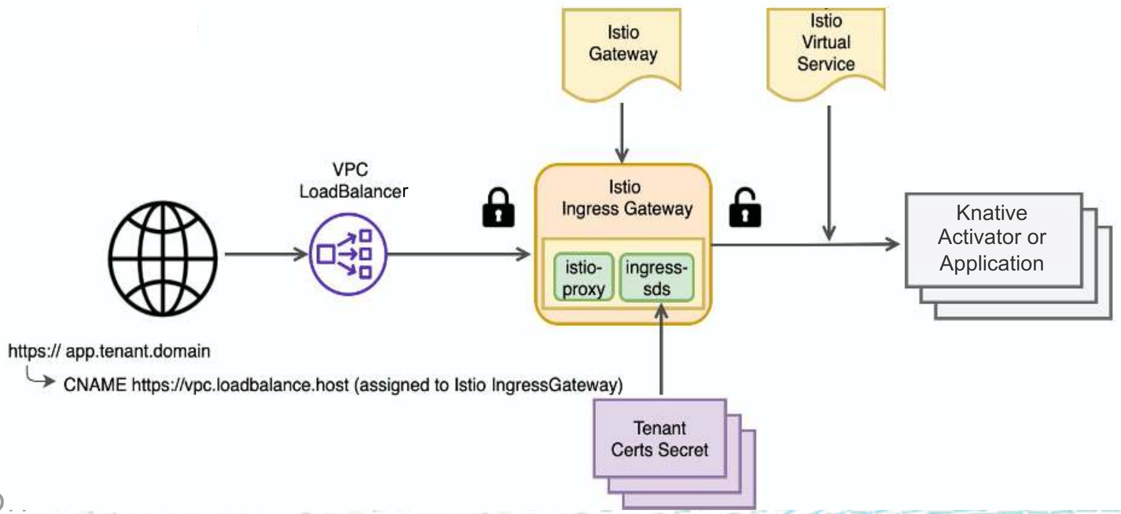
- ✓ traffic management
- ✓ observability
- ✓ security
- ✓ …

Net-istio is A Knative ingress controller for Istio.

How Istio is leveraged in a Knative based platform

- Istio as an Ingress Gateway

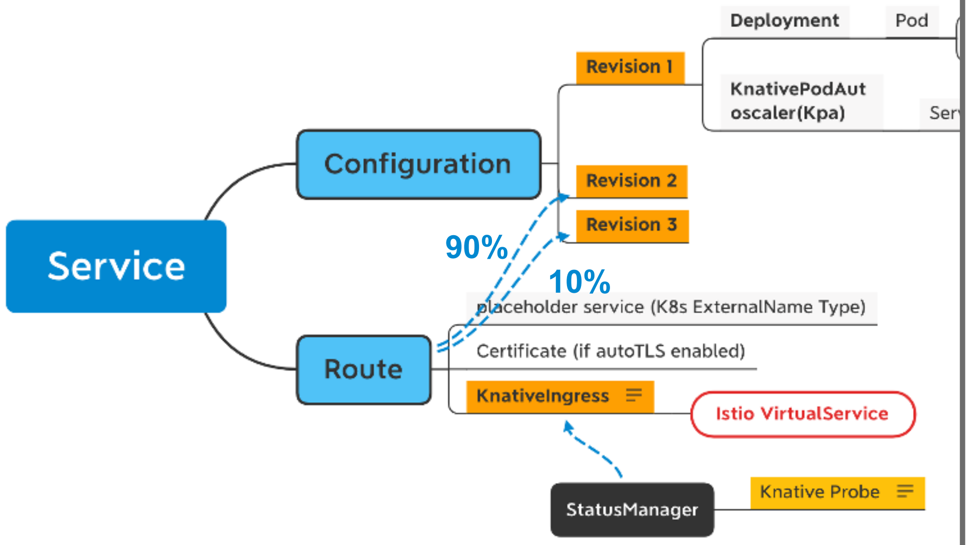
- By default, Knative does **not** enable service mesh, it uses Istio as an Ingress Gateway.
- Enable **Secret Discovery Service** (SDS) to monitor and mount secrets under `istio-system` to ingress gateway which contains credentials for https support of **multi tenants**.
- Knative has `knative-ingress-gateway` for **external** access and `knative-local-gateway` for **cluster local** access. They use Istio gateway service `istio-ingressgateway` as its underlying service.


Istio Gateway
Istio Virtual Service
VPC LoadBalancer
Istio Ingress Gateway
istio-proxy
ingress-sds
Knative Activator or Application
https:// app.tenant.domain
CNAME https://vpc.loadbalance.host (assigned to Istio IngressGateway)
Tenant Certs Secret

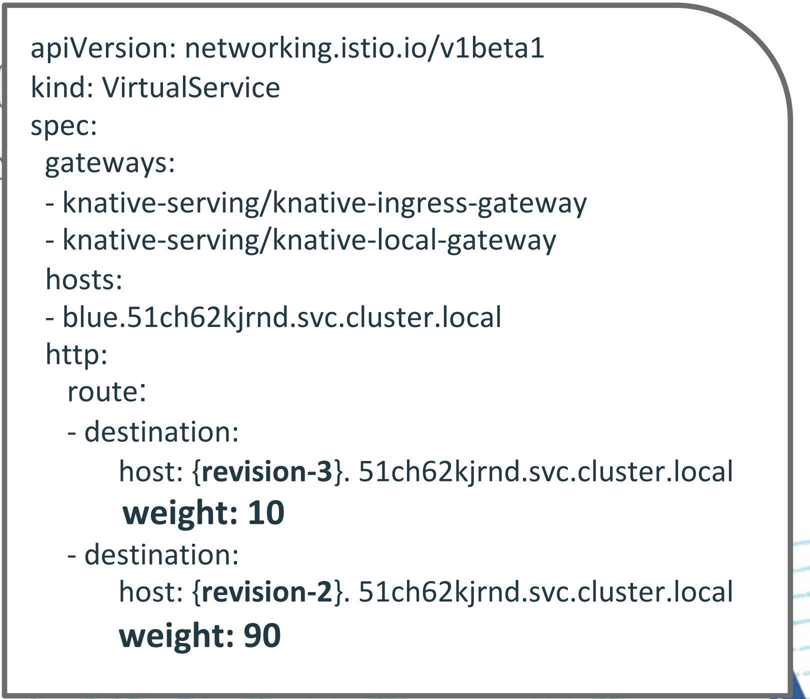
Front door design

How Istio is leveraged in a Knative based platform

- Traffic Splitting, blue/green deployment



Knative Service Inspection



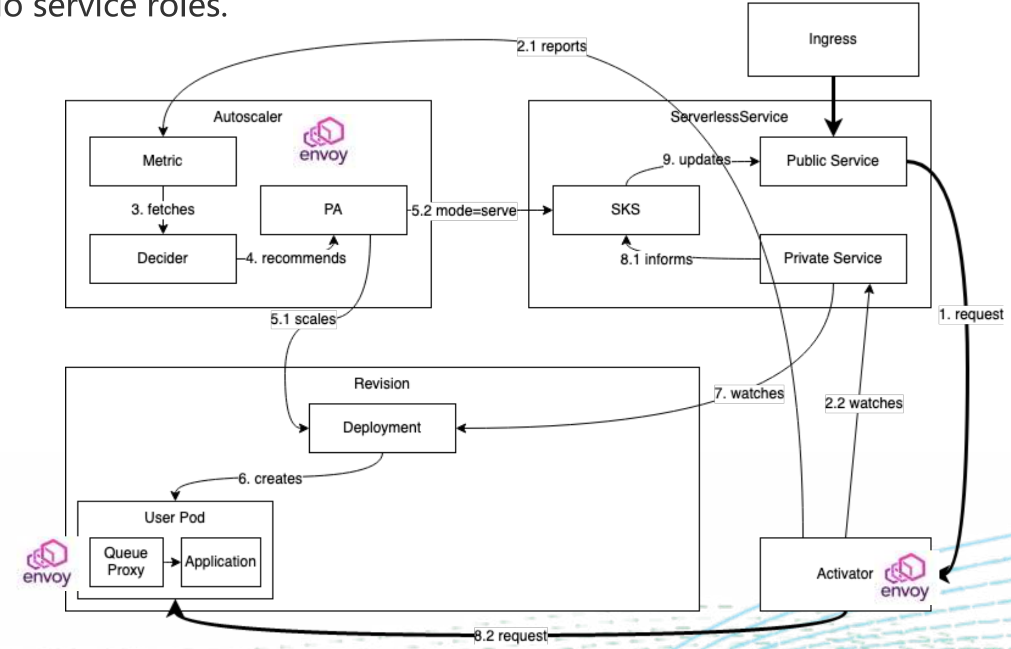
```
apiVersion: networking.istio.io/v1beta1
kind: VirtualService
spec:
 gateways:
 - knative-serving/knative-ingress-gateway
 - knative-serving/knative-local-gateway
 hosts:
 - blue.51ch62kjrnd.svc.cluster.local
 http:
   route:
   - destination:
        host: {revision-3}. 51ch62kjrnd.svc.cluster.local
       weight: 10
   - destination:
        host: {revision-2}. 51ch62kjrnd.svc.cluster.local
        weight: 90
```

#IstioCon

How Istio is leveraged in a Knative based platform

- Security with Service Mesh enabled

- **mutual TLS** is enabled to secure the user application traffic end to end in production
- Allow platform to use Istio authorization policy to control the access to each Knative service based on Istio service roles.



Traffic on Knative with mesh enabled (based on https://github.com/knative/serving)

Performance bottleneck analysis and tuning

Istio scalability optimization during Knative Service provisioning

- Performance Criteria: the platform has multiple shard k8s clusters, each cluster should support 1000 sequential (interval 5s) Knative service provisionings with **route ready time <= 30s**.

Type	Info
K8s Cluster Capacity	12 nodes in 3 zones, 16 vCPU * 64 Gi MEM
Knative Version	Knative 0.16, 0.17, 0.18
Istio Version	1.5, 1.6, 1.7

- Benchmark: **Kperf (**https://github.com/knative-sandbox/kperf**)** is a benchmark tool for Knative which can generate specific Knative Service provisioning workload and provides aggregated data of Knative Service ready duration.
    - Knative Performance Testing Framework 2 Design

Istio scalability optimization during Knative Service provisioning

- **Tune CPU/MEM to ensure enough capacity**

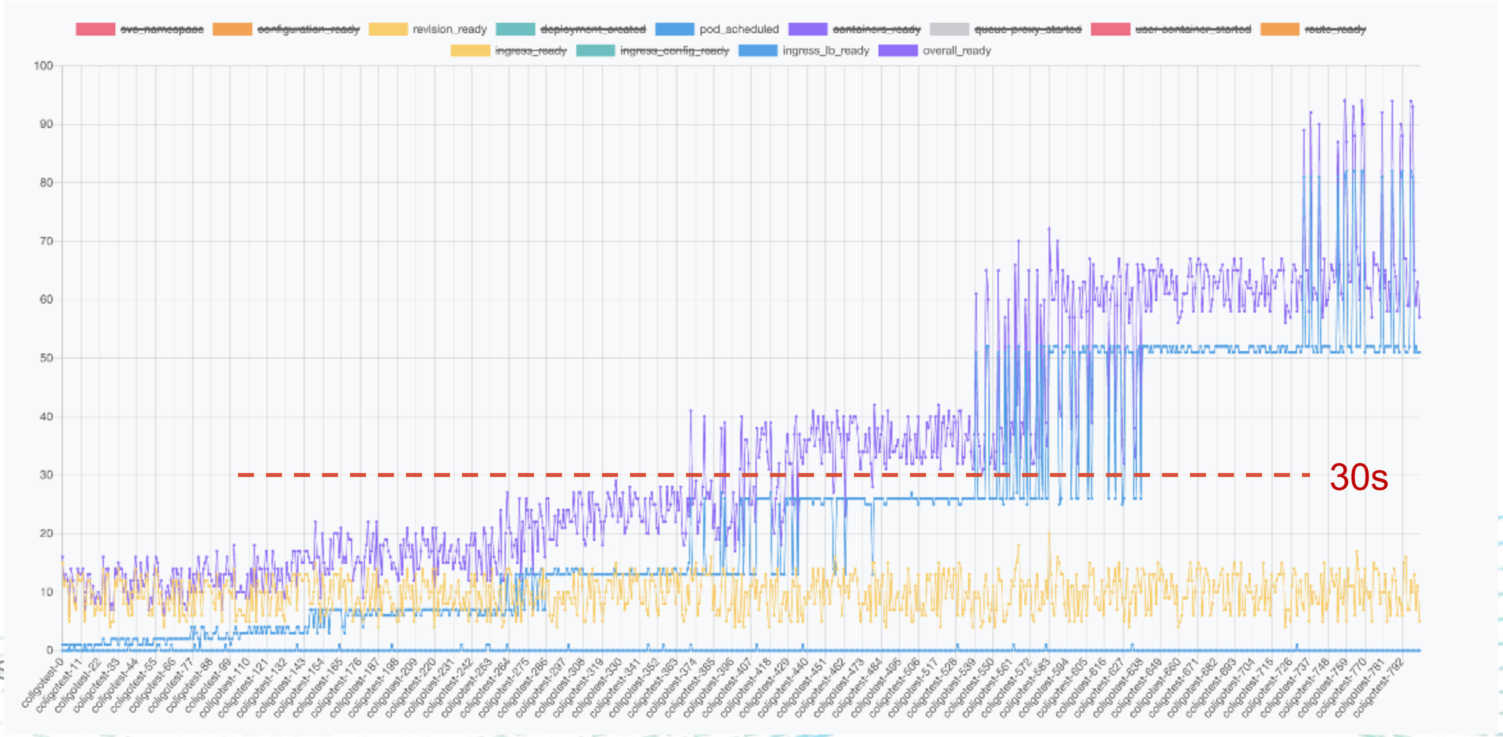
Leveraged Metrics to monitor Istio & Knative components' CPU and MEM under workload to avoid CPU throttling and OOM and ensure enough capacity. In Istio 1.5.4:

- Ingress gateway MEM has linear growth, and it consumes ~=750k for 1 Knative Service (#25145).
  The envoy mem release fix included in Istio 1.6.0+ resolved this issue.
- Istiod MEM bumped with large numbers of Knative Services (#25532)
  Mem usage optimization of pilot resolved this issue.

Project	**Component**	**CPU**		**MEM**		**HorizontalPodAutoscaler (HPA)**
		request	limit	request	limit	
Istio (1.7.3)	istio-ingressgateway	2 vCPU	2 vCPU	2 Gi	4 Gi	Y, min=3, max=20
	Istiod	1 vCPU	1 vCPU	2 Gi	4 Gi	Y, min=3, max=6
Knative	Networking-istio	30m	80Mi	900m	2 Gi	N

Istio scalability optimization during Knative Service provisioning

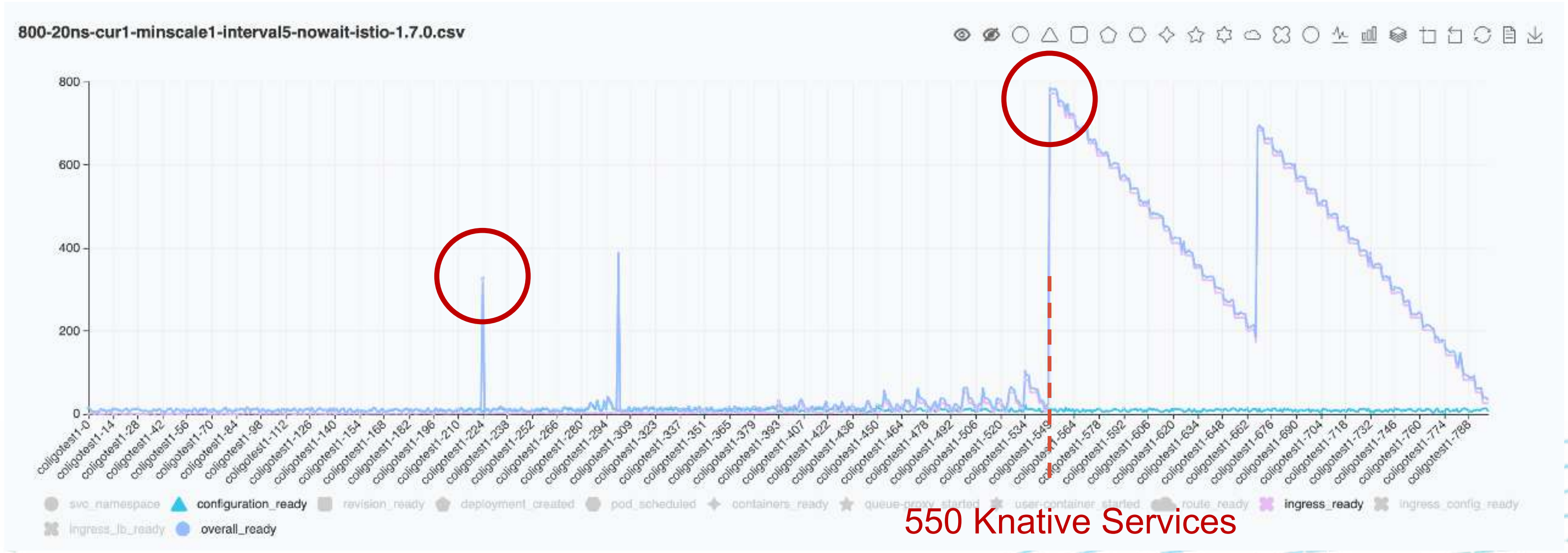
- **Detect and analyze Istio scalability issue**
  - **Ingress_lb_ready** is the duration from Knative Ingress and istio VirtualService are created to Knative probe thinks the configuration works.
  - [**Istio 1.5.4**] Istio is picking up new VirtualService slowly



Istio scalability optimization during Knative Service provisioning

- **Detect and analyze Istio scalability issue**

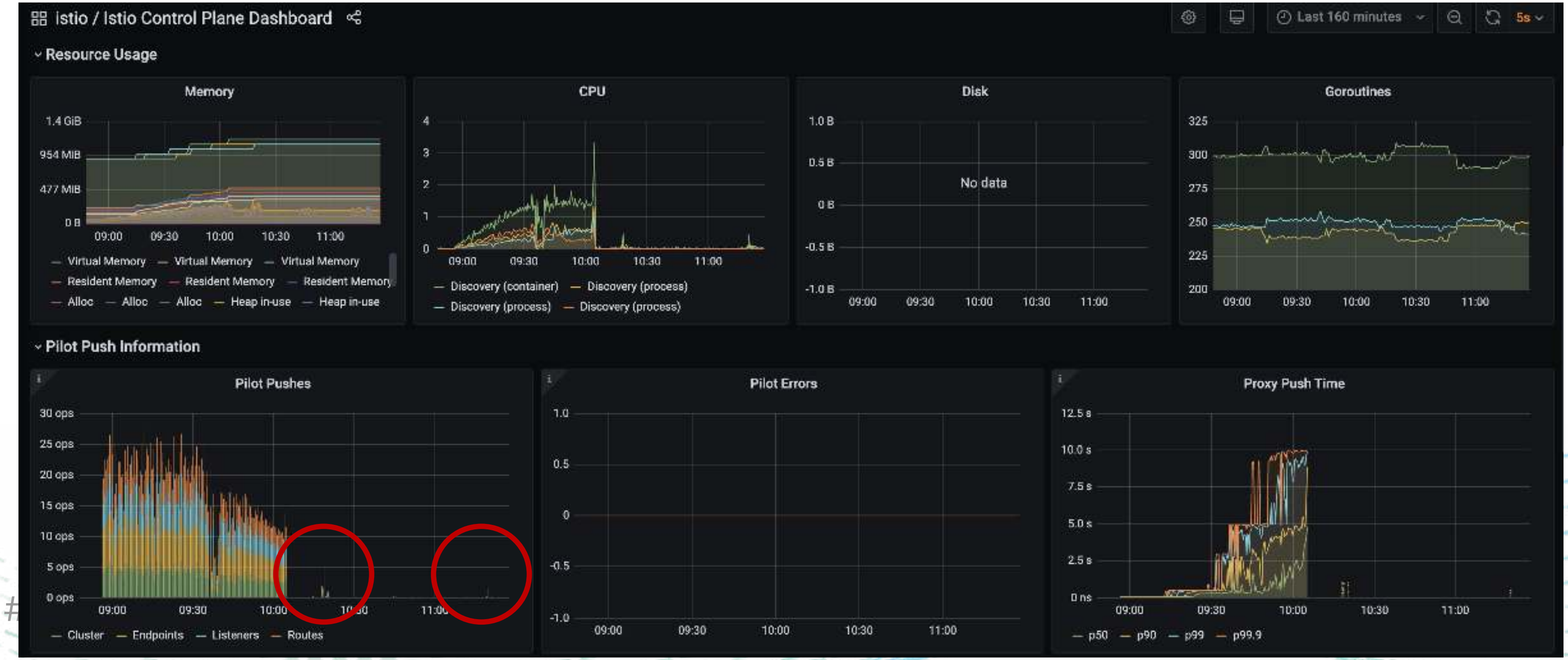
[Istio 1.6.5&1.7.0] There're two main issues
- ingress_ready has random peak values
- ingress_ready bumped to ~=800 seconds with 500+ Knative Services



#IstioCon

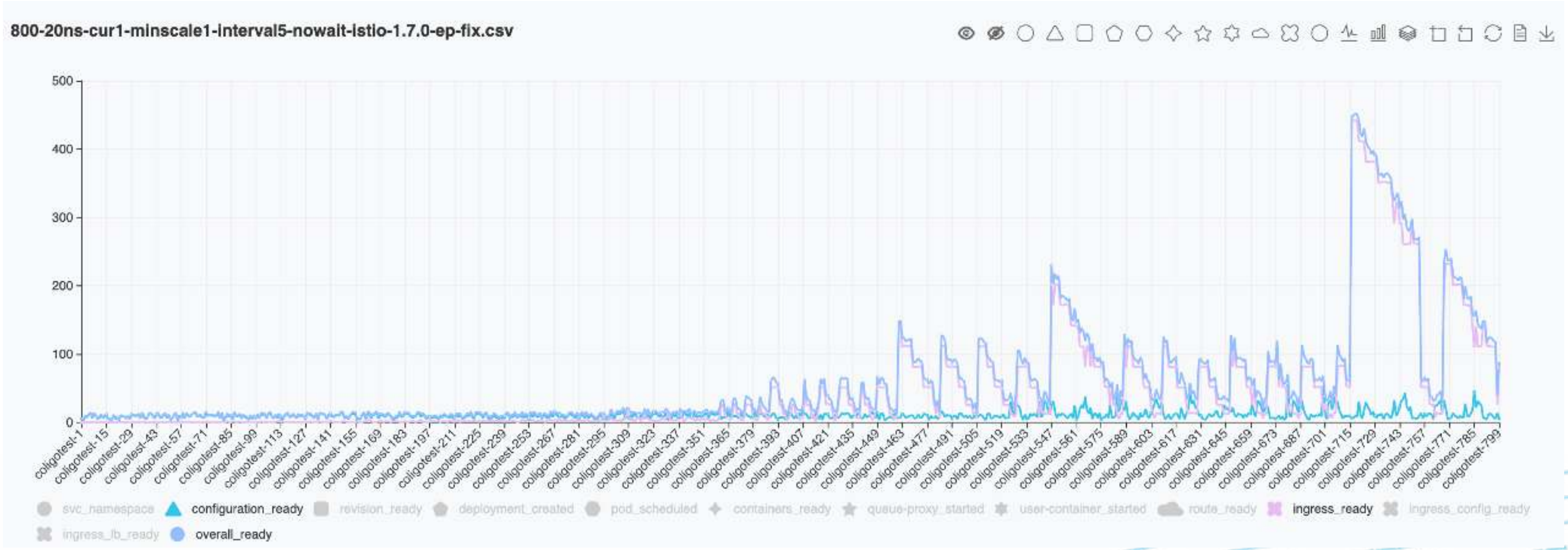
Istio scalability optimization during Knative Service provisioning

- **Detect and analyze Istio scalability issue**
  - `istioctl proxy-config` shows missing endpoint in some of the ingress gateway and recover automatically after ~30mins or restart Istiod.
  - From envoy logs, transient 503 UH "no healthy upstream" errors.
  - From Grafana dashboard, Pilot Pushes shows long latencies.



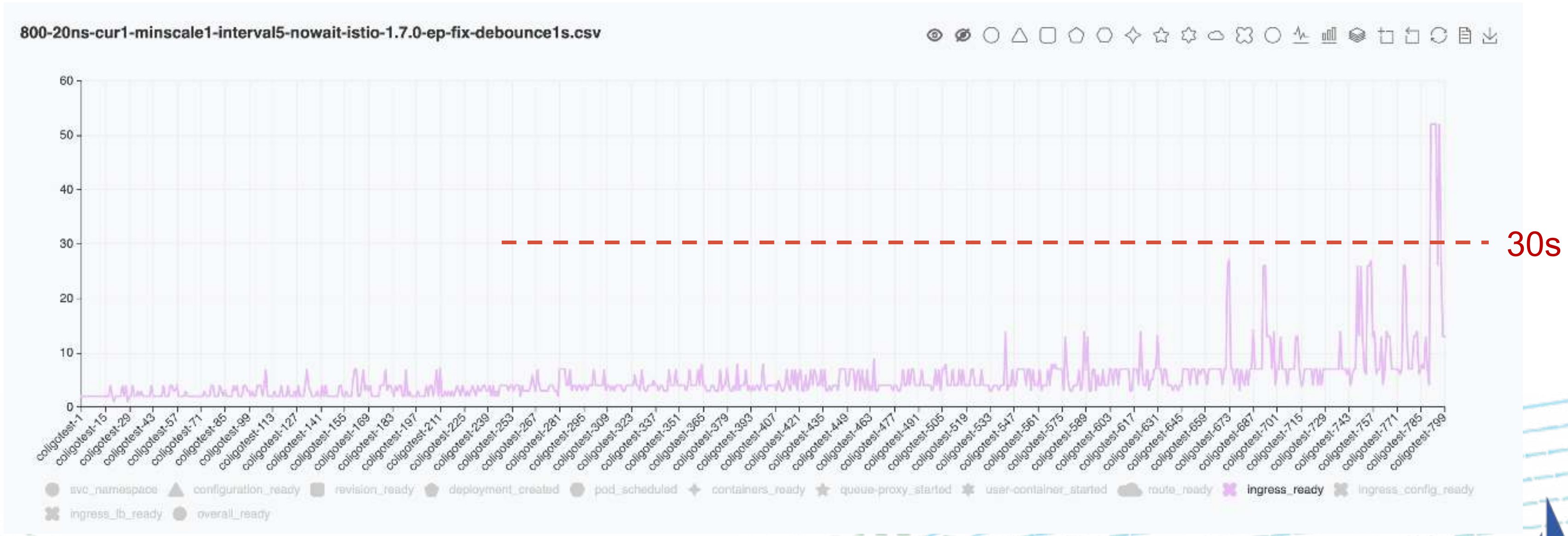
Istio scalability optimization during Knative Service provisioning

- **Random missing endpoint issue is fixed**
  - Radom peaks are fixed in **istio 1.7.1** (istio #23029, envoyproxy #13037)
  - envoy still suffers from overload of XDS pushes in a high churn environment.



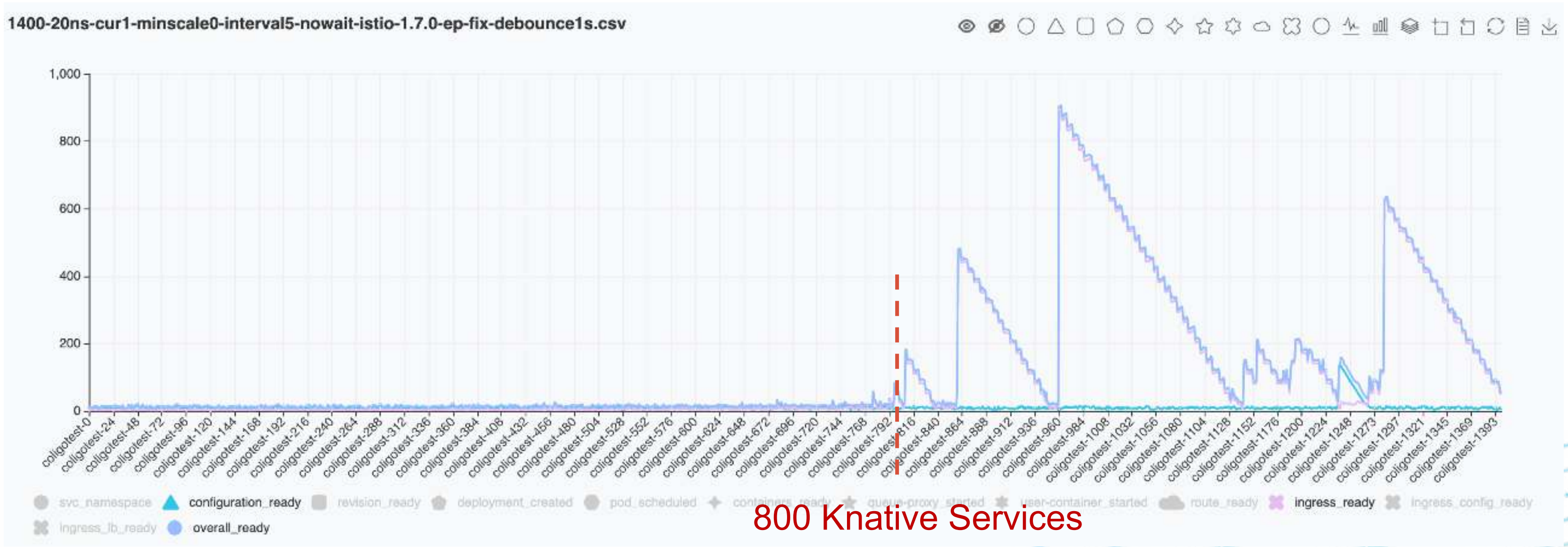
Istio scalability optimization during Knative Service provisioning

- **Tuning debounce time could mitigate the envoy overload issue**
  - `PILOT_ENABLE_EDS_DEBOUNCE=true`,`PILOT_DEBOUNCE_AFTER=100ms` and `PILOT_DEBOUNCE_MAX=10s` are the env vars on pilot that can be tuned.
  - Set `PILOT_DEBOUNCE_AFTER=1s` helps under our workload. (we tested with 100ms, 1s, 2s, 5s, 10s)
  - With 800 Knative Services in total, ingress_ready p98 < 30s



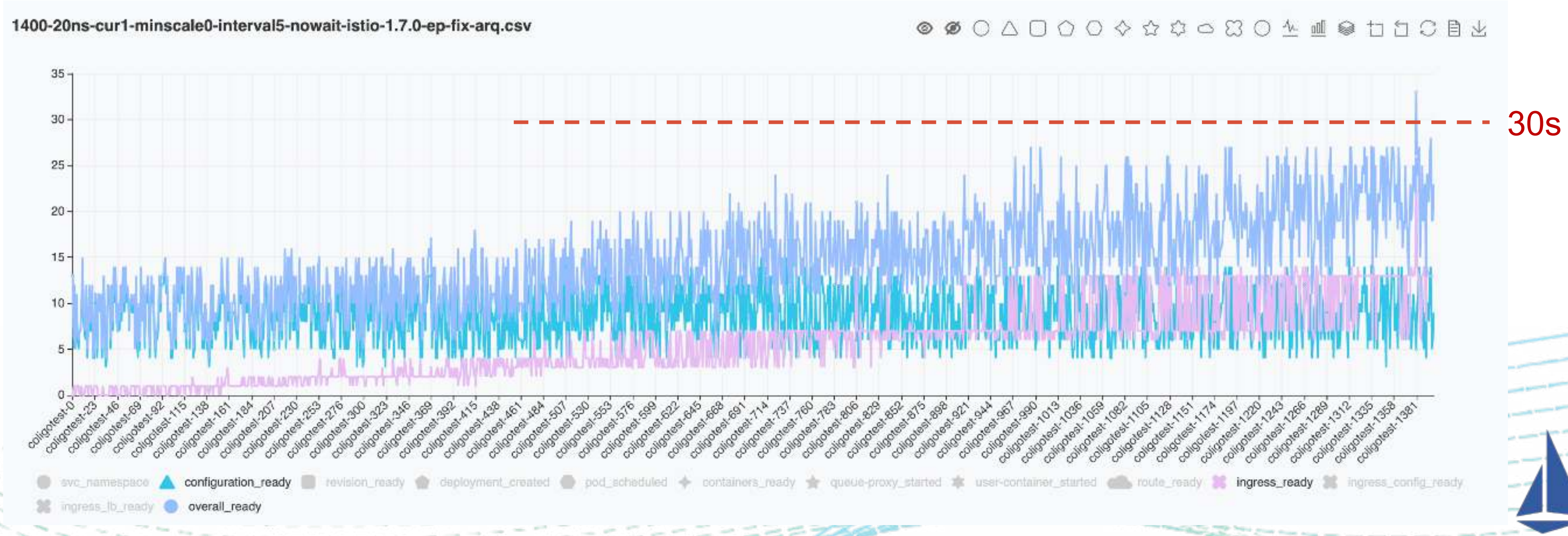
Istio scalability optimization during Knative Service provisioning

- **Envoy overload issue still exits**
  - With 1400 Knative Services in total, envoy overload issue still exists



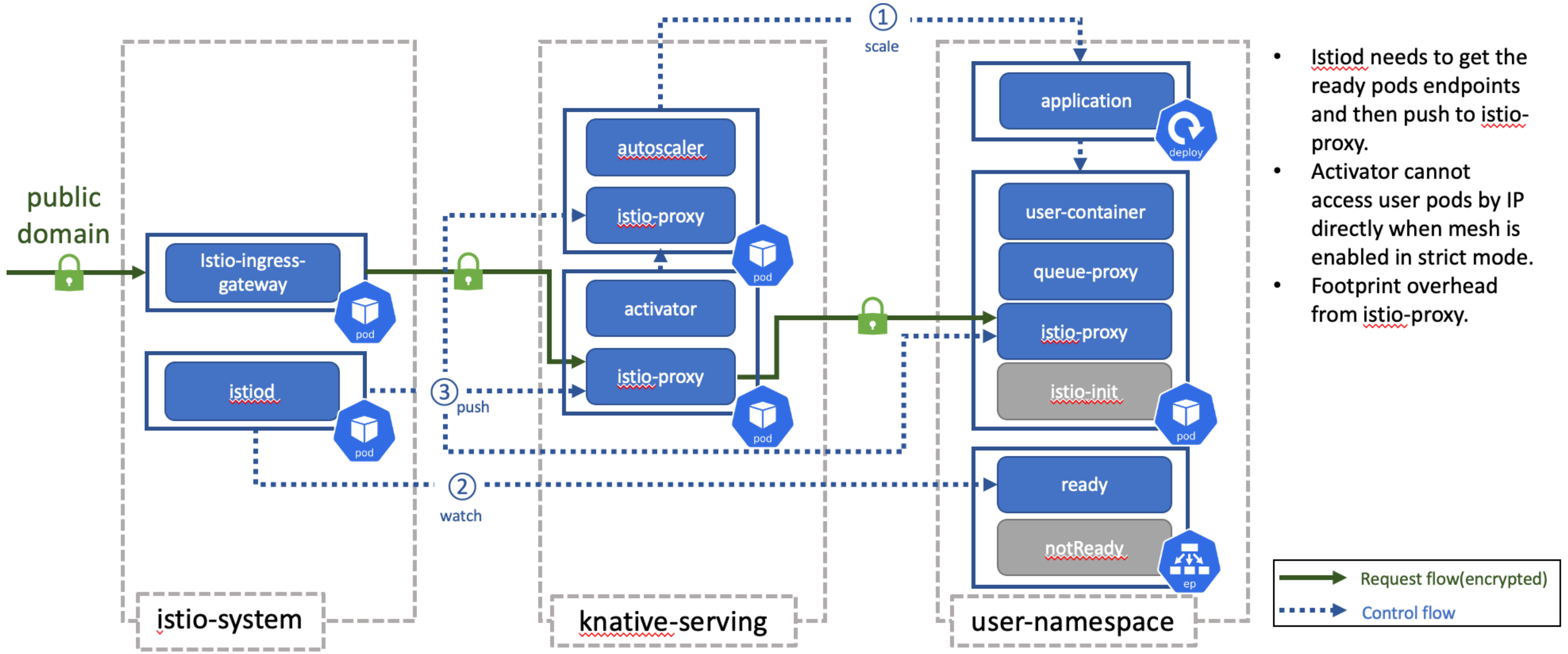
Istio scalability optimization during Knative Service provisioning

- **support for backpressure on XDS pushes to avoid overloading Envoy during periods of high configuration churn**
  - 1400 total with dev release with flow control fix looks great, ingress_ready p100 < 30s
  - [Istio 1.9.x] Support for **backpressure on XDS pushes** to avoid overloading Envoy during periods of high configuration churn. This is disabled by default and can be enabled by setting the `PILOT_ENABLE_FLOW_CONTROL` environment variable in Istiod.
  - Final solution is envoy **delta-XDS** push in future Istio release.



Unleash maximum scalability by fully leveraging Istio features in Knative with service mesh enabled

- **Enable Istio mesh on Knative – Data flow with Istio mesh/mTLS**



- Istiod needs to get the ready pods endpoints and then push to istio-proxy.
- Activator cannot access user pods by IP directly when mesh is enabled in strict mode.
- Footprint overhead from istio-proxy.

#IstioCon

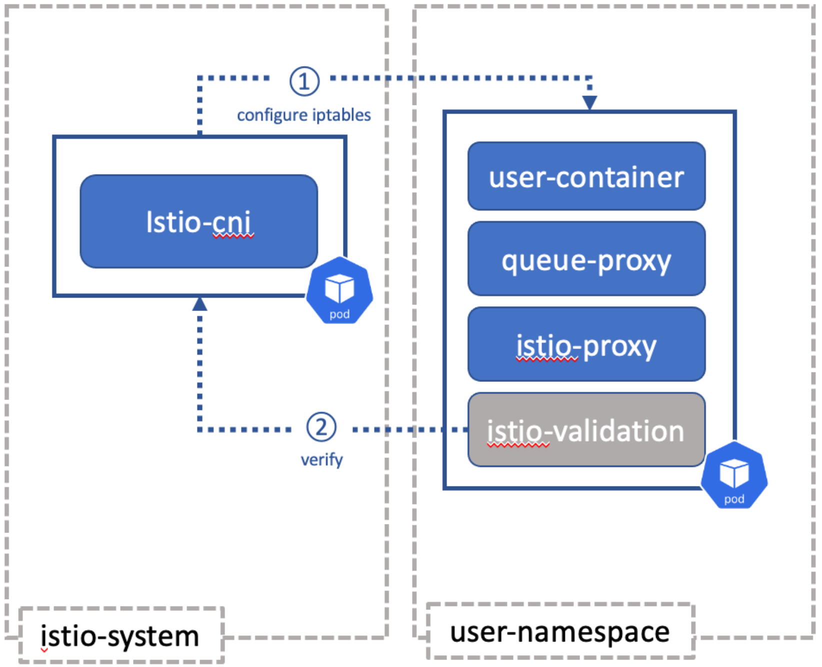
Unleash maximum scalability by fully leveraging Istio features in Knative with service mesh enabled

- **Enable Istio mesh on Knative – Impact without optimization**
  - Init-container added which cost ~5 seconds for Knative application pod code start.
  - Every sidecar needs full mesh information by default. Not a scalability solution.
  - Activator needs to probe the service endpoint since it cannot access pods by IP directly. And it takes time for Istiod to discover the endpoint of ready pods and then push them to the sidecar.
  - Istio-proxy (envoy) sidecar costs ~2 seconds for Knative application pod cold start.

Unleash maximum scalability by fully leveraging Istio features in Knative with service mesh enabled

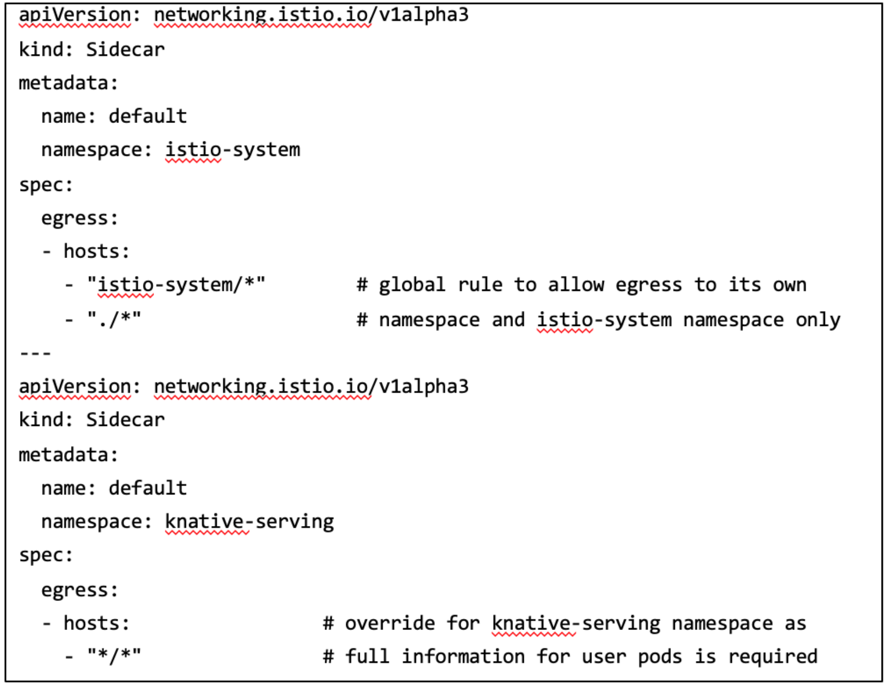
- **Enable Istio mesh on Knative – enable istio CNI plugin**
  - With istio CNI plugin, we can move the iptables configuration parts to CNI. But another init-container, the istio-validation is introduced.
  - We can remove the istio-validation container by modifying the injection template.

  Mitigations:
  - When adding new worker node, make sure daemonset pod of istio CNI plugin is up and running before knative pods scheduling on the node.
  - Crontab job could help to detect whether pod was configured correctly and restart pod



Unleash maximum scalability by fully leveraging Istio features in Knative with service mesh enabled

- **Enable Istio mesh on Knative – Reduce mesh size in app sidecar**
    - User cases: no service access cross user namespace.
    - The sidecar CR helps to limit the known egress hosts for sidecars, sidecar needs to knows mesh in his own user namespace only.
    - We can limit the mesh size to namespace scope for all user namespaces easily.



```
apiVersion: networking.istio.io/v1alpha3
kind: Sidecar
metadata:
  name: default
  namespace: istio-system
spec:
  egress:
  - hosts:
    - "istio-system/*"        # global rule to allow egress to its own
    - "./*"                   # namespace and istio-system namespace only
---
apiVersion: networking.istio.io/v1alpha3
kind: Sidecar
metadata:
  name: default
  namespace: knative-serving
spec:
  egress:
  - hosts:                  # override for knative-serving namespace as
    - "*/*"                 # full information for user pods is required
```

Unleash maximum scalability by fully leveraging Istio features in Knative with service mesh enabled

- **Enable Istio mesh on Knative – Pod IPs addressable directly in mesh**
  - Knative needs to access POD by IP directly for performance and efficiency.
  - When mesh enabled, all traffic through Kube service managed by istio mesh.
  - Knative community is working to use Destination rules for Pod IPs addressable directly.
    Knative issue: https://github.com/istio/istio/issues/23494

Reference

- IBM Cloud Code Engine which fully managed, serverless platform(including knative and istio) that can host all of your cloud native workloads: https://www.ibm.com/cloud/code-engine
- Kperf, a public Knative benchmark tool helps everyone to understand the issue and accelerate the whole debug and fix process: https://github.com/knative-sandbox/kperf
- Get Istio CPU/MEM stats: https://github.com/istio/istio/wiki/Analyzing-Istio-Performance
- Debugging Envoy and Istiod https://istio.io/latest/docs/ops/diagnostic-tools/proxy-cmd/
- Pilot agent config https://istio.io/latest/docs/reference/commands/pilot-agent/
- Istio Sidecar Configuration https://istio.io/latest/docs/reference/config/networking/sidecar/
- Istio CNI plugin https://istio.io/latest/docs/setup/additional-setup/cni/

Thank you!

#IstioCon