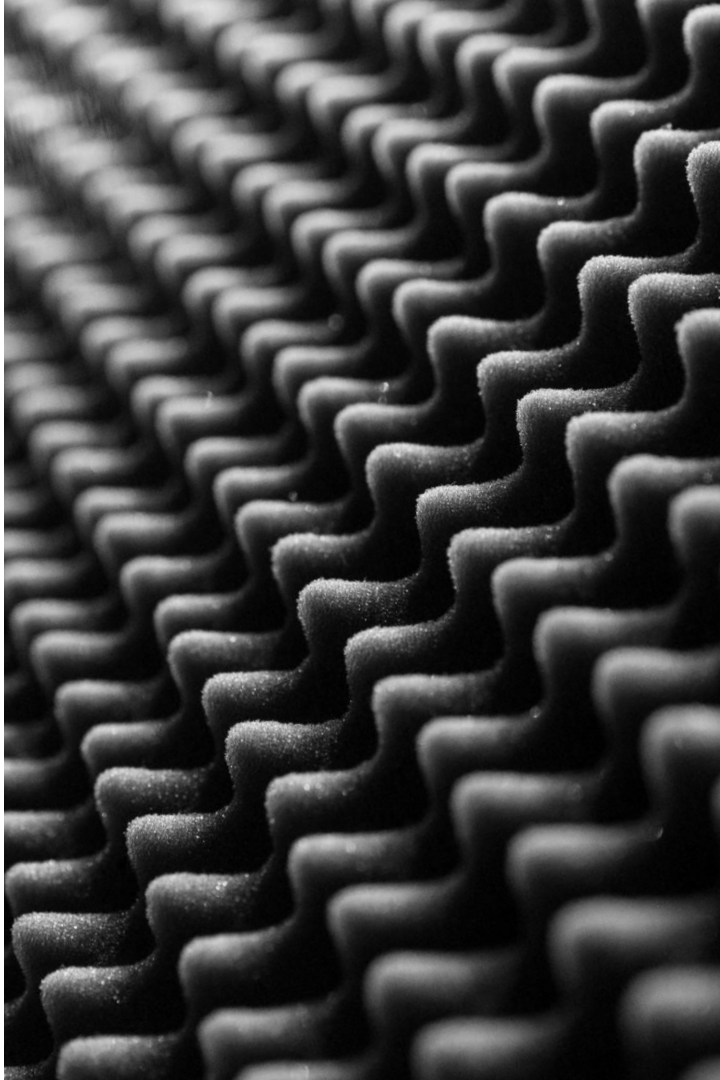
Turn the Volume Down on Noisy Neighbors!

Sándor Guba, Axoflow

AXOFLOW
LOGGING UNLEASHED



Why am I talking about this?

- Co-founder and CTO at Axoflow
- Formerly: Co-founder at Banzai Cloud
- Main fields: monitoring, logging, tracing, alerting
- Kubernetes user since the early days
- Logging Operator maintainer & Telemetry Controller designer
- Observability enthusiast

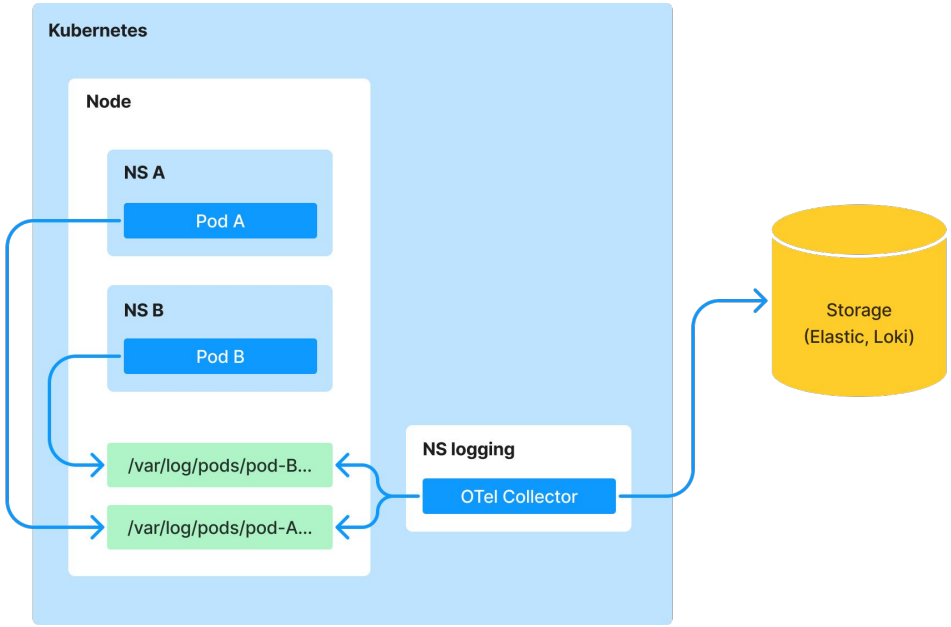
Intro to Kubernetes logging

Logs are cluster resources

Kubernetes has a lot of isolation methods (Namespaces, RBAC, Policy, ...) but these boundaries don't apply to logs!

Collector Agent

Deploying an agent that reads the logs from the node filesystem and transports them to the desired output



Intro to Kubernetes multi-tenancy

Tenants as resources

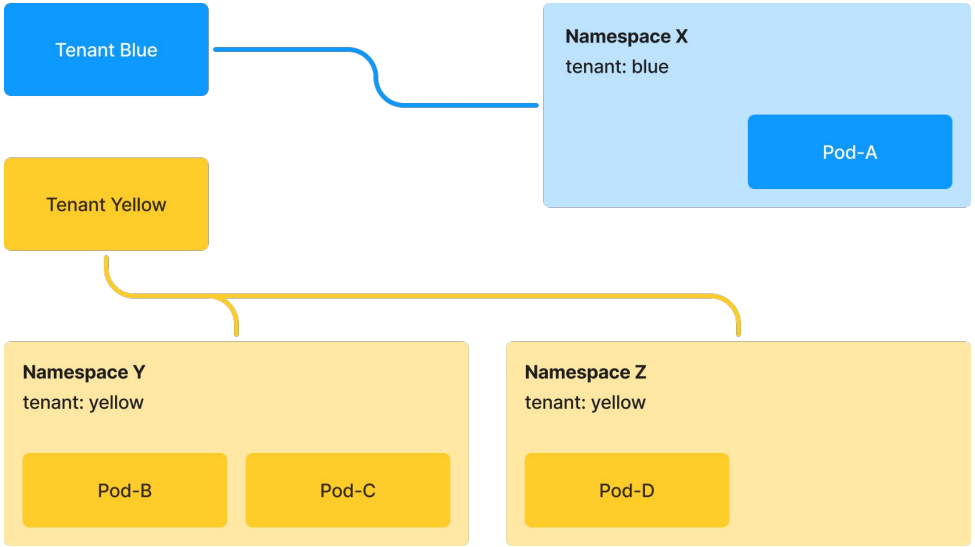
Each solution uses Kubernetes custom resources to define tenant

Labelled namespaces

Namespaces remains the main boundaries between tenants

Examples

Rancher project, Capsule or homegrown solution



OpenTelemetry the Cloud Native observability

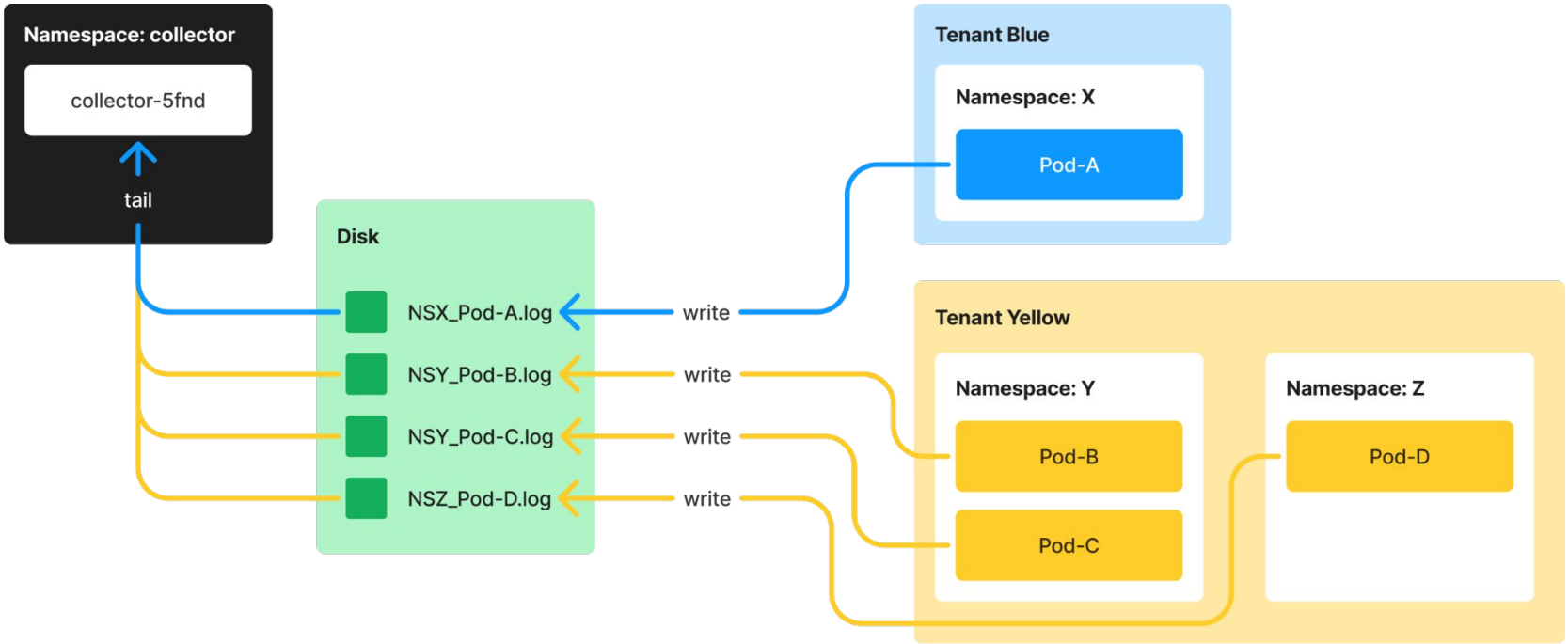
- *“OpenTelemetry is a collection of APIs, SDKs, and tools. Use it to instrument, generate, collect, and export telemetry data”*

- A vendor neutral protocol for telemetry interoperability
  - There are several custom protocol i.e: fluent, lumberjack (filebeat),...
  - General purpose protocols: HTTP, gRPC

- **OpenTelemetry Collector**
  - Reference implementation in Go
  - Widely adopted by big players

Why multi-tenancy is hard?

Multi-tenancy from the node perspective



Log collection from the agent perspective

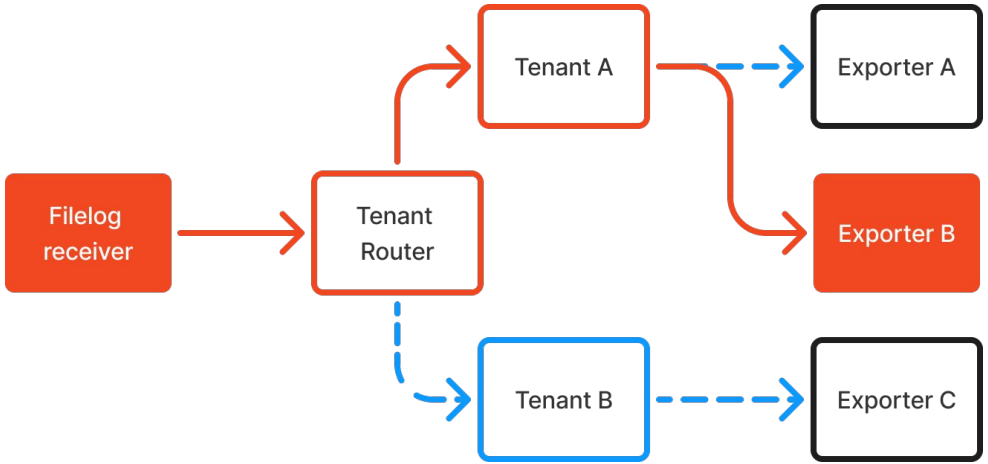
```
/var/log/pods/$ns_$pod_$id.log
kubernetes:
   pod_name: "log-gen-..-...",
   namespace_name: "default",
```


```
80.26.45.72 - -
[07/Mar/2024:22:07:29 +0100]
"POST /app/v1/unknown
HTTP/1.1" 404 17202 "-" …
```

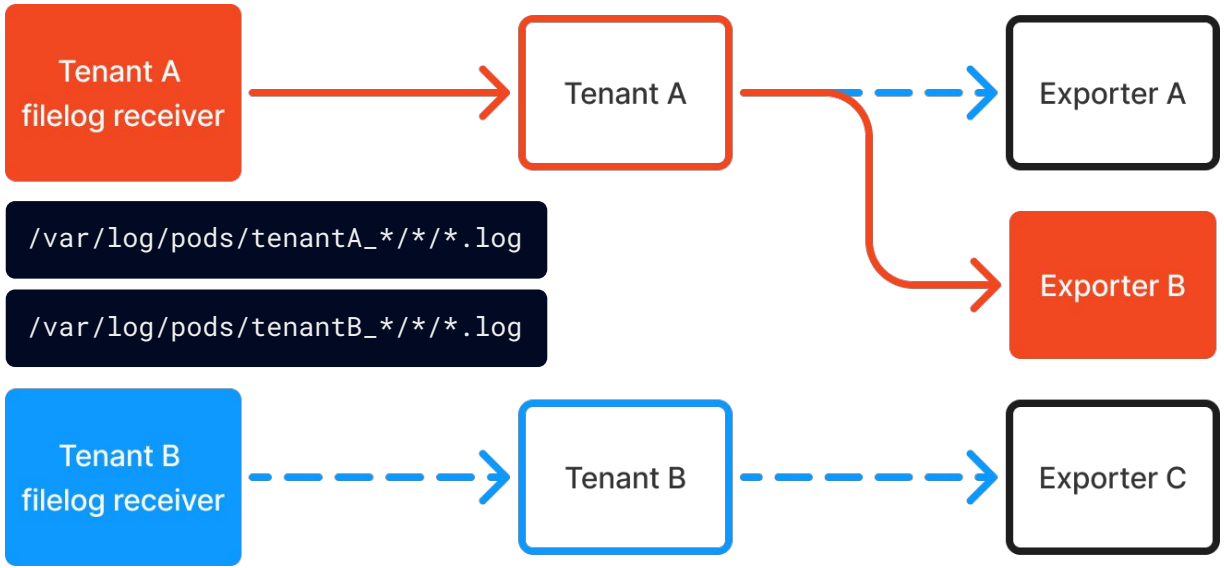
```
kubernetes:
    cluster: "dev",
    container_name": "log-gen",
    pod_name: "log-gen-..-...",
    namespace_name": "default",
    labels":
        app: "log-generator",
        team: "web"
```

What could go wrong in a shared environment?

- Huge shared configuration
- Noisy neighbours
  - Configuration conflicts
  - Resource exhaustion
- Backpressure
- Error handling
  - What to do when a parse fails?
- Buffering
  - How-to



Working around backpressure



Note

We need to **update** the config whenever a namespace or a tenant **changes**

Telemetry Controller

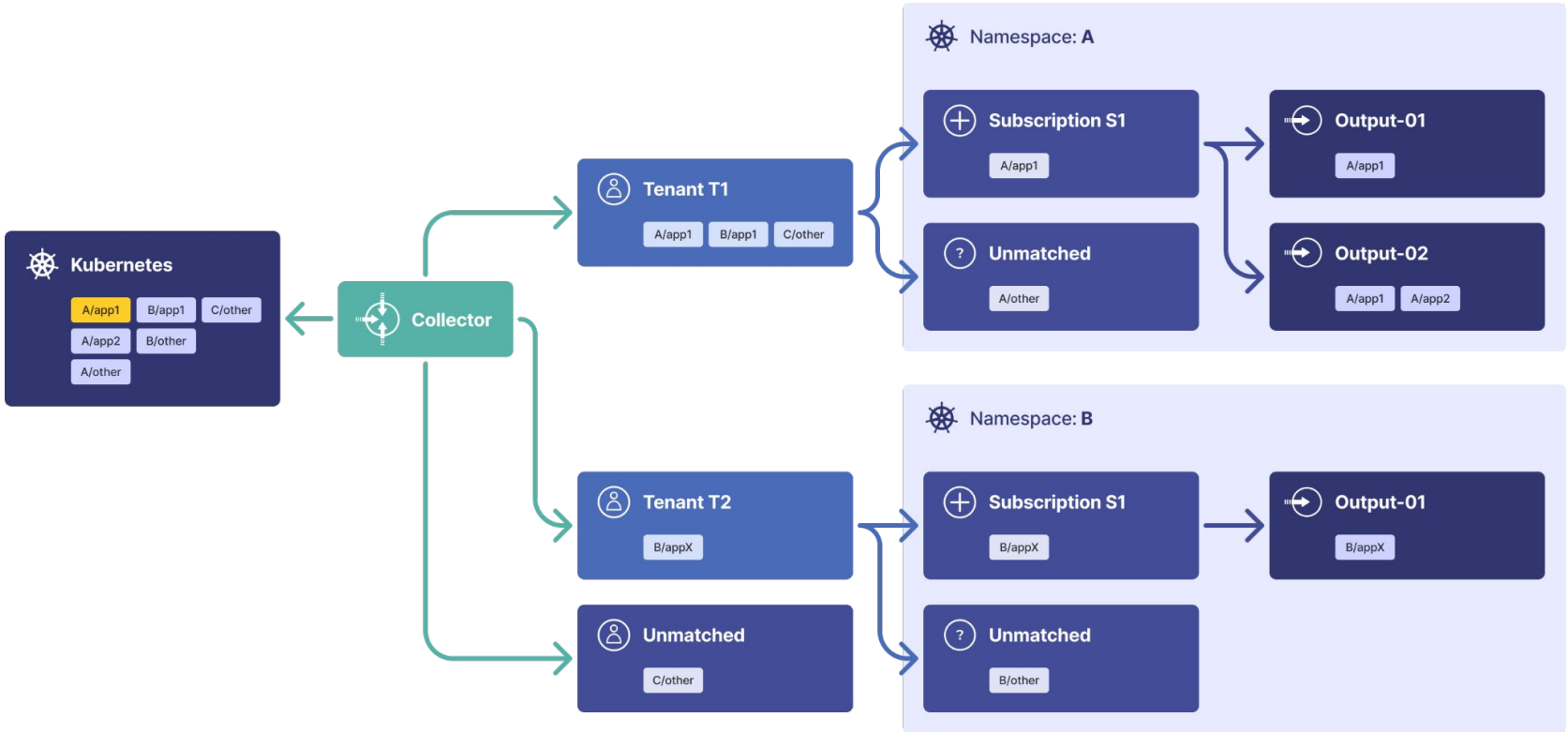
Telemetry Controller

- Telemetry Controller **turns telemetry** event streams **into** Kubernetes **resources**!
- **Isolation** and **access control** for telemetry (logs) data

Why?

- OpenTelemetry Collector is a low-level tool to manage telemetry pipelines
- Defining a simple Kubernetes pipeline easily grow around 100-200 lines of configuration
- Provides a well defined **opinionated** way to manage logging

Telemetry Controller resource model

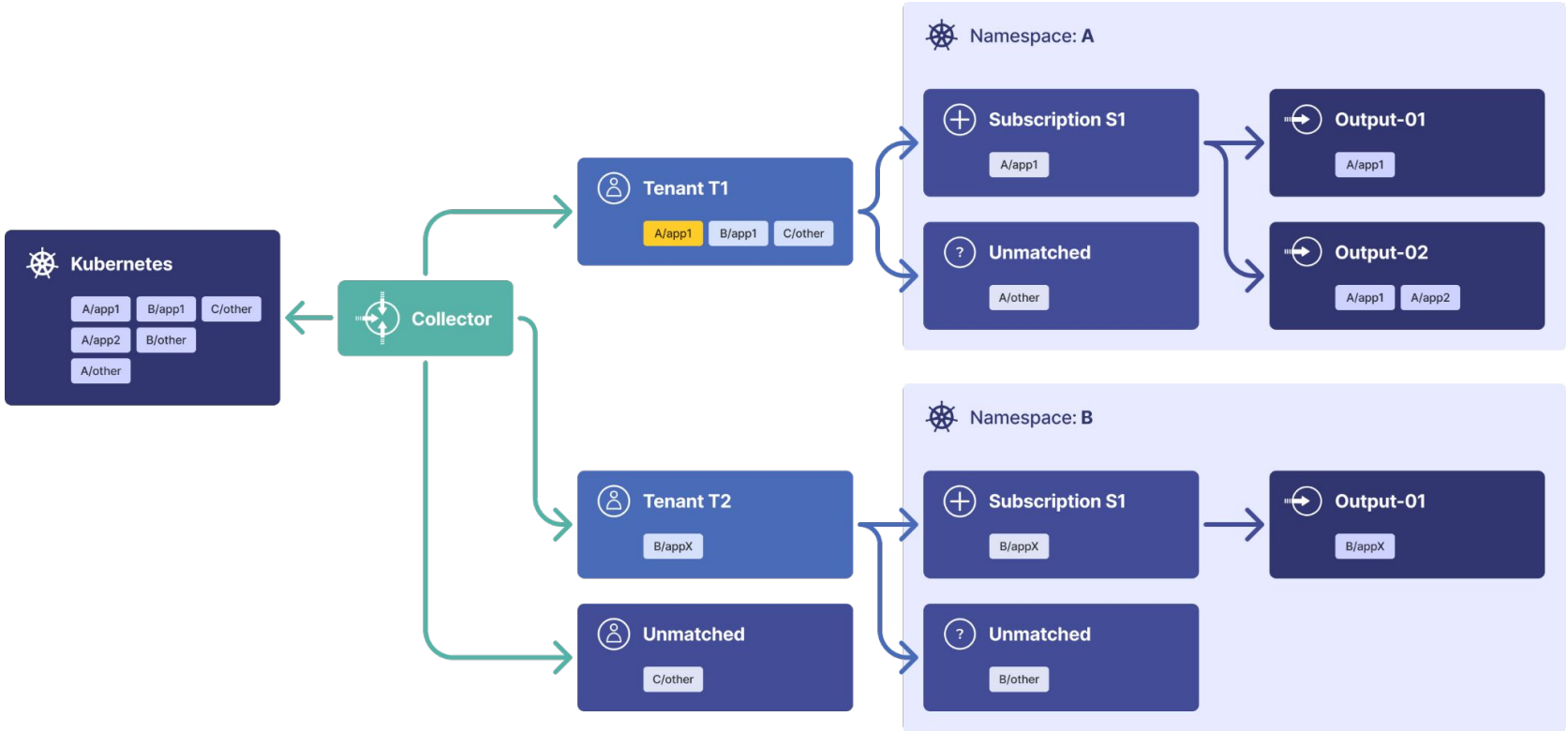


Collector (cluster) resource

- OTel Collector settings
- Agent namespace
- Tenant Selector
  - Hard multi-tenancy
  - Multi-arch setup

```yaml
kind: Collector
metadata:
  name: cluster
spec:
  controlNamespace: collector
  tenantSelector:
    matchLabels:
      collector: cluster
```

Telemetry Controller resource model



Tenant (cluster) resource

Tenant configuration

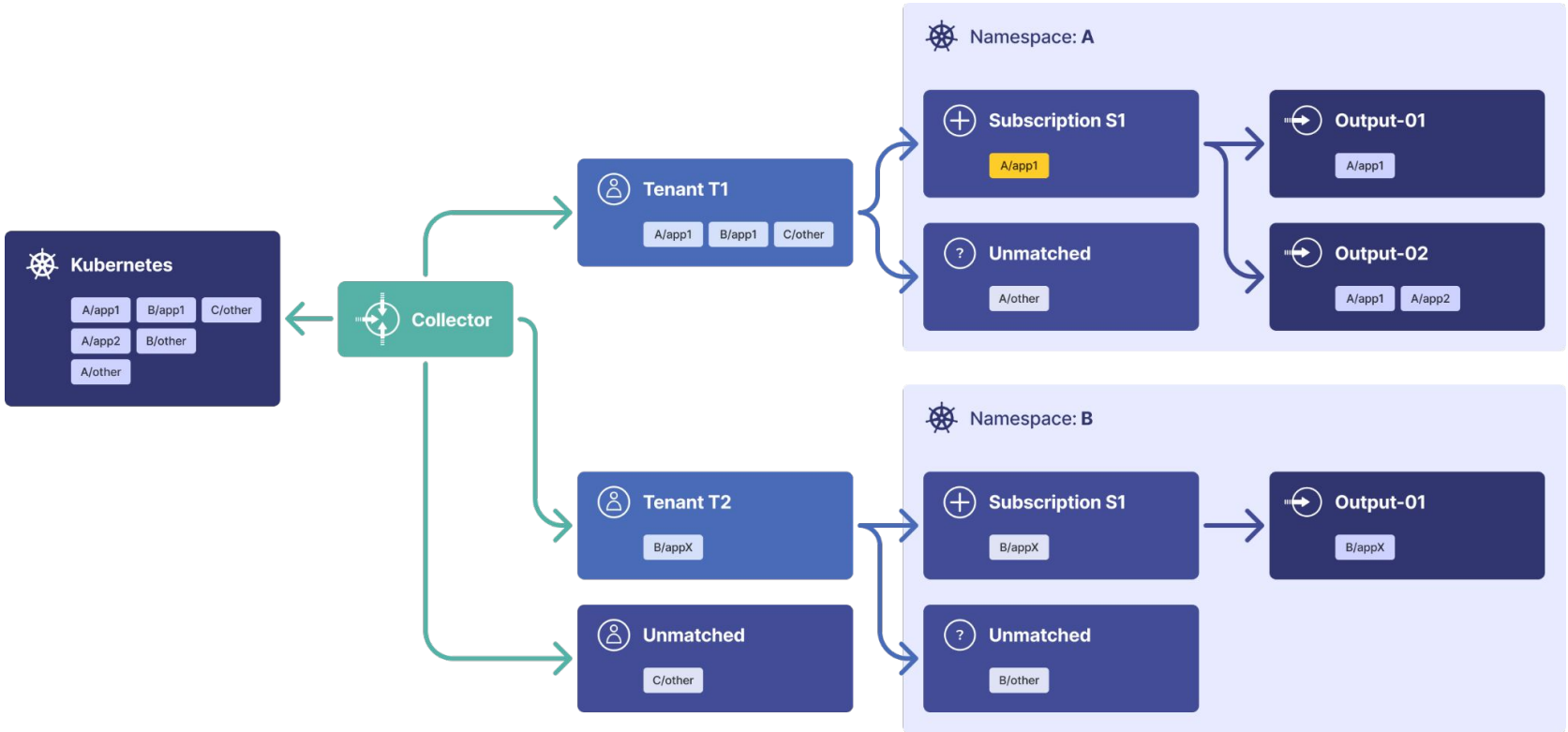
- Labels for the collector
- Namespace selectors to allow access to logs

Differentiate between

- Configuration namespace
- Log collection namespace

```yaml
kind: Tenant
metadata:
  name: web
  labels:
    collector: cluster
spec:
  subscriptionNamespaceSelectors:
    - matchLabels:
        tenant: web
  logSourceNamespaceSelectors:
    - matchLabels:
        tenant: web
```

Telemetry Controller resource model

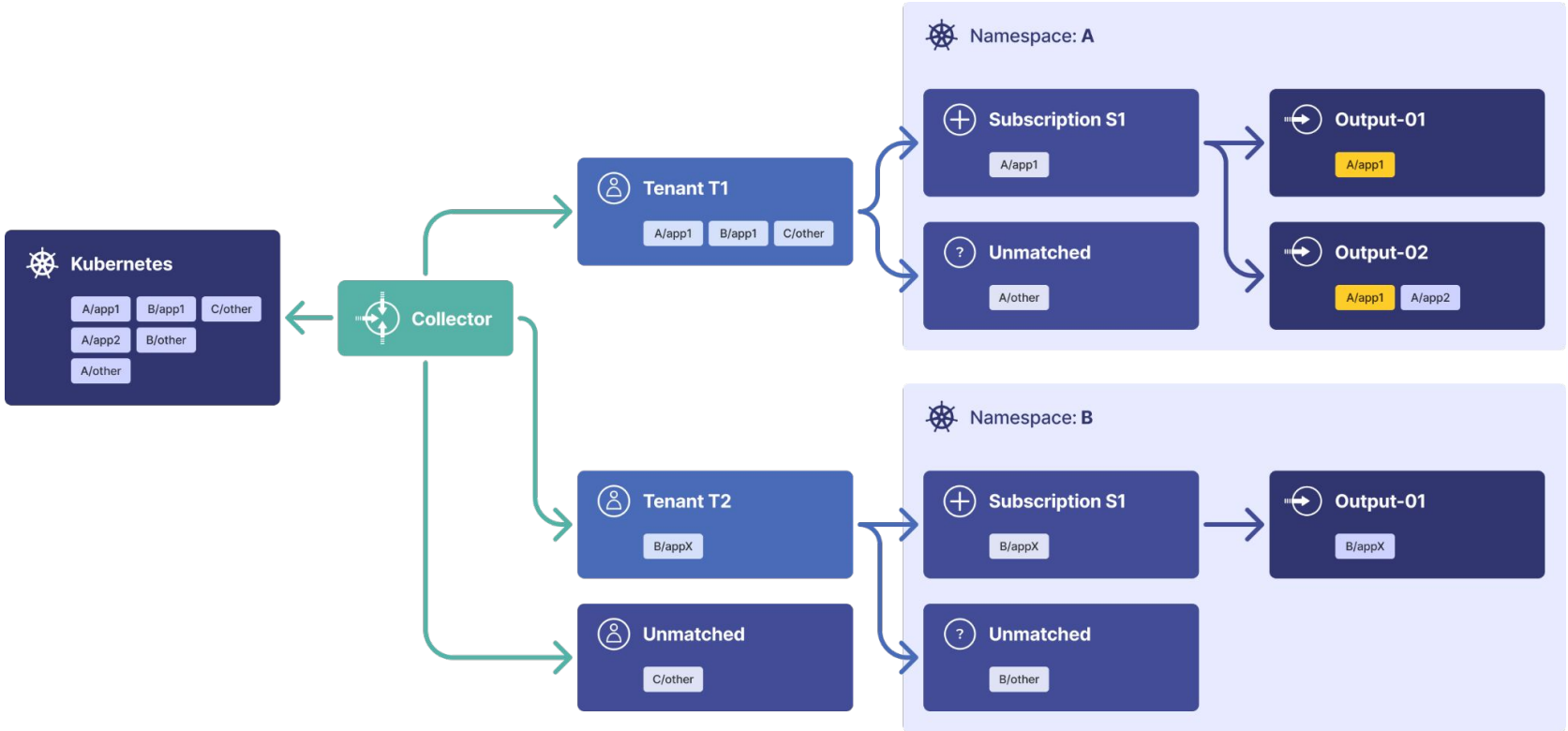


Subscription resource

- With a subscription a user can “subscribe” to all or a portion of the logs and route them to one or more destinations.
- Filtering provided by an OTTL expression
  - (wip use condition for filtering instead of OTTL expression)

```yaml
kind: Subscription
metadata:
  name: all-logs
  namespace: web
spec:
  ottl: "route()"
  outputs:
    - name: openobserve
      namespace: web
```

Telemetry Controller resource model



Output resource

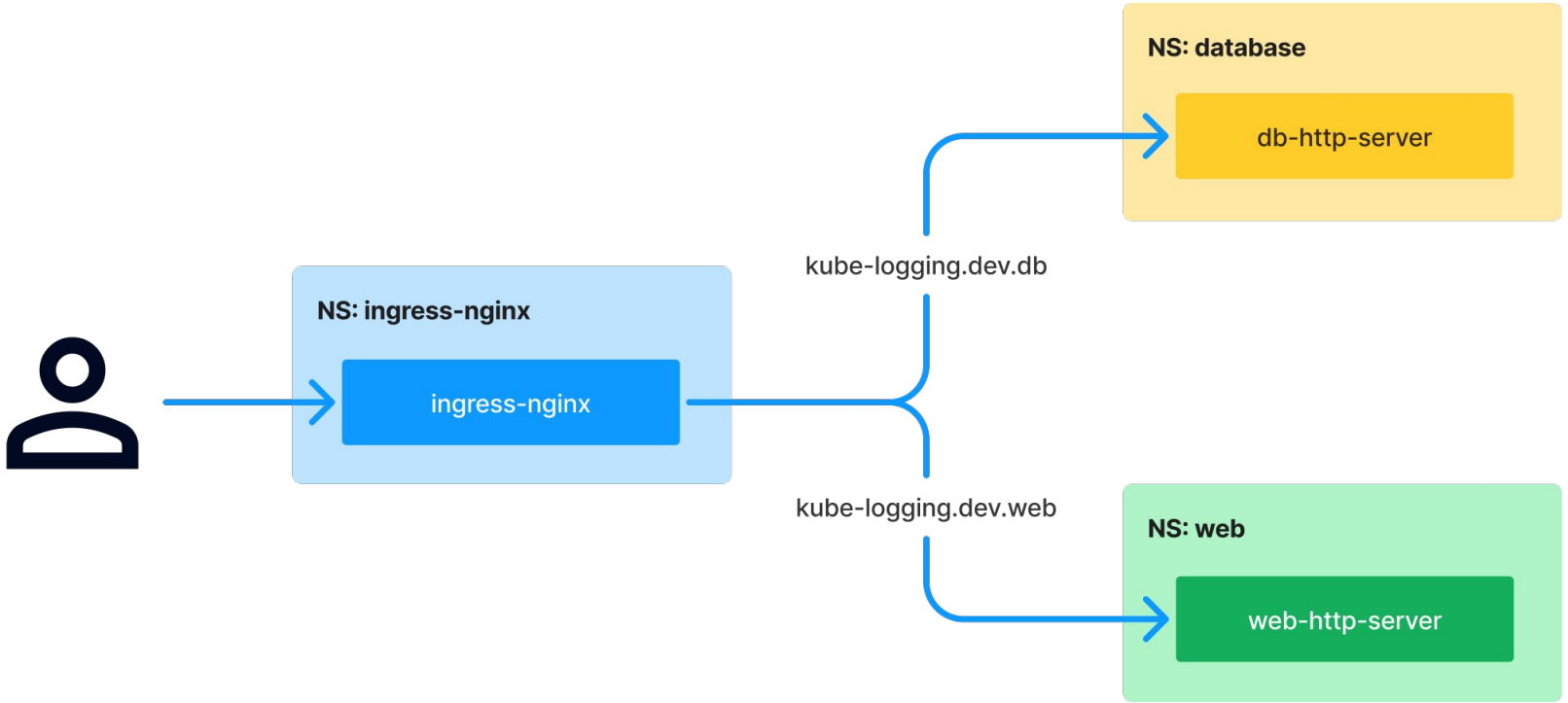
Output configuration

- Currently support limited number of destinations
- Secret management is still work in progress as

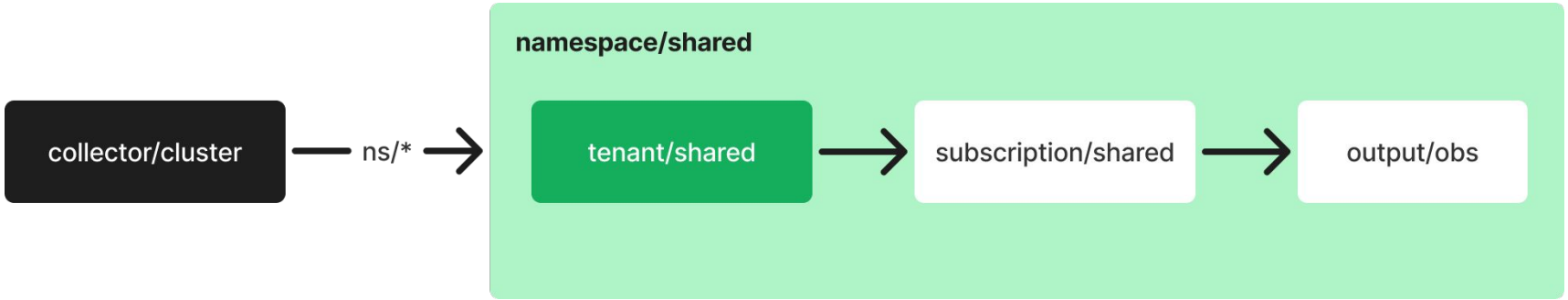
```yaml
kind: Output
metadata:
  name: openobserve
  namespace: web
spec:
  otlp:
    endpoint: openobserve...:5081
    headers:
      Authorization: "Basic ..."
      organization: default
      stream-name: default
    tls:
      insecure: true
```

Demo Time

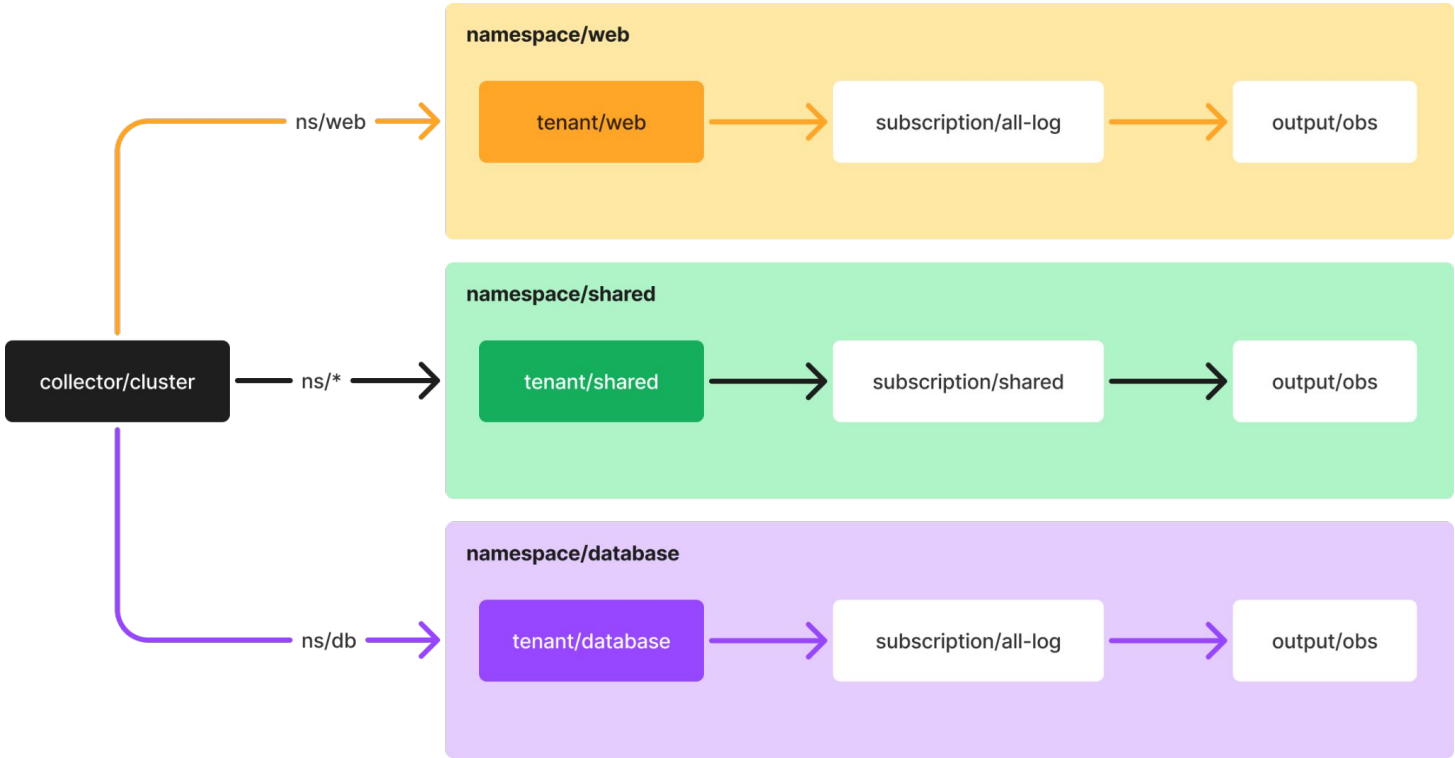
Demo environment workloads



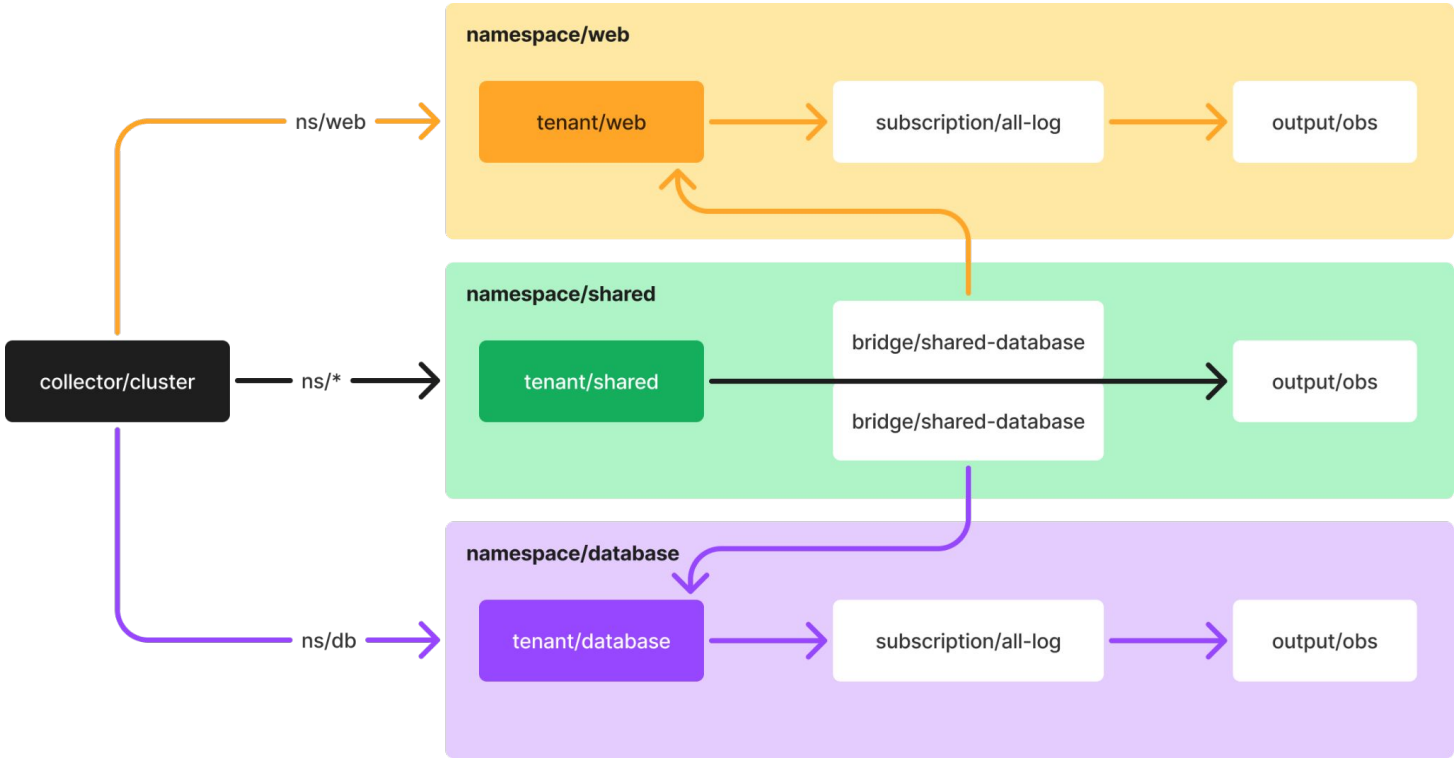
Single tenant cluster



Multi-tenant Cluster



Shared tenant multi-tenant cluster



Bridge resource

With a bridge it is possible to route messages between tenants based on a condition

```
kind: Bridge
metadata:
  name: shared-web
spec:
  sourceTenant: shared
  targetTenant: web
  ottl: |
   route() where
attributes["vhost"] ==
"kube-logging.dev.web"'
```

Not in the scope of the Telemetry Controller

- Telemetry Controller is not a generic OpenTelemetry Collector abstraction, so it doesn’t aim to provide an API for all the knobs and levers of the Collector.
- Also it doesn’t aim to provide aggregation capabilities, as it is primarily focused on the log collection problem space.

Feature completeness

The four different log sources

- **Containers**
- Applications (SDK/sidecar)
- Platform (Hosts, Events)
- External logs (Pull/Push)

Traces & Metrics signals

Still some work ahead

- **Monitoring the pipeline**
  - Reference metrics are in place
- Outputs
  - **Secrets**
  - Out of box multi-tenancy
- Default route for subscription
- **Shared logs**

Useful links

- **Github**
  - https://github.com/kube-logging/telemetry-controller
  - https://github.com/axoflow/tc-demo-kubecon (coming soon)
- **Intro blog**
  - https://axoflow.com/reinvent-kubernetes-logging-with-telemetry-controller
- **Me**
  - https://www.linkedin.com/in/sandorguba
  - https://x.com/Tarokkk

Questions?