Letting Your Containers Dream of Electric Sheep

Cyrill Troxler

Contents

- Intro
- Motivation
- Existing solutions
- Demo
- Technical details
- What’s next

Intro

Cyrill Troxler

- Platform Engineer at Nine
- Infrastructure provisioning with Kubernetes
- Backend for deplo.io
- Exploring interesting technologies on the side

Motivation

- *«If only we could scale to zero.»*
- Previous experimentation with container checkpoint/restore
- Excuse to play with eBPF
  - SOCKMAP - TCP splicing of the future

Existing solutions

- Keda HTTP addon
  - Uses a layer 7 proxy
  - Sets replicas on-demand
  - State is lost when scaling to zero
  - Pod startup latency becomes very relevant
  - Image pulls make things worse
- Knative
  - Works similar to Keda (pods created on-demand)
  - Deploys a proxy sidecar per pod
  - Quite a beast to configure

Knative from zero

Architecture

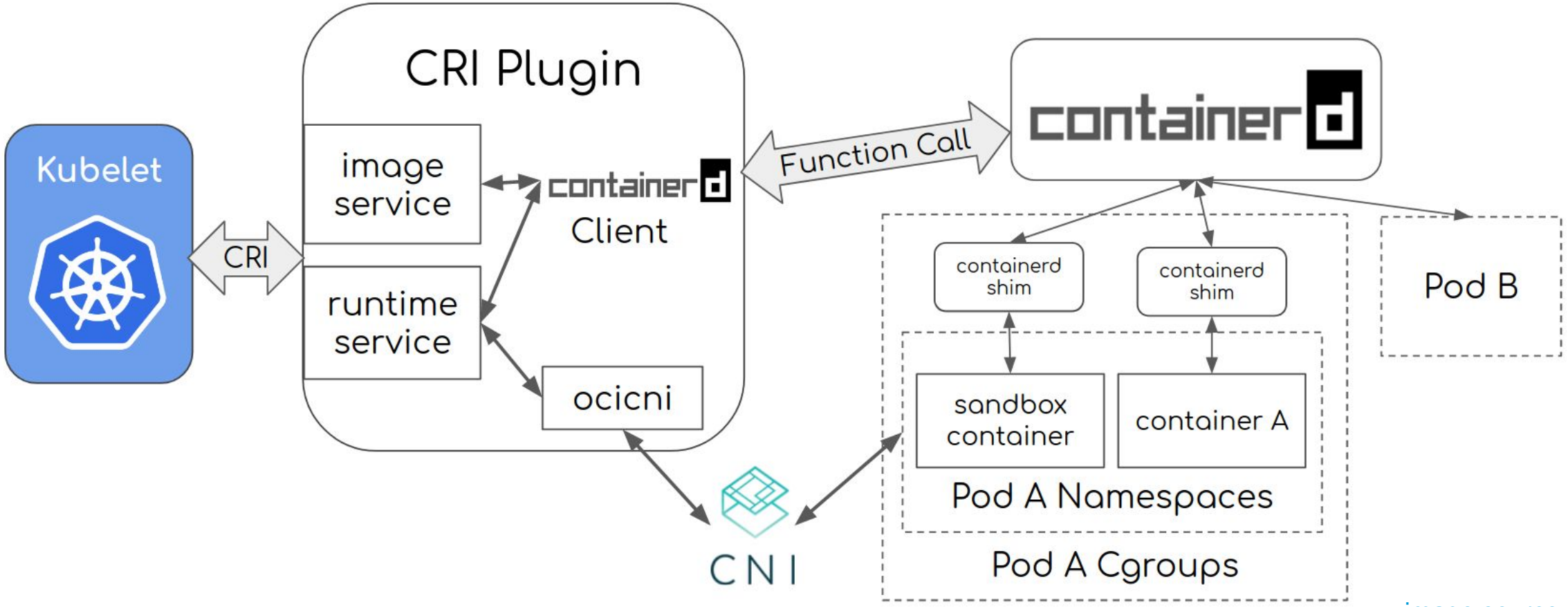


image source

Zeropod Demo

What we get with Zeropod

- Serve on-demand with sub-second* delay
- Retains state when scaling to zero
- Only proxies requests during restore
- Supports raw TCP, is not limited to HTTP
- No code changes, use your existing k8s resource definitions
- Scale down duration can be configured

*depends on many factors such as RSS of process and CPU and Storage

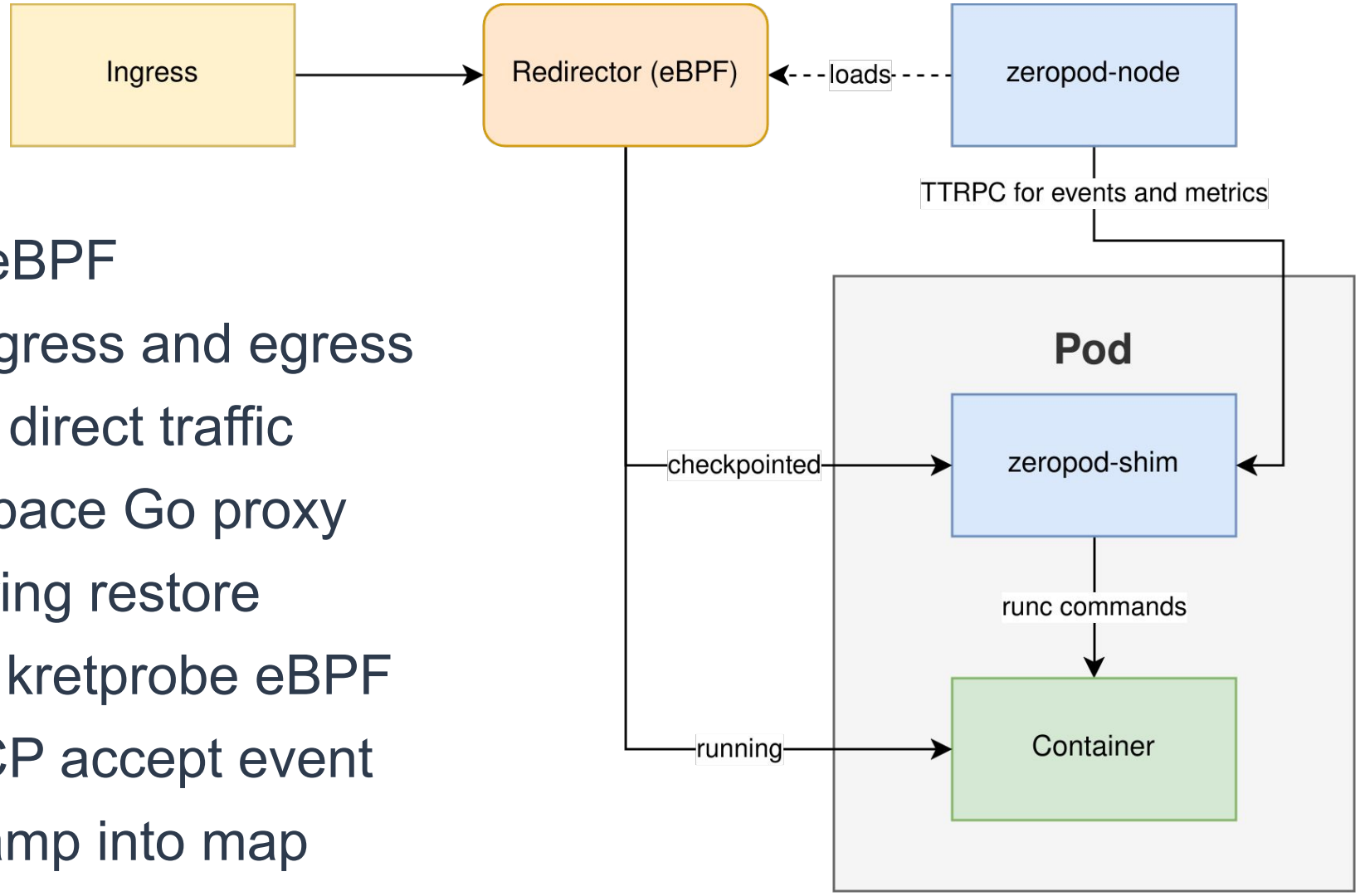
Some use cases

- Low traffic apps
- Dev/Staging environments
- "Mostly static" sites that still need some server component

Technologies

- Containerd & runc
- CRIU – Checkpoint/Restore In Userspace
  - Watch any talk from Adrian Reber (KCD 2023)
- eBPF (cilium/ebpf)

Architecture



- Redirector – TC eBPF
  - Attaches to ingress and egress
  - Uses maps to direct traffic
- Activator – userspace Go proxy
  - Only used during restore
- Socket Tracker – kretprobe eBPF
  - Tracks last TCP accept event
  - Writes timestamp into map

Shim memory usage

- Import only what you need (Go)
- Loading eBPF programs outside of shim
- Grouping (?)
  - `io.containerd.runc.v2.group: "key"`

In-place resource updates

- Currently (as of k8s 1.30) behind a feature flag
  `--feature-flags=InPlacePodVerticalScaling=true`
- If enabled, Zeropod will patch pod requests on scaling events

```
requests:                    requests:
  cpu: 100m          ⟶          cpu: 1m
  memory: 128Mi                 memory: 1Ki
```

How to get started

```
kind create cluster
kubectl apply -k https://github.com/ctrox/zeropod/config/production
kubectl label node kind-control-plane zeropod.ctrox.dev/node=true
```

- Installer DaemonSet is scheduled onto labelled node
- Shim binary and pre-built CRIU is copied to /opt/zeropod on host
- Containerd config is patched to load runtime
- RuntimeClass is created

What's next

- Containerd 2.0
  - Current shim already works with it
  - Some new features require the shim to be upgraded
  - User namespaces support
- Adoption at Nine
- Compatibility testing
  - for example: NodeJS libuv uses io_uring

Resources and Links

- github.com/checkpoint-restore/criu
- github.com/cilium/ebpf
- github.com/kedacore/http-add-on
- Knative scale to zero
- Containerd CRI Architecture

Q&A

github.com/ctrox/zeropod

nine