KubeCon CloudNativeCon Europe 2021

Virtual

Using Sloop for monitoring highly available services

Hemanth Siddulugari and Sana Jawad
Salesforce

Using Sloop for monitoring highly available services

Kubecon EU 2021

Hemanth Siddulugari, Principal Software Engineer
@siddulugari | hsiddulugari@salesforce.com

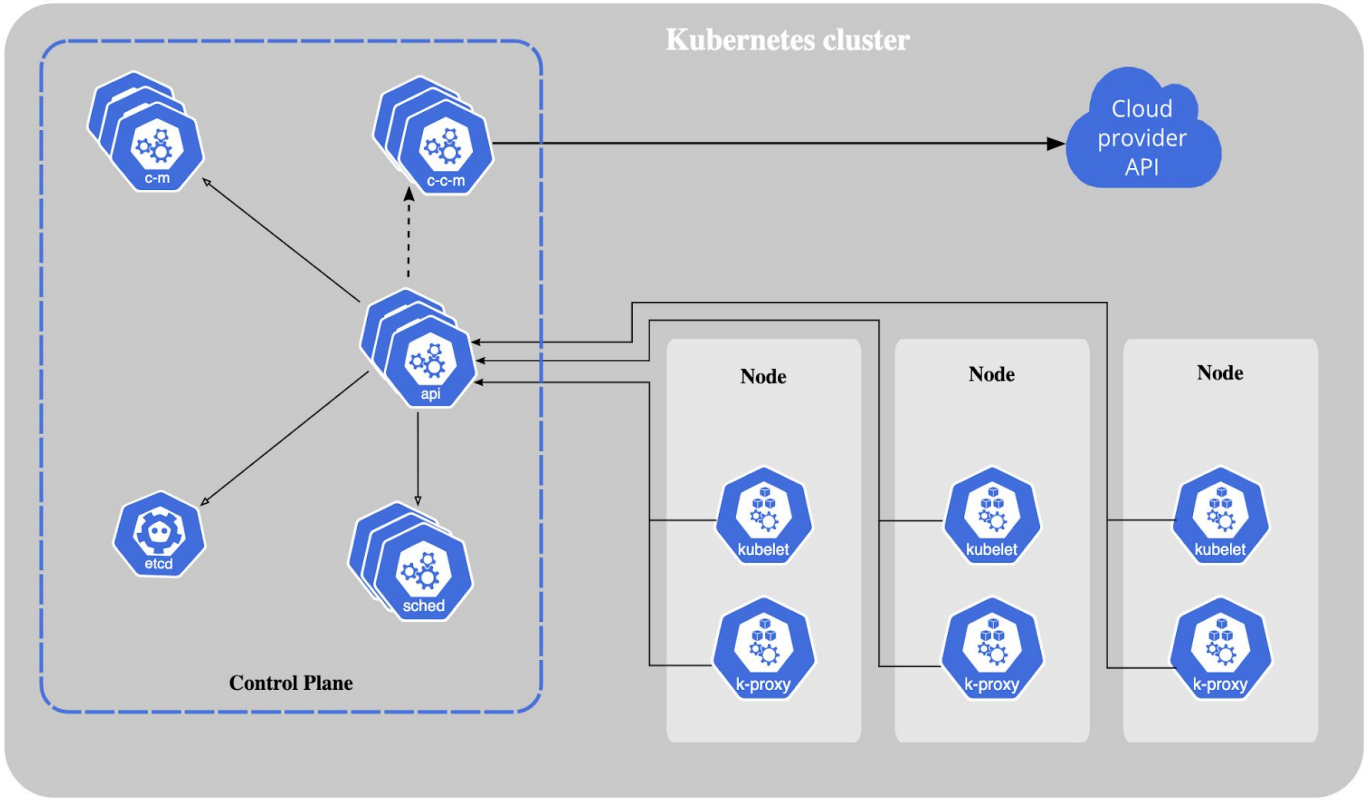
Sana Jawad, Lead Software Engineer
@sjawad| sjawad@salesforce.com

Introduction

- Who is this talk for?
- Agenda
  - Quick overview of Sloop
  - Incidents/Use cases

Quick overview

Kubernetes Architecture



API server

Cloud controller manager *(optional)*

Controller manager

etcd (persistence store)

kubelet

kube-proxy

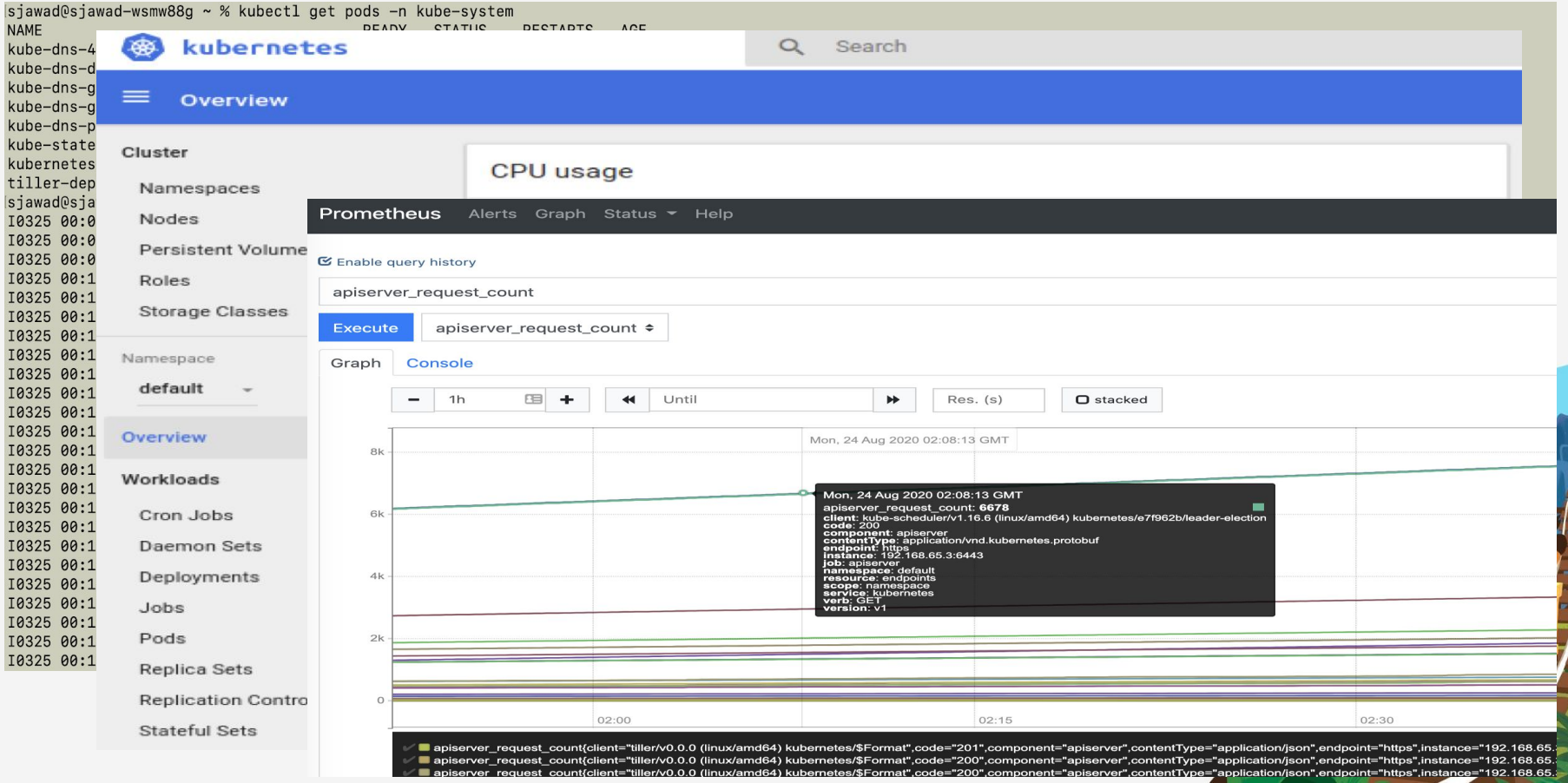
Scheduler

Control plane

Node

Kubernetes Components, https://kubernetes.io/docs/concepts/overview/components/

Tools for current state



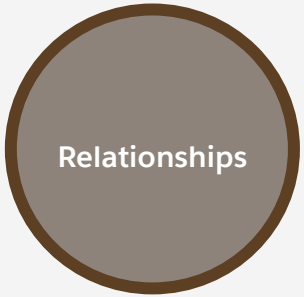
Why build new?

Find means to find following data?

- Details of resources that no longer exist
- Information about a pod from an evicted node?
- Rollout details of older deployments
- Timings of pods replacements and their health checks

Features

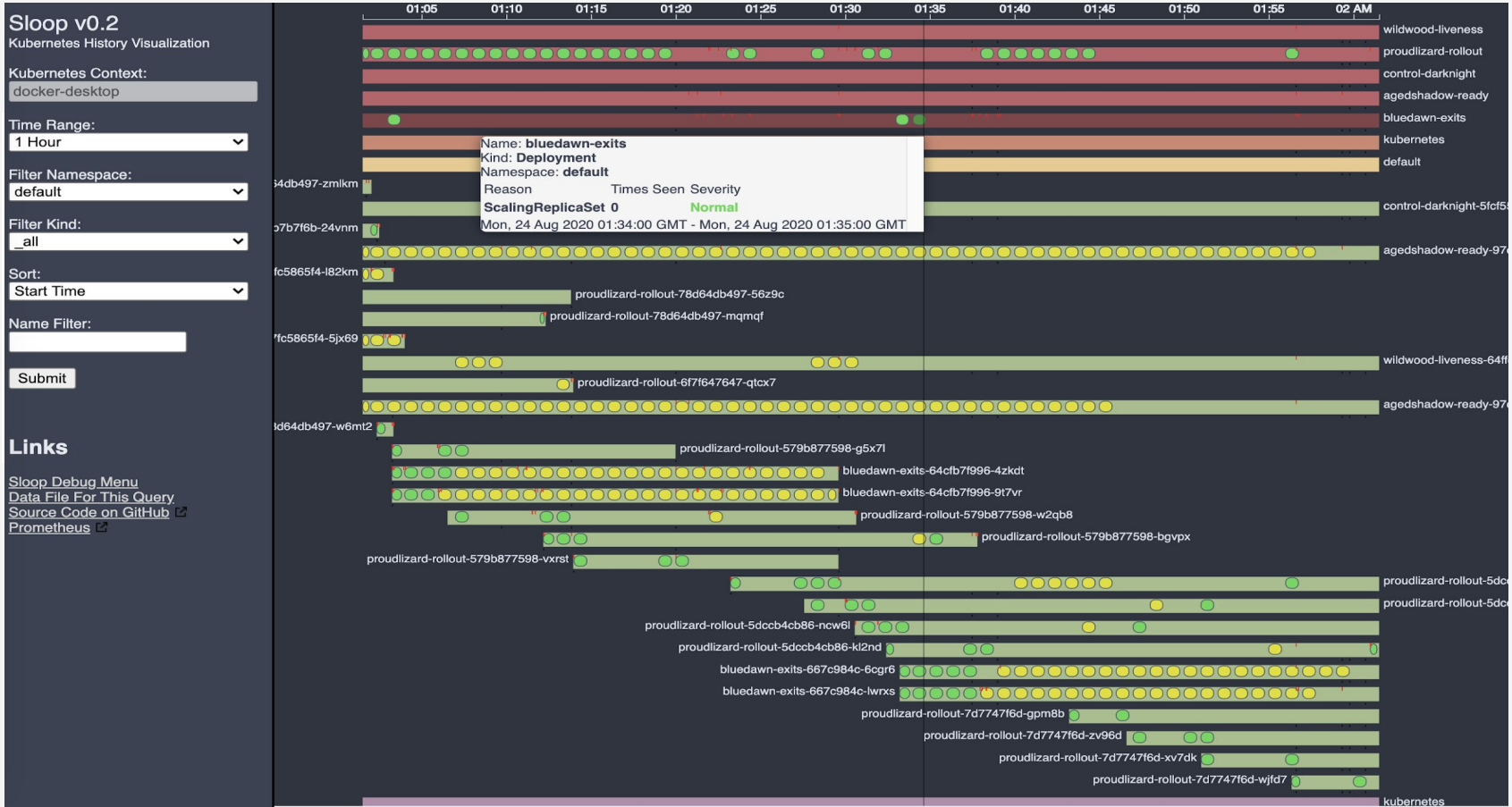
Display k8s resource life cycle along with timestamps



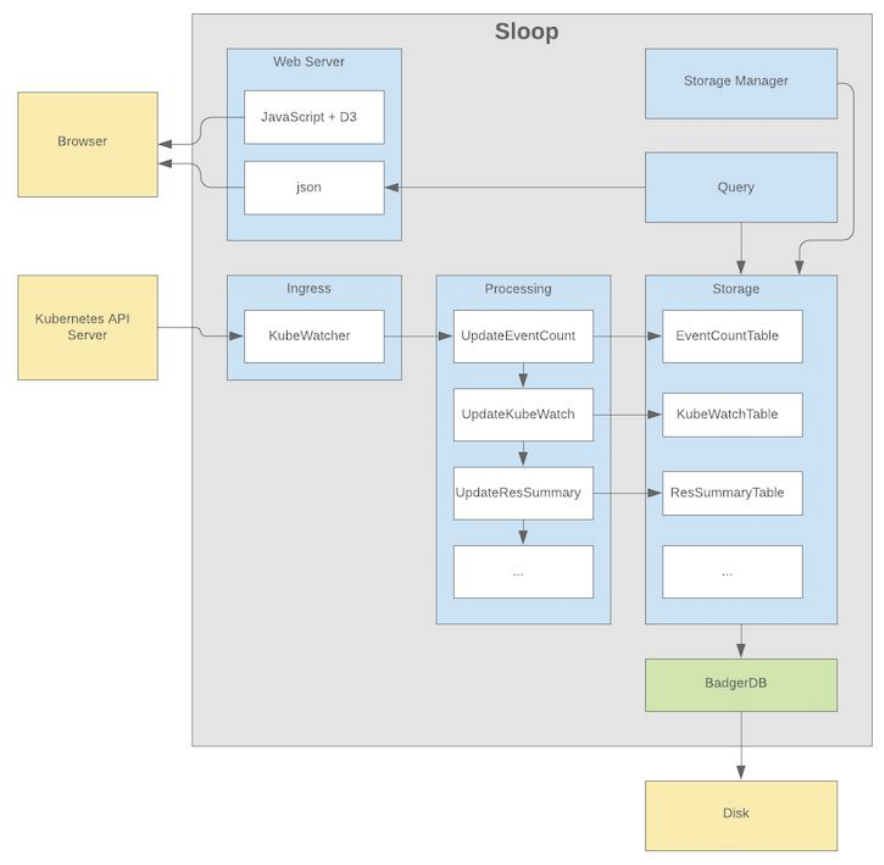
Display the relationships of K8s resources, especially for related resources

Bar charts and diagrams are more visible and straightforward

Sloop


Sloop v0.2
Kubernetes History Visualization
Kubernetes Context:
docker-desktop
Time Range:
1 Hour
Filter Namespace:
default
Filter Kind:
_all
Sort:
Start Time
Name Filter:
Submit
Links
Sloop Debug Menu
Data File For This Query
Source Code on GitHub
Prometheus
01:05
01:10
01:15
01:20
01:25
01:30
01:35
01:40
01:45
01:50
01:55
02 AM
wildwood-liveness
proudlizard-rollout
control-darknight
agedshadow-ready
bluedawn-exits
kubernetes
default
Name: bluedawn-exits
Kind: Deployment
Namespace: default
Reason
Times Seen
Severity
ScalingReplicaSet
0
Normal
Mon, 24 Aug 2020 01:34:00 GMT - Mon, 24 Aug 2020 01:35:00 GMT
proudlizard-rollout-78d64db497-56z9c
proudlizard-rollout-78d64db497-mqmqf
proudlizard-rollout-6f7f647647-qtcx7
proudlizard-rollout-579b877598-g5x7l
bluedawn-exits-64cfb7f996-4zkdt
bluedawn-exits-64cfb7f996-9t7vr
proudlizard-rollout-579b877598-w2qb8
proudlizard-rollout-579b877598-bgvpx
proudlizard-rollout-579b877598-vxrst
proudlizard-rollout-5dccb4cb86-ncw6l
proudlizard-rollout-5dccb4cb86-kl2nd
bluedawn-exits-667c984c-6cgr6
bluedawn-exits-667c984c-lwrxs
proudlizard-rollout-7d7747f6d-gpm8b
proudlizard-rollout-7d7747f6d-zv96d
proudlizard-rollout-7d7747f6d-xv7dk
proudlizard-rollout-7d7747f6d-wjfd7
kubernetes

Sloop architecture




salesforce
Incident # 1

Landlord kicks out greedy tenant

Resource exhaustion

- **Symptom** - Service down for a few hours due to pod restarts

- **Root cause:** Service's Leader pod would exhaust ephemeral storage (emptyDir backed by node's root volume) on node and gets evicted.
- This would happen to each pod that takes over as leader
- It happens for sometime when a scheduled job runs at midnight and the problem stops after the scheduled job is complete

- **Challenge:** Kubernetes events to pinpoint the issue right away are gone from cluster.
- We need to sift through control plane log files and correlate events

- **Sloop to the rescue:** Choose history of events to include the activity time and identify the root cause in seconds by browsing the events with couple of clicks


salesforce
Incident # 2

2 competing actors

Kubernetes vs Cloud provider

- **Symptom** - Unexpected restarts of pods in a statefulset using persistent volumes (bound to availability zones) and service degraded for a few mins

- **Root Cause**: Service was running on nodepool that is spread across multiple AZs.
- Cloud provider was at limit in an AZ
- When a node was terminated for maintenance, Cloud provider tried to compensate from a different zone
- Then when the capacity was restored in the zone, actions to auto-balance nodes across zones terminated nodes running service pods in remaining zones

- **Challenge:** K8s events rolled over from the cluster by the time debugging started. Sifting through control plane and auto-scaler logs is painful
- **Sloop to the rescue:** In a single view, we can see all the nodes that got terminated, at what times they were terminated and correlate which pods they affected, etc in seconds


salesforce
Incident # 3

Poor pods without a home for sometime

The dreaded pod state of Pending

- **Symptom** - Pods were Stuck in pending state for sometime and broke Continuous Delivery automation

- **Challenge** : Well, you get the point by now...

- **Sloop to the rescue** : See a bunch of yellow pills in Sloop for the service pods and click on them to determine why pods were in Pending state during the time.
- Gather all information needed to proactively avoid this from happening next time
  - Unavailable webhooks
  - FailedMounts
  - Nodes not meeting resource constraints
- All this with a few clicks

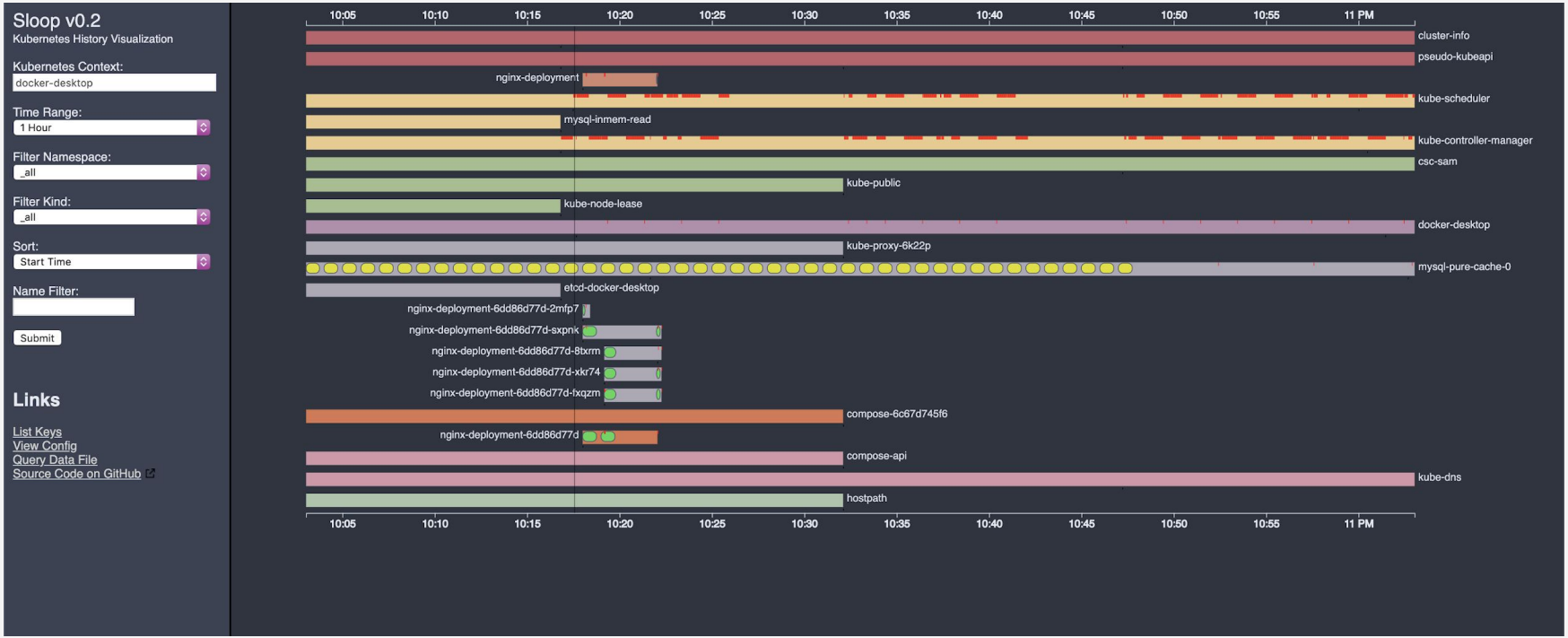
Summary

- Sloop for Kubernetes history visualization
- Come join us in our journey!


thank you
BLAZE YOUR TRAIL
salesforce

Sloop

Historical Kubernetes Visualization



War story #4 - Unhealthy pods (no, not COVID)

Failed/hanging probes

- **Symptom** - Service unavailable. Pods were in CrashloopBackoff
- **Root cause:** Liveness probe failures caused the primary container in pods to be restarted
- **Flavors of probe failures:** Broken dependencies, hung exec commands
- **Sloop to the rescue:** If the issue is intermittent, Sloop can show the history of probe failures