

salesforce

Operating enterprise grade Kubernetes clusters at Salesforce on bare metal

Kubecon + CNCF Europe 2020

Anubhav Dhoot, Senior Director

@anubhavdhoot | adhoot@salesforce.com

Mayank Kumar, Architect

@krmayank | mayank.kumar@salesforce.com



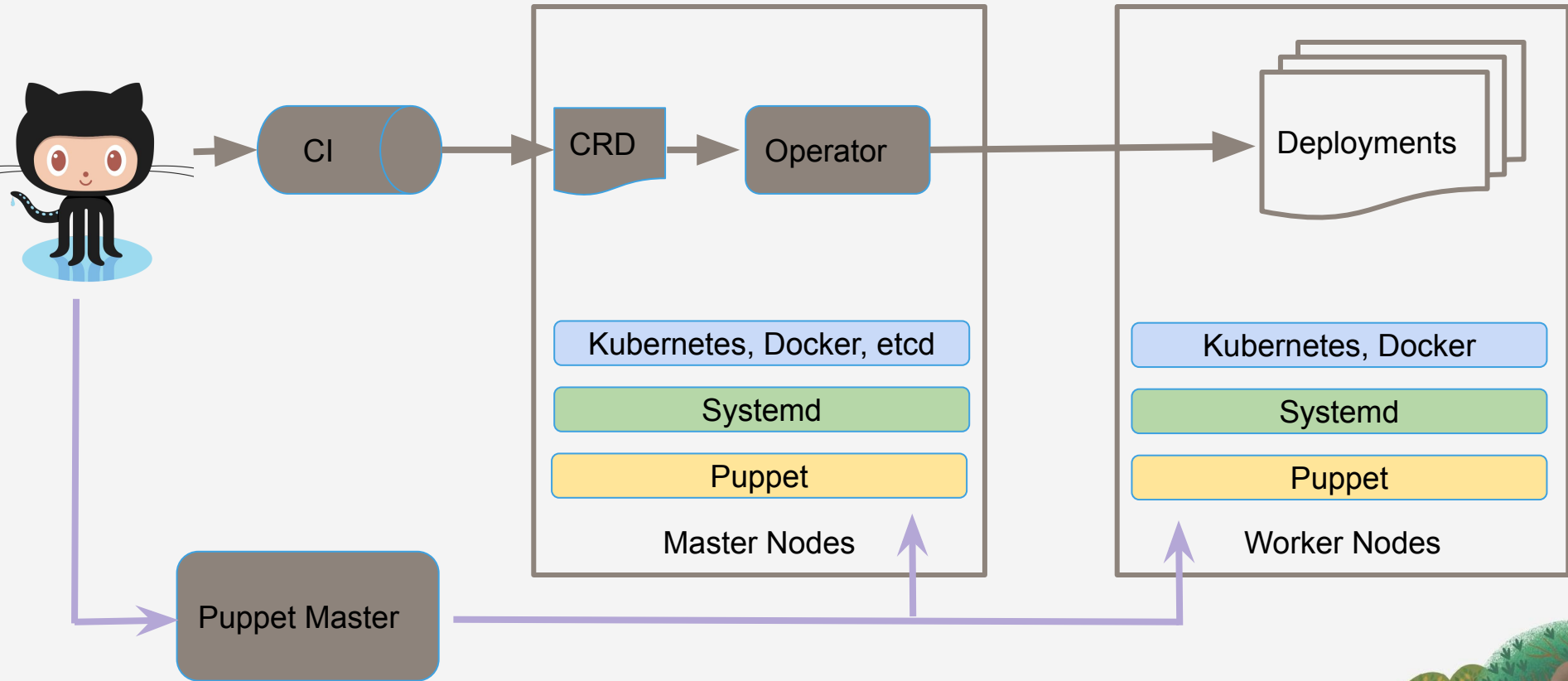
Introduction



- Who is this talk for?
- Agenda
 - Deep dive into our bare-metal Kubernetes implementation
 - War stories
 - Current and future investments



Overview of Salesforce on-premise Kubernetes platform



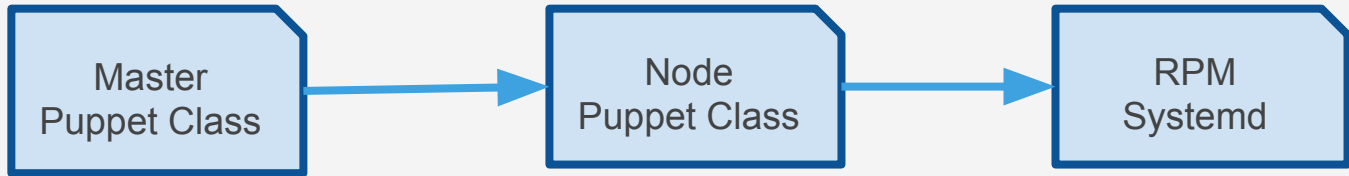
Puppet module deep dive



Goals: Fully automated management with high availability and security built in

Usage:

```
class{ 'kubernetes':  
    master => true  
}
```



- Etcd
- ApiServer
- Controller Manager
- Scheduler

- Docker
- kubelet
- kube-proxy
- Flannel/SDN
- HAproxy

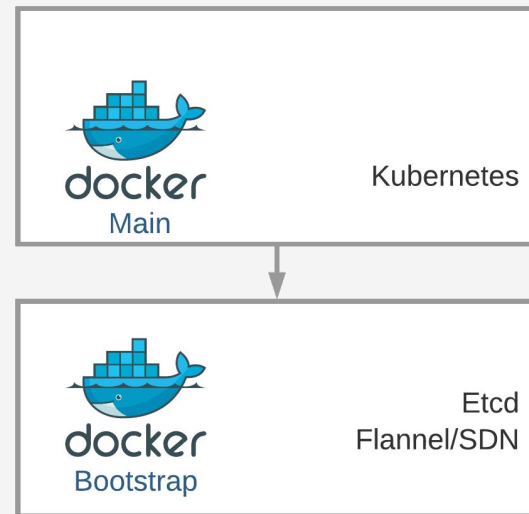
- Binaries
- Configuration



Puppet module features





- Completely automated deployment of etcd, docker and kubernetes
- Configuration flags for services
- Etcd service waits for quorum
- Service account supports key rotation
- Additional Docker instance for running bootstrap services
 - Docker needs flannel which we needed to run in Docker
 - Flannel also needs etcd



Kubernetes on Puppet review



- 
 - Its automated
 - Declarative
 - Can stagger across machines

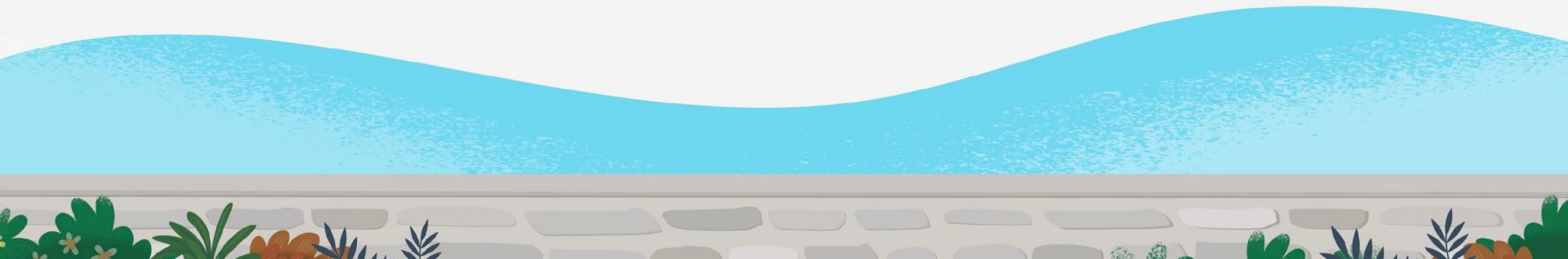
- 
 - No health mediation or orchestration
 - Iteration cycles are expensive
 - Base module changes can break your module



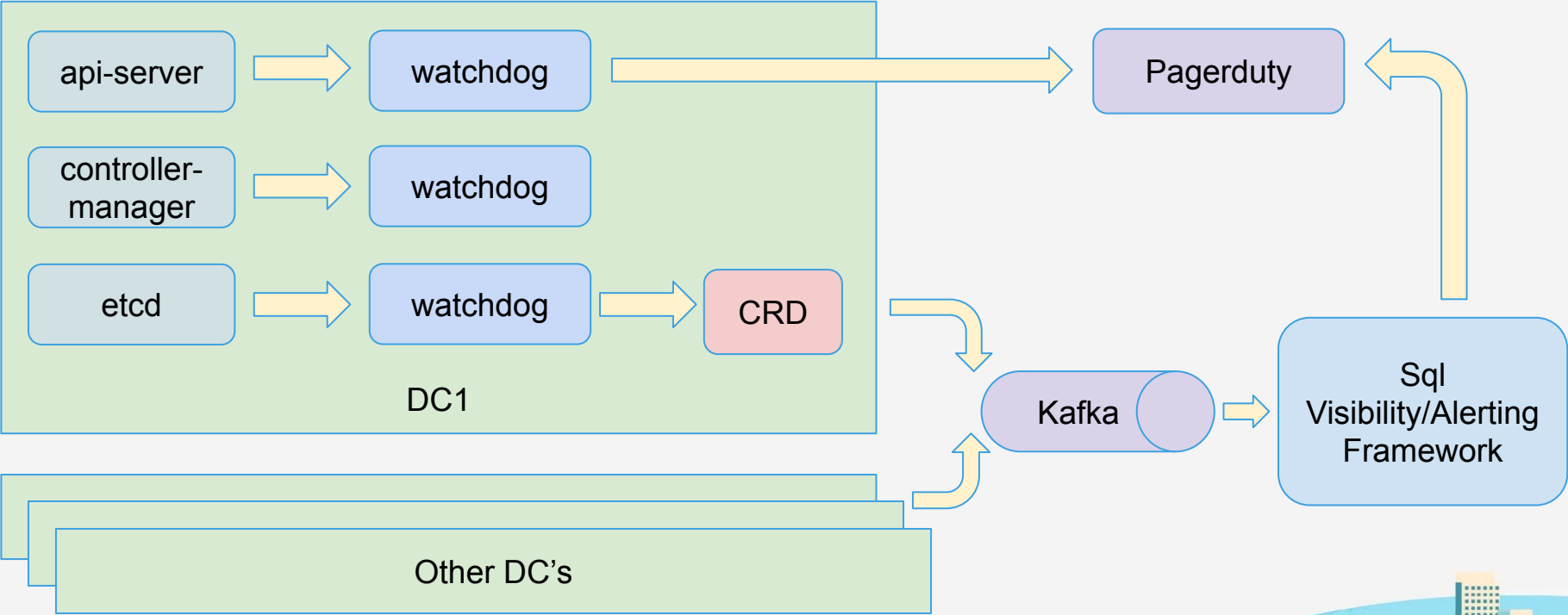
Operationalize



- Etcd and Kubernetes control plane high availability setup
 - HAproxy for supporting multiple api servers
- Security hardening
 - mTLS for all communication
 - Security isolation
- Monitor everything
 - Watchdogs that monitor and alert
 - SQL based monitoring pipeline that provides snapshot visibility and custom alerting
 - [Sloop](#) for historical visibility
- Automate operations



Monitoring bare metal Kubernetes



War stories



War story #1 - Perils of mounting hostPath

Dangerous knobs in kubernetes

- **Symptom** - Pods were stuck in ContainerCreating for a long time

- **Root cause:** Some pods were mounting the root filesystem(/). (/) includes folder /var/lib/kubelet where kubelet mounts emptyDir and secrets
- During pod deletion the tear down of mounted volumes fails because its mounted inside the problematic container.
- This prevents a new pod from coming up

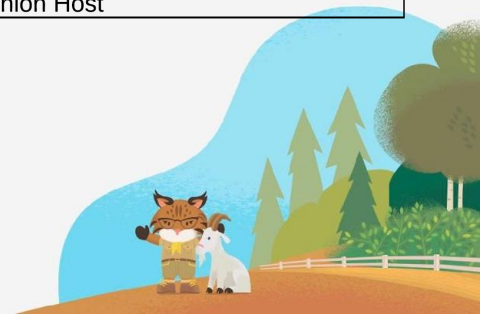
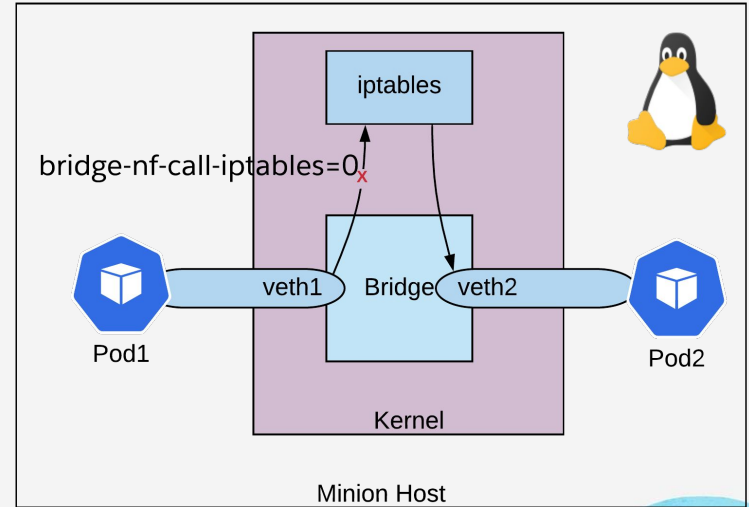
- **Fix:** Dont mount root, validate what hostPaths are allowed
- **Learning:** hostPath is dangerous and should be avoided. This was at the time when Local volumes were still being designed



War story #2 - Bridge networking failures

Network troubleshooting is hard

- **Symptom** - Intermittent connectivity failure for microservices communicating through a Kubernetes service
- Deeper analysis pointed to failures only when client and server pods are on the same host
- **Root Cause:** some team set `bridge-nf-call-iptables=0` as an optimization
- This skips iptables for packets traversing the bridge which broke kube-proxy iptables rule.
- **Learning:** You should invest in robust network monitoring



War story #3 - Bullying quorum members

Etcd configuration challenges

- **Symptom** - Etcd state was wiped out in our R&D cluster
- 2 out of the 3 nodes got reinitialized with empty state and since they were in quorum, convinced the third to replicate empty state
- **Fix** : Change the etcd flags from new to existing after “initialization period”
- **Learnings** : Etcd doc is not clear on the dangers of the flags (see screenshot below). do regular backups of etcd data, game day exercises

-initial-cluster-state

- Initial cluster state (“new” or “existing”). Set to `new` for all members present during initial static or DNS bootstrapping. If this option is set to `existing`, etcd will attempt to join the existing cluster. If the wrong value is set, etcd will attempt to start but fail safely.
- default: “new”
- env variable: `ETCD_INITIAL_CLUSTER_STATE`



War story #4 - Handle sharp tools with care

Dangers of mutating webhooks

- **Symptom** - R&D cluster control plane servers going down with memory leak
- No obvious usage of memory other than excessive number of pods
- **Short term fix:** Restart each ApiServer on memory limit to prevent cluster tipping over

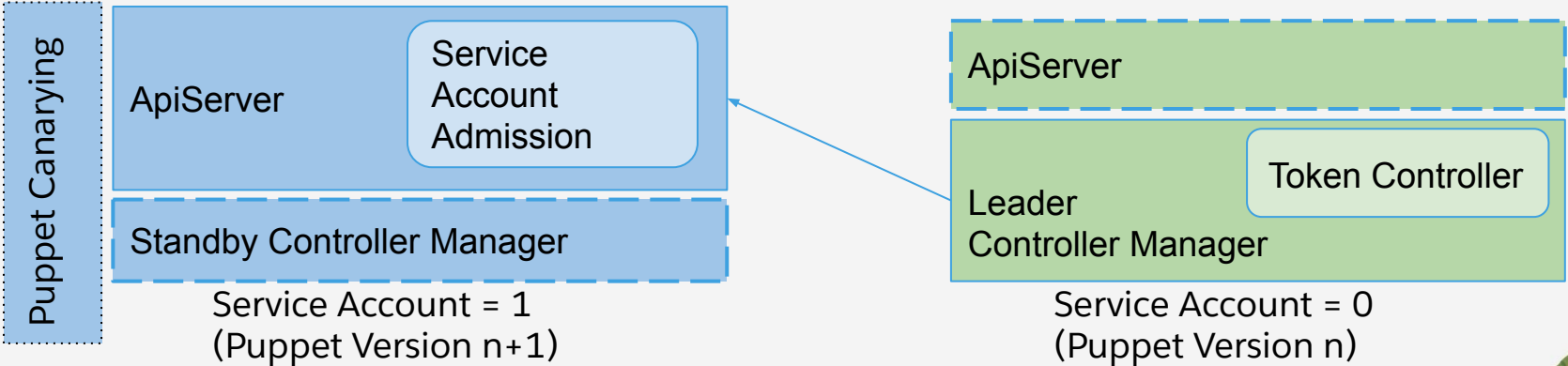
- **Root cause:** A new Mutating webhook admission controller was leaking pods
- The json patch in trying to add a label, dropped all other labels on the pod being created, thus preventing the deployment controller from adopting the pods
- **Detection:** Alert on number of pods
- **Long term mitigation:** Proper json patch and Systemd memory limits for Apiserver
- **Learnings:**
 - Mutating webhook admission controllers need validation and canarying
 - Adding limits is better than making the node unusable



War story #5 - Inconsistent Api Servers flags

Canarying feature flags in Kubernetes

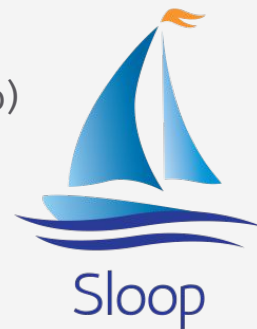
- **Symptom** - ReplicaSet controller was failing to create pods
- **Root cause:** Puppet canarying of service account flags across masters
- We did not catch in testing as this failure depends on combination of canarying and leader election across masters
- **Fix:** Synchronize the rollout of Api server and controller manager flags everywhere
- **Learning:** Staged rollout does not always work when rolling out new feature flags



Summary



- Roll your own kubernetes requires a lot of expertise and investments
- We have invested in the following
 - Fully automated, highly available and secure, Docker, Etcd and Kubernetes infrastructure
 - Integrations with networking, security, monitoring (logs, metrics, alerts)
 - Watchdog monitoring and visibility pipeline
 - Cost-to-serve visibility at container, namespace and team
 - Robust on call rotation for infrastructure and runbooks
- Ongoing and Future projects
 - PaaS layer for containers and other cloud resources (DB, Blob)
 - Open source more of our investments
- Open source projects
 - [Blog](#) describing our [Generic sidecar injector](#)
 - [Sloop](#) for Kubernetes history visualization
- Come join us in our journey!



Thank
you

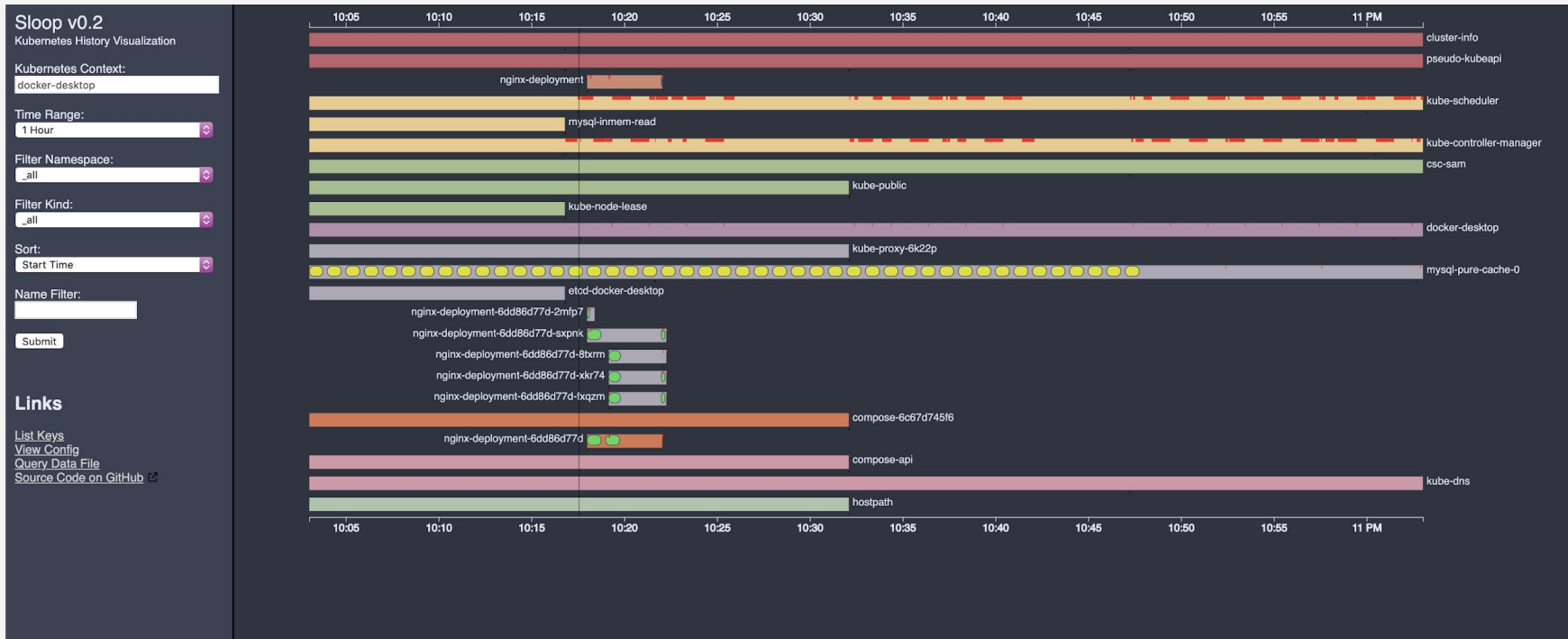
BLAZE
YOUR
TRAIL

salesforce



Sloop

Historical Kubernetes Visualization



Json Patch (War story 4)



```
{
  "op": "add",
  "path": "/metadata/labels",
  "value": {
    "sherpa-injector.service-mesh/status": "injected"
  }
}
```

Wrong Patch

```
{
  "op": "add",
  "path": "/metadata/labels/sherpa-injector.service-mesh~1status",
  "value": "injected"
}
```

Correct Patch