CONTINO

Programmable Infrastructure with Kubernetes

Hibri Marzook and Talieson Sisson

Whoami

Hibri Marzook

hibri.marzook@contino.io
@hibri
http://github.com/hibri

Taliesin Sisson

talieson.sisson@contino.io
http://github.com/taliesins

Agenda

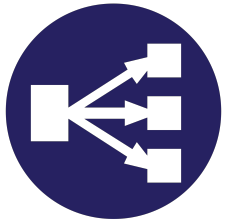
1. Complexity in distributed systems (5 min)
2. The Reconciliation Loop (5 min)
3. Extensibility in K8S (5 min)
4. Building a Custom Controller with Go (15 min)
5. What have others done (5 min)
6. A Service Mesh - Istio (5 min)
7. Q&A (10 min)

Something is always *brokn*

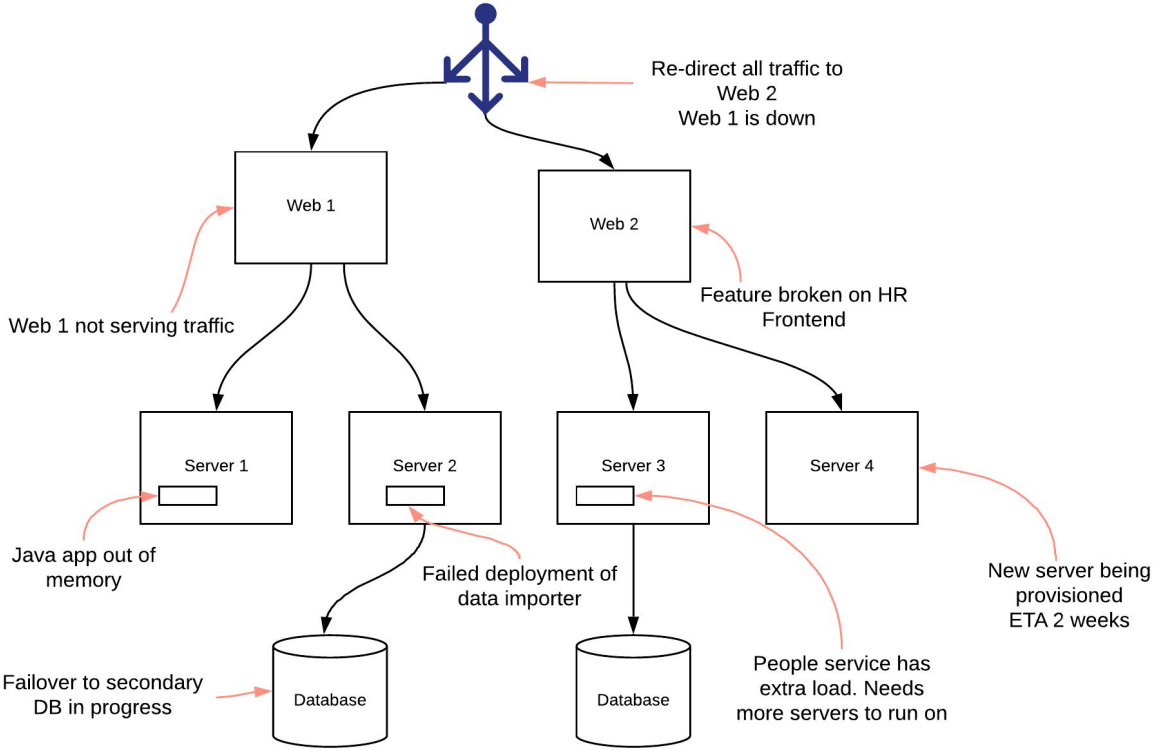
CONTINO

Infrastructure has always been complex

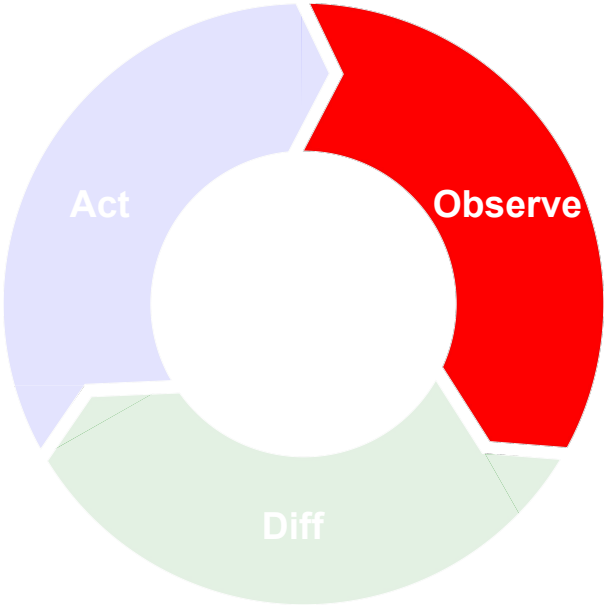
- Need a hodgepodge of different tools to build what we need
- This leads to a multitude of tooling and code bases
- Multi component systems have always been hard to predict and control
- We have to context switch between different abstractions and tooling to build and troubleshoot infrastructure



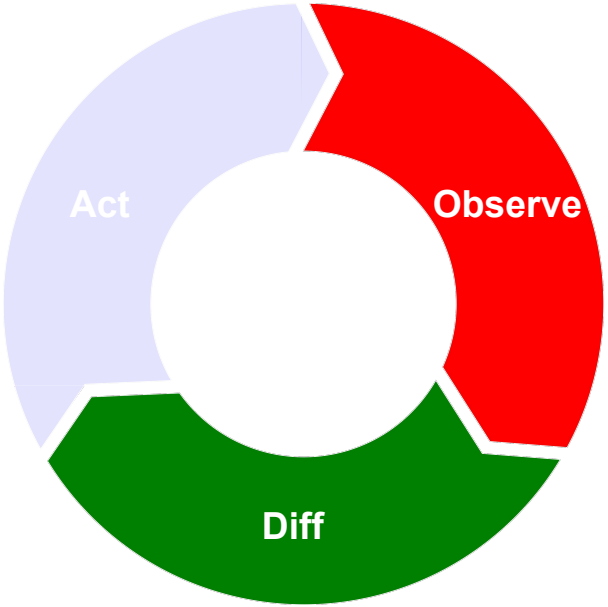
Something is always broken

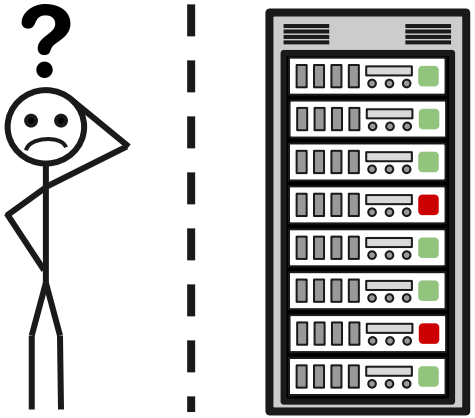


The Human Reconciliation Loop

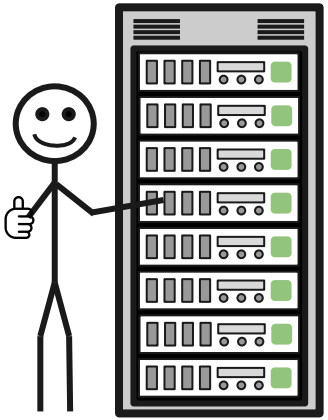


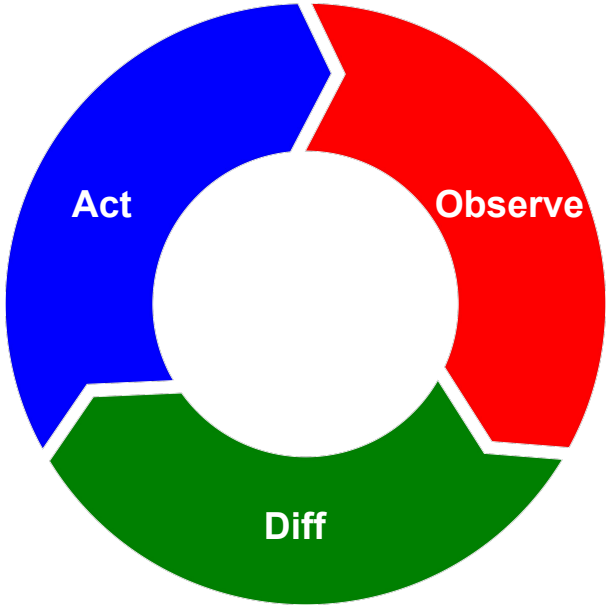
The Human Reconciliation Loop

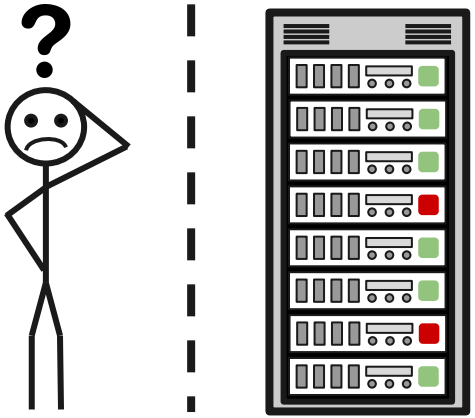




The Human Reconciliation Loop







Automated Reconciliation Loop

Start Loop

Event

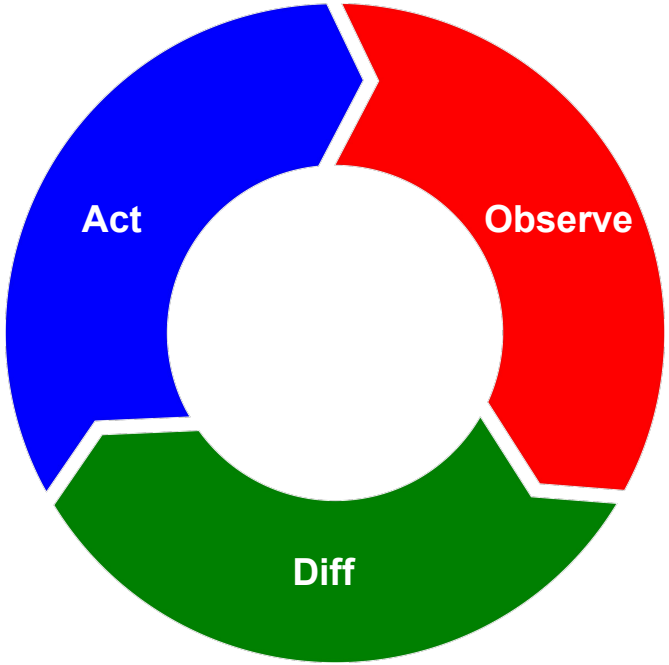
Observe - What happened

Event Handler

Diff - Decide what actions to take

Act - Take action

Repeat Loop



Reconciliation in Kubernetes

There are three states of the world;

An Idealized desired state which is a declarative statement of what the world should be like

An actual state the actual state of the system.

A current state which approximates the actual state, and might be noisy, incomplete, or out of date.

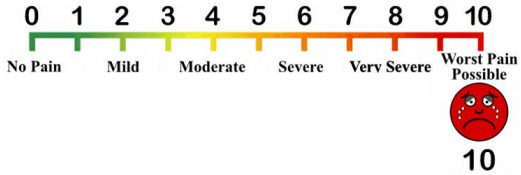
The role of the reconciliation loop is to repeatedly compare the **current state** against the **desired state**, and take action to drive the **actual state** to match the **desired state**

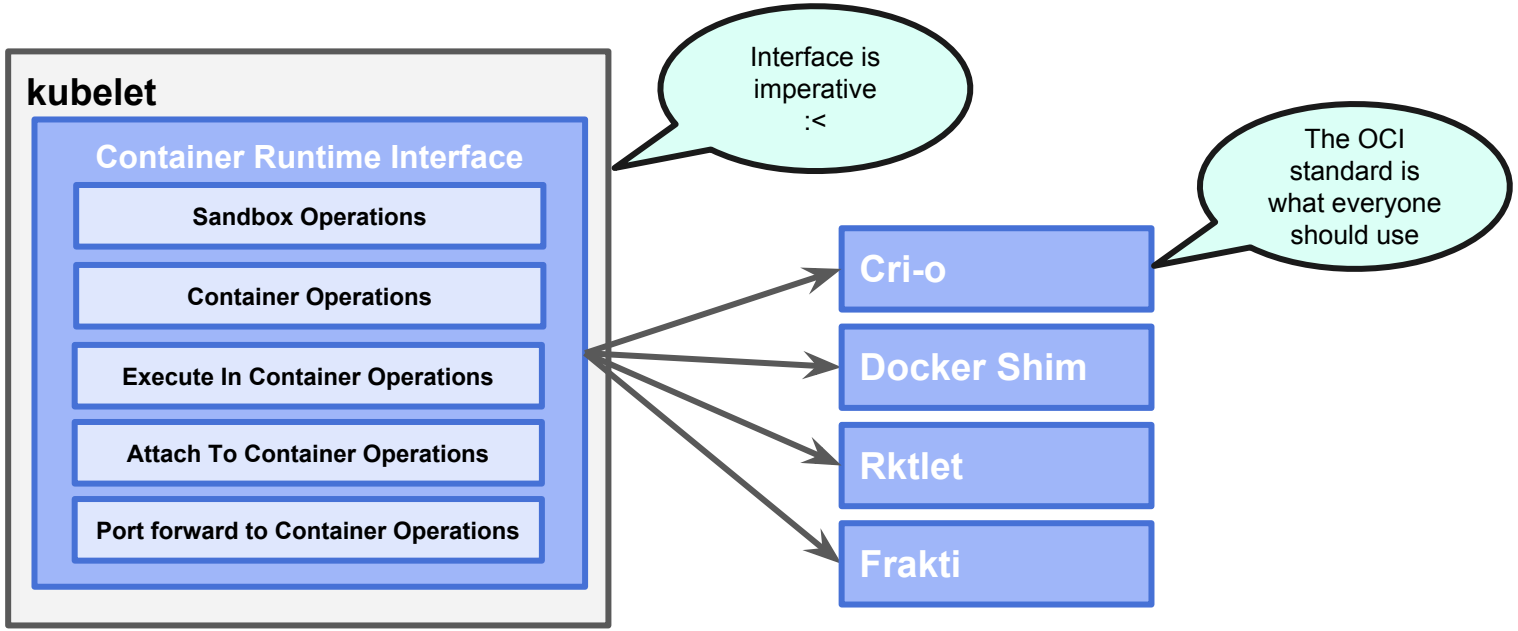
Brendan Burns - How Kubernetes Changes Operations, Login Magazine, October 2015

Extensibility in K8S

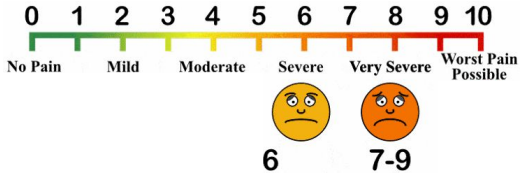
CONTINO

Container Runtime Interface





Cloud controllers



controller-manager

Config for cloud provider to use

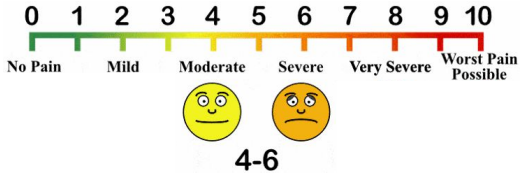
cloud-controller-manager

- CloudNodeController
- PersistentVolumeLabelController
- service controller
- Route controller

Not all cloud managers are pulled out of Kubernetes master yet

- **azure-cloud-controller-manager**
- **digitalocean-cloud-controller-manager**
- **openstack-cloud-controller-manager**

Custom controllers



Pod
extension-apiserver
Controller

Pod
Controller

Pod
Controller

proxied API traffic
custom resource
crd resource
built in resource
services, pods etc

kube-api-server
Custom API
Custom Resource Definitions
Built In Resources
Web Hooks
Mutating Admission Webhook
Validating Admission Webhook

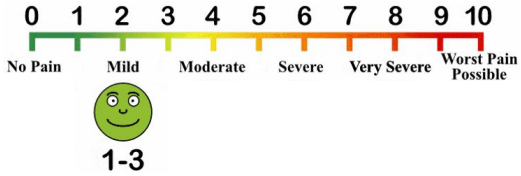
ingress resource
webhook
webhook

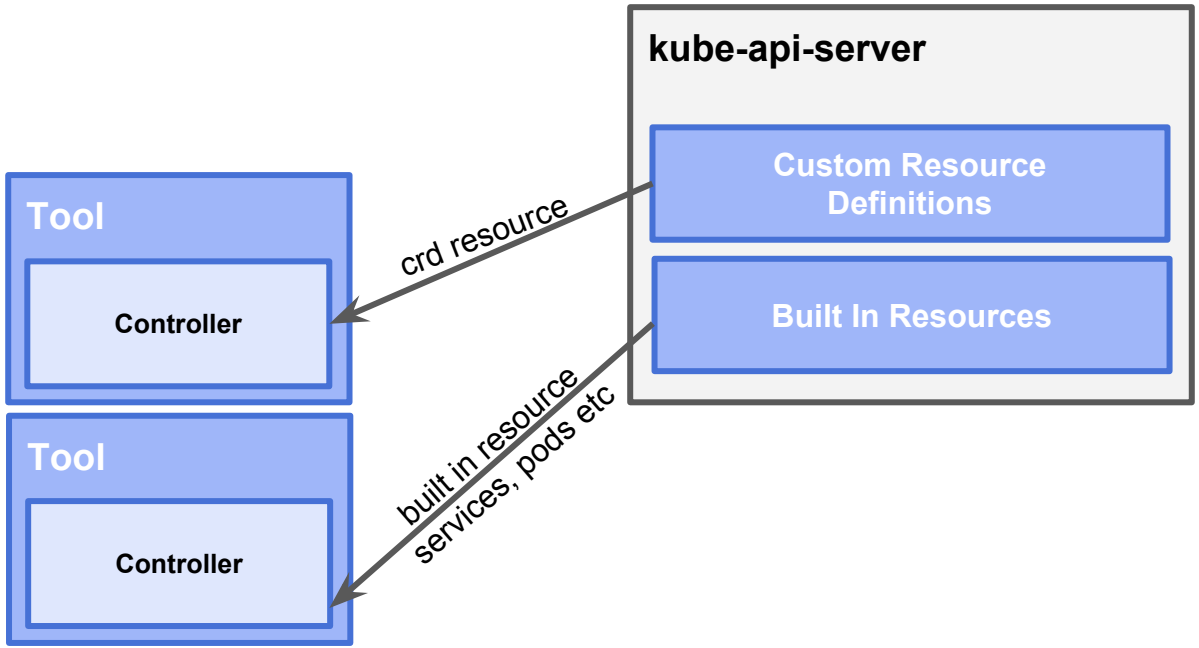
Pod
Ingress Controller

Pod
REST API

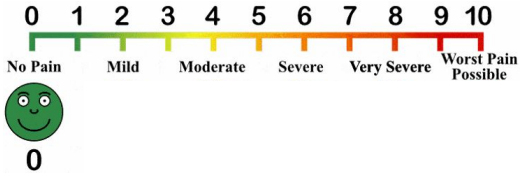
Pod
REST API

Command line tool





Labels & Annotations



Labels

- Identifying information
- Can be used for queries
- Each key must be unique for the object
- Restricted size (63 chars)

Annotations

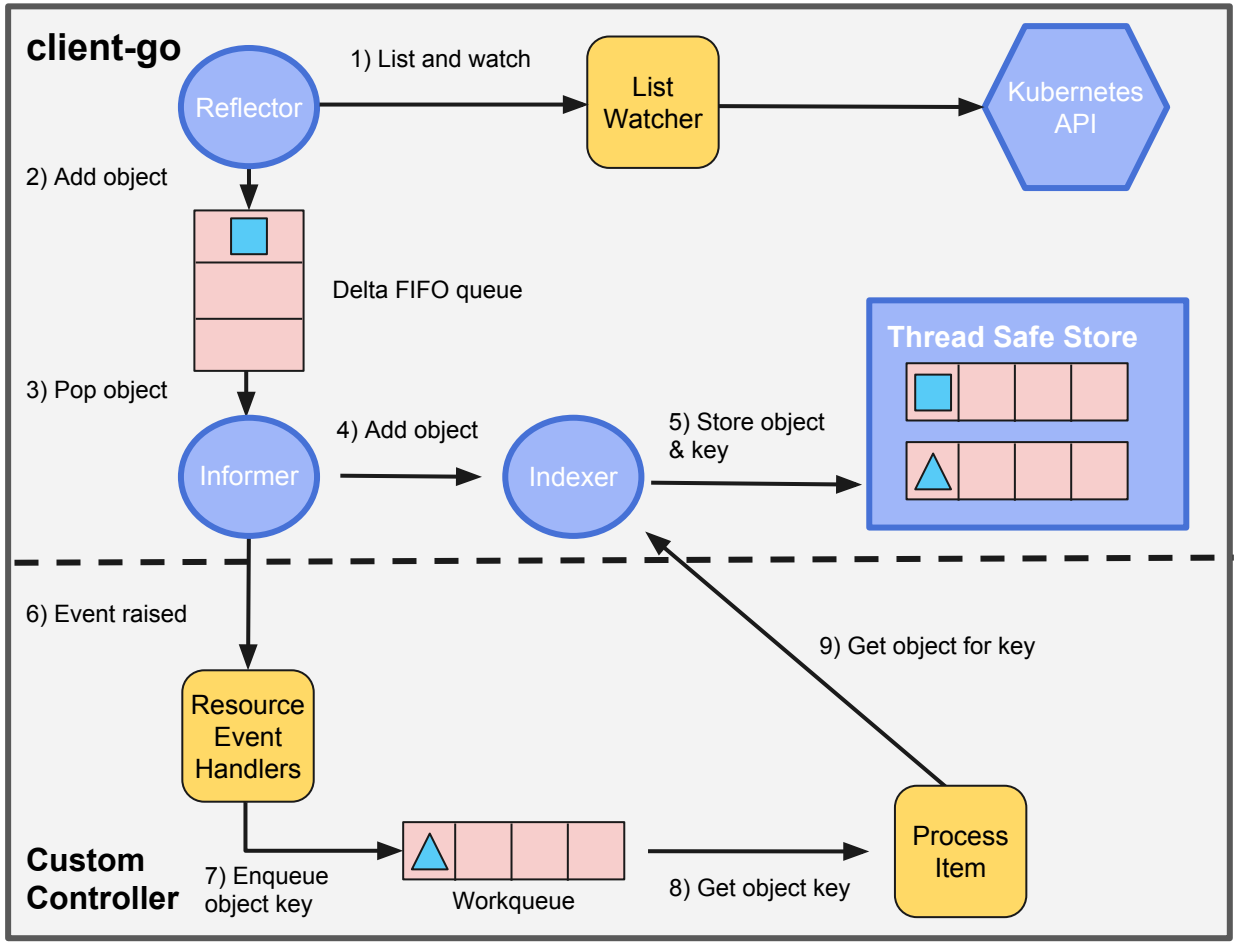
- Non-identifying information
- Can be used for queries
- Duplicates keys are allowed for the object
- Unrestricted size

```
"metadata": {
  "labels": {
    "cd" : "blue",
    "owner" : "Team Fox"
  }
}
```

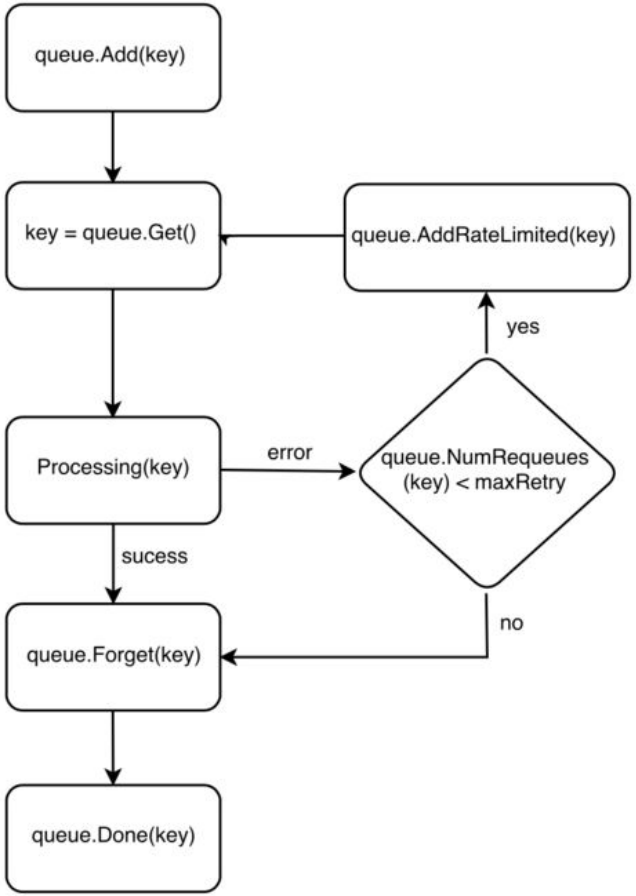
```
"metadata": {
  "annotations": {
    "releaseNotes" : "Fixed ie 6 compatability",
    "qaTests" : '{"results": {"passed": "99", "failed":”1" }}'
  }
}
```

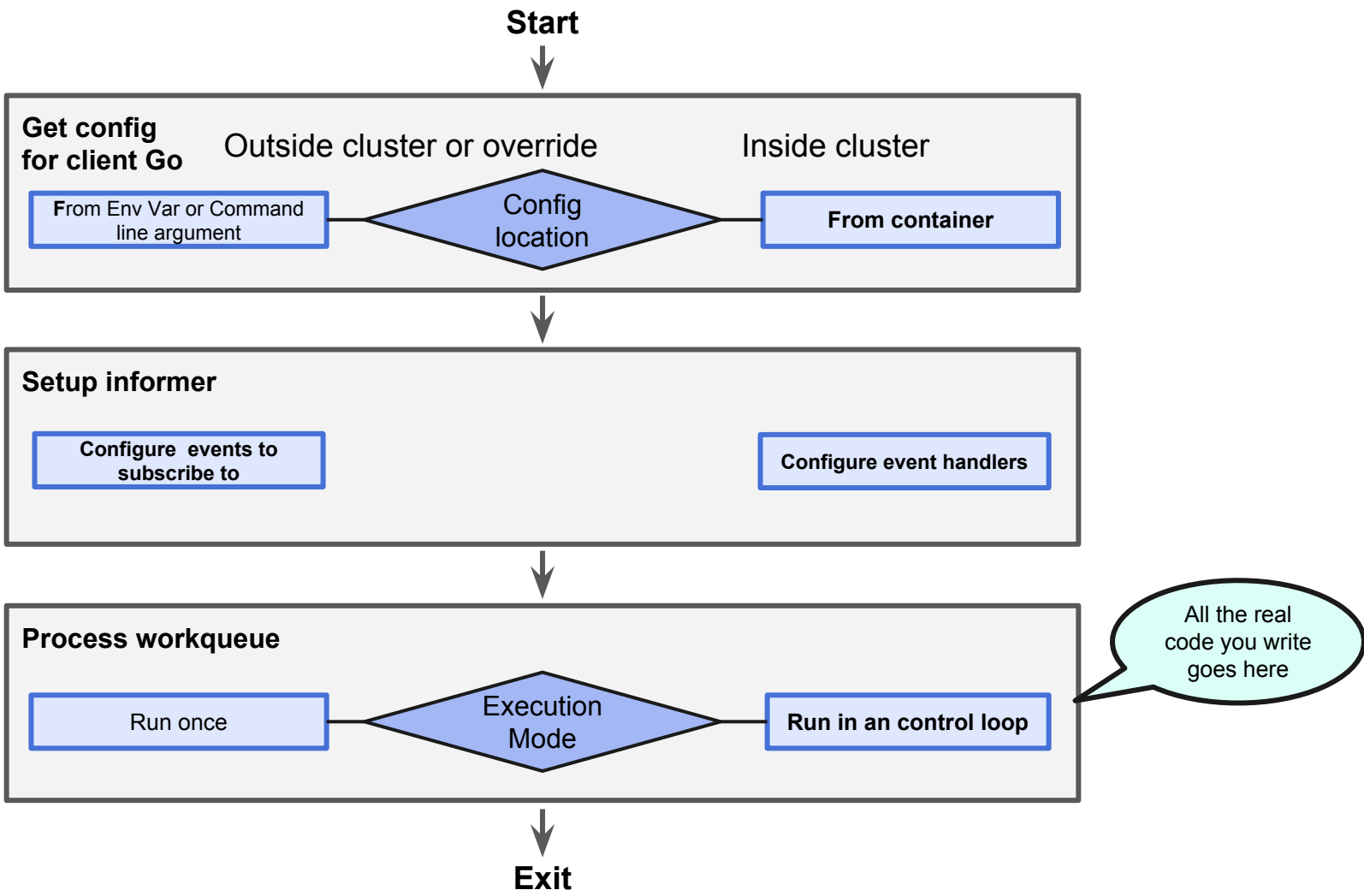
Make your own automated reconciliation loop

CONTINO


client-go
Reflector
1) List and watch
List Watcher
Kubernetes API
2) Add object
Delta FIFO queue
Thread Safe Store
3) Pop object
Informer
4) Add object
Indexer
5) Store object & key
6) Event raised
9) Get object for key
Resource Event Handlers
Custom Controller
7) Enqueue object key
Workqueue
8) Get object key
Process Item

Workqueue


queue.Add(key)
key = queue.Get()
queue.AddRateLimited(key)
yes
Processing(key)
error
queue.NumRequeues
(key) < maxRetry
sucess
no
queue.Forget(key)
queue.Done(key)


Start
Get config for client Go
Outside cluster or override
Inside cluster
From Env Var or Command line argument
Config location
From container
Setup informer
Configure events to subscribe to
Configure event handlers
Process workqueue
Run once
Execution Mode
Run in an control loop
All the real code you write goes here
Exit

Building a Custom Controller that uses CRDs

Get-Config -> main.go 45 getKubernetesConfig
Setup informers -> controller.go 88 NewController
    Events to listen to -> main.go 95 main
    Events to listen to -> main.go 96 main
    Handlers for events -> controller.go 117 NewController
    Handlers for events -> controller.go 129 NewController
Run informers -> 101 main.go main
    Start x number of consumers in parallel -> controller.go 166 Run
        "Observe" Get item off workqueue -> controller.go 188 processNextWorkItem
            "Diff" retrieve desired state -> controller.go 250 syncHandler
            "Diff" retrieve current state -> controller.go 272 syncHandler
            "Act" create resource -> controller.go 275
            "Act" update resource -> controller.go 298
            "Act" does not have delete resource in this example (often handled by event handler)
        Repeat Until Exit Program
    Until Exit Program
Until Exit Program

What have others done

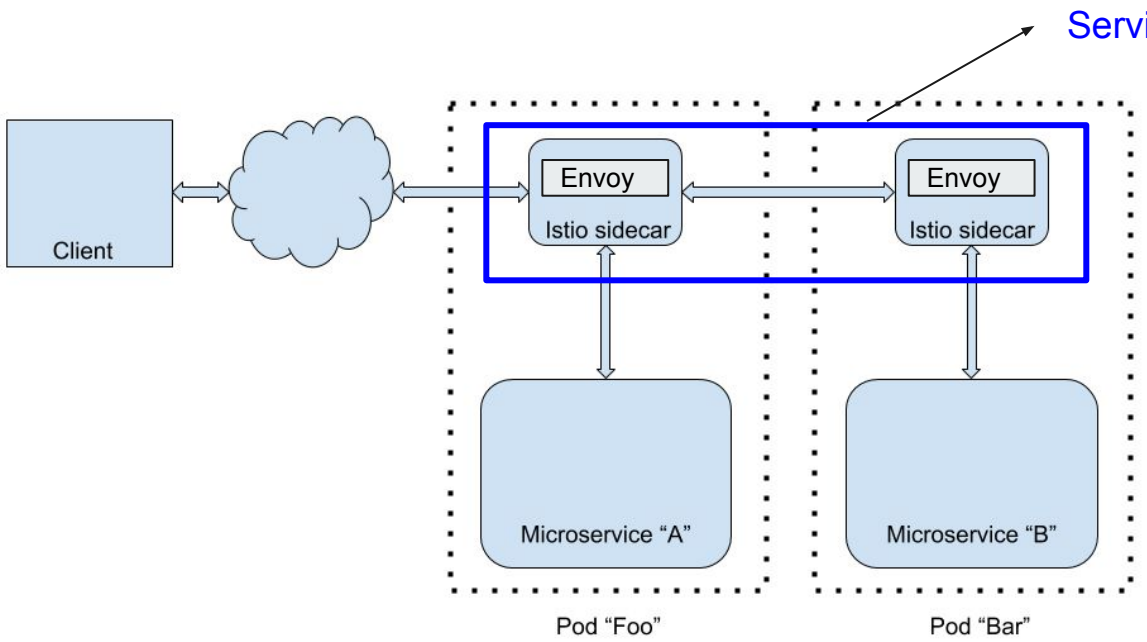
CONTINO

Leveraging the extensibility in K8S

K8S provides abstractions over the underlying infrastructure, we can use these abstractions to build more interesting things. Some of these are

- Service Mesh - An infrastructure layer for inter-service communication
- Ingress Controllers - Do smart things with Layer 7 traffic
- Serverless - If you can orchestrate containers, why not extend it to orchestrate code?
- Operators - Write applications to manage other applications
- Service catalog - Use Kubernetes to deploy non-K8S infrastructure
- Policy as Code - Validate your infrastructure, do code reviews before changes are applied.
- Vulnerability Management & Runtime defense - Build tools to continuously monitor threats and deal with them

A Service Mesh - Istio



Authentication

Retries

Logging

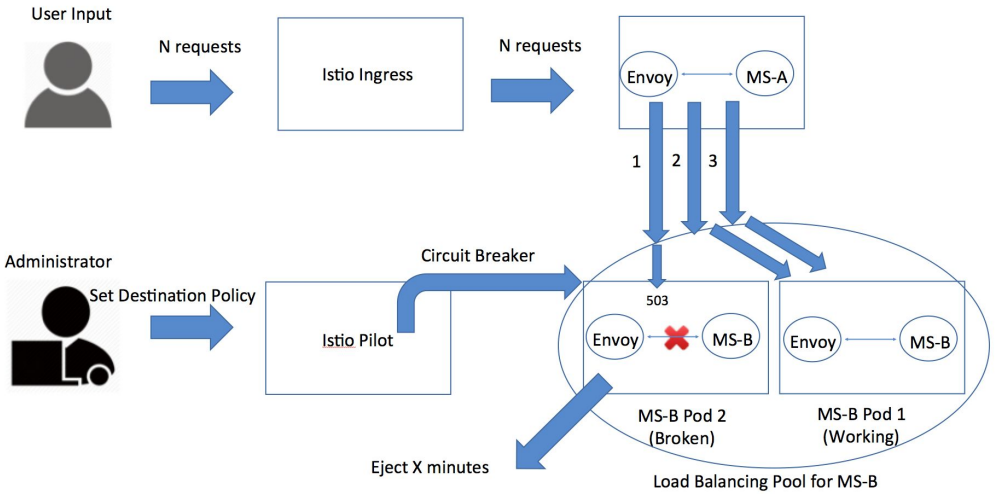
Distributed Tracing

A/B testing

Failure Injection

Circuit Breaking with Istio

```
apiVersion: networking.istio.io/v1alpha3
kind: DestinationRule
metadata:
  name: httpbin
spec:
  host: httpbin
  trafficPolicy:
    connectionPool:
      tcp:
        maxConnections: 1
      http:
        http1MaxPendingRequests: 1
        maxRequestsPerConnection: 1
```



Resources

- https://github.com/kubernetes/community/blob/master/contributors/devel/controllers.md
- https://github.com/kubernetes/community/blob/master/contributors/design-proposals/architecture/principles.md
- https://docs.openstack.org/kuryr-kubernetes/latest/devref/kuryr_kubernetes_ingress_design.html
- Template to create custom controller - https://blog.openshift.com/kubernetes-deep-dive-code-generation-customresources/
- Building controller from scratch (code looks good) - https://medium.com/@trstringer/create-kubernetes-controllers-for-core-and-custom-resources-62fc35ad64a3
- Build controller from scratch - https://www.youtube.com/watch?v=QIMz4V9WxVc
- Walk through kubernetes code - https://www.youtube.com/watch?v=ryeINNfVOi8
- Programming Kubernetes with the Go SDK - https://www.youtube.com/watch?v=qiB4RxCDC8o
- Istio - https://istio.io/

QUESTIONS?

London

1 Fore Street,
Moorgate,
London,
EC2Y 9DT,
UK

london@contino.io

New York

404 5th Avenue,
New York, NY,
10018EC2Y 9DT,
UK

newyork@contino.io

Melbourne

Level 2,
Hub Southern Cross,
696 Bourke St,
Melbourne VIC 3000,
Australia

melbourne@contino.io

@ContinoHQ
@ContinoHQ
Contino

CONTINO

contino.io

info@contino.io