Katalog-Sync

Reliable Integration of Consul and Kubernetes

Me: Thomas Jackson

- Head of Core Infrastructure @ Wish
- Work experience:
  - Network Engineer
  - Corporate IT
  - Small Startups
  - Freelance Work
  - LinkedIn (professional social network)
  - Wish (mobile-first ecommerce platform)

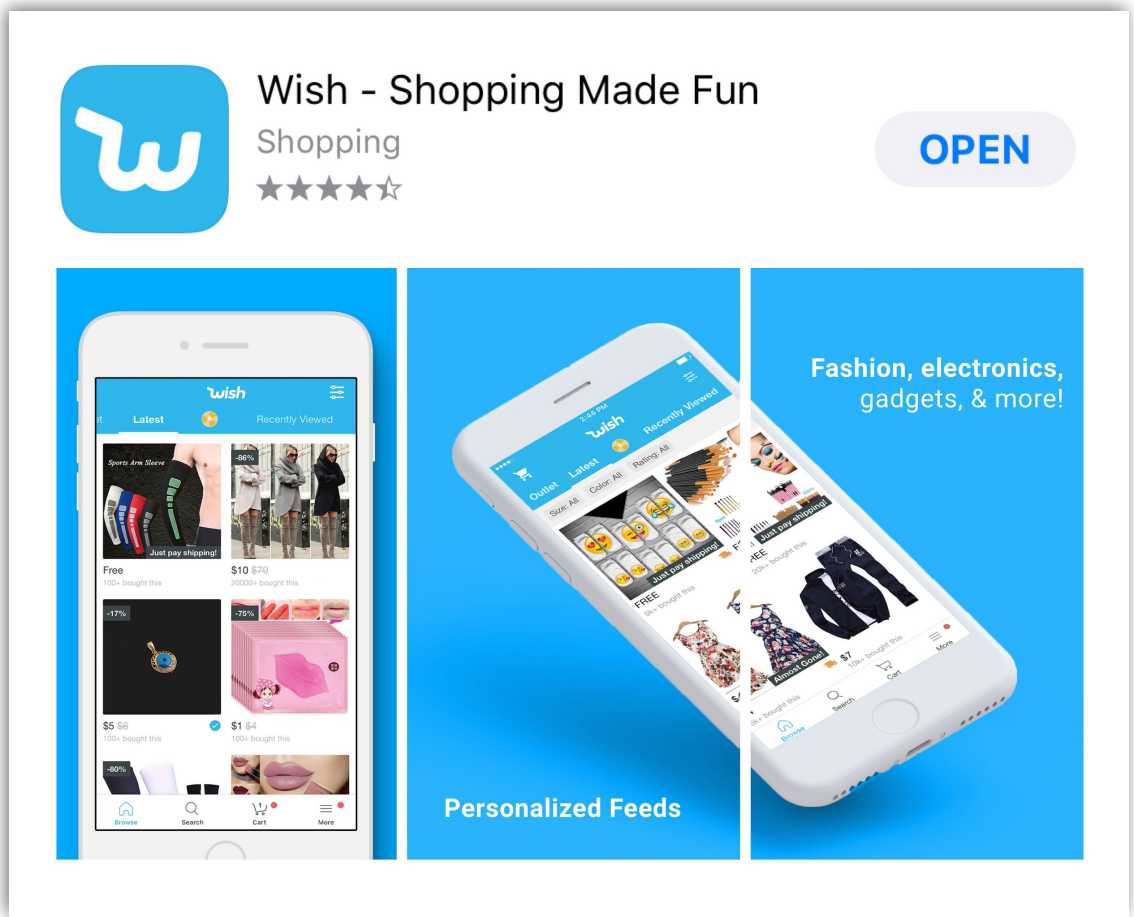
About Us

Who We Are

Leading mobile commerce platform in US and EU.

Our Mission

To offer the most affordable, convenient, and effective mobile shopping mall in the world.



Global Reach

300M+
Registered Users

10M+
Daily Active Users

250K+
Active Shoppers per Day

How this talk will go

- Context
  - There will be memes!
  - Feel free to laugh
  - Please don’t fall asleep (if you do… just don’t snore)
- Agenda
  - K8s: what is it, why do you want it, how do you get it
  - Iterations using consul on k8s: process, design, testing, and results

Why Kubernetes (k8s)?

- First, we should talk a bit about how it was done before

Pre-k8s: The Dark Ages

- What
  - High-level, we want to run apps
  - To accomplish this we manage fleets of servers
- How
  - Configuration management for app deployments (e.g. Chef, salt, ansible, etc.)
  - Tar.gz or package to deploy/revert app

Pre-k8s: The Dark Ages

- Pain points
  - Managing stateful systems (state def needs to account for everything that could happen to a system)
  - Rollbacks are difficult (if not impossible)
  - Coordination is complicated
  - Limited introspection
  - Limited access control
  - Hard to test and review

With K8s: The future!

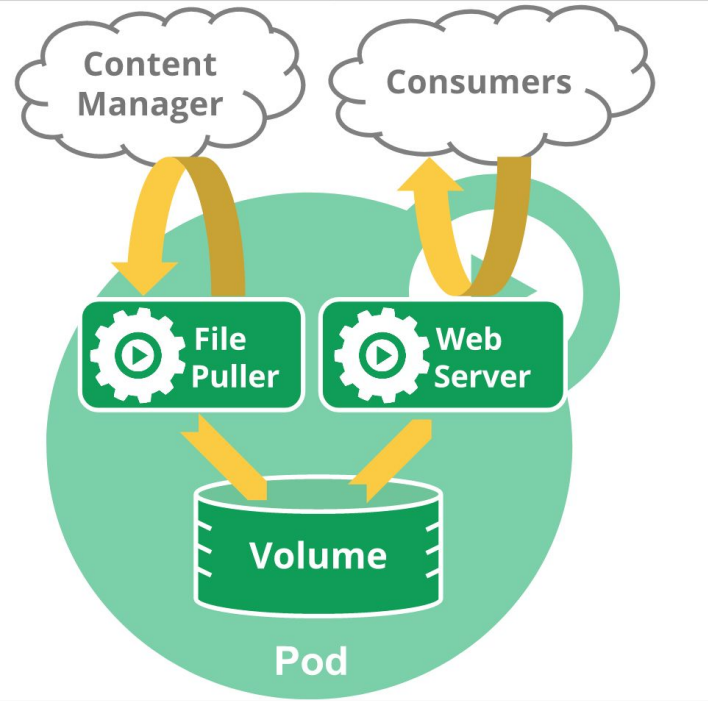
With K8s: The future!

- Declarative state, config, automation
- Rollbacks are exact: just a “push” to previous state
- Great introspection and access control
- Easily tested: containers are “the same” everywhere
- Better abstraction: pods vs “instances”

kubernetes

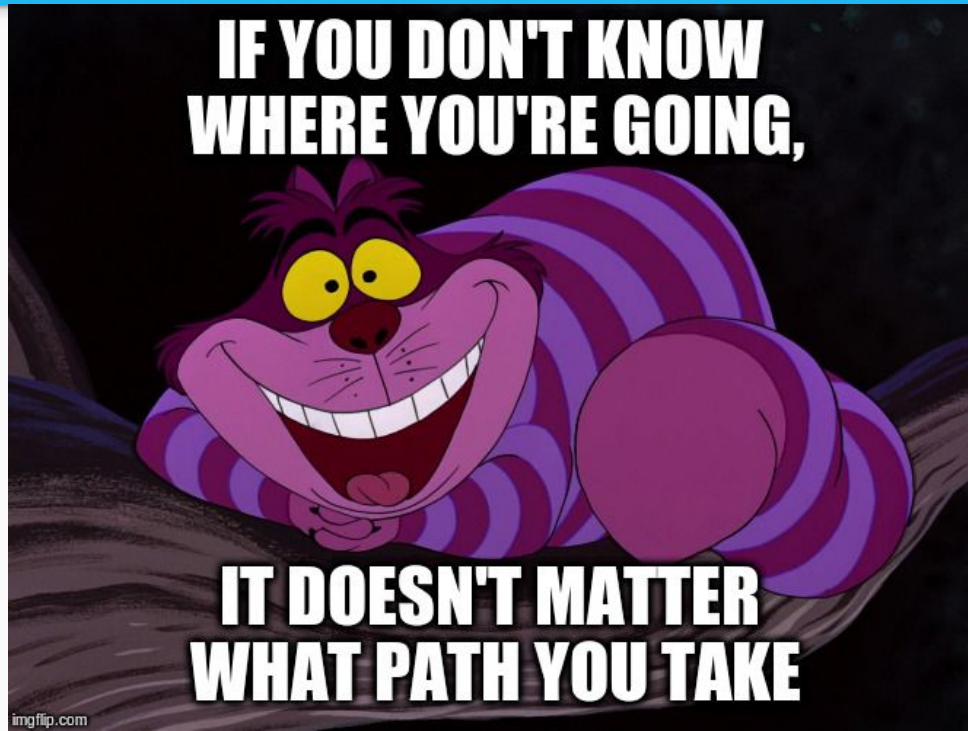
K8s: a crash course (emphasis on crash)

- K8s is a container orchestrator
- Base unit is a “pod”
  - N containers
  - shared network namespace
- Some K8s “pieces”
  - Kube-apiserver
  - Scheduler
  - Controllers
  - Kubelet
  - Kubectl



How do I K8s?

- That sounds great, I want it!
- K8s is a platform, you have to decide what to build



Disclaimer

- Lots of trade-offs when building/designing k8s deployment
- No way to cover all items in detail in our time
- I’m going to cover a subset of these points, and cover what we did

K8s: Question 1 -- network

- What?
  - CNI plugins
- (High-level) Options?
  - Overlay: not route-able (usually) from outside of cluster -- depend on service endpoints
  - Non-overlay: pod IPs are routable in the network
- Which?
  - Non-overlay network
    - Avoid “access” issues with service-only ingress
    - Enables “all” services to move into k8s
    - We’re using https://github.com/aws/amazon-vpc-cni-k8s


YO DAWG, I HEARD YOU LIKE NETWORKS
SO WE PUT A NETWORK IN YOUR NETWORK SO YOU CAN NETWORK WHILE YOU NETWORK
made on imgur

K8s: Question 2 -- cluster layout

- What?
  - How many clusters, where to put them, planned failure domains
- (High-level) Options?
  - Global: Single cluster; enables some controllers
  - Per region: Some separation for failures
  - Per AZ: maximum separation for failures,
- Which?
  - Per AZ: fits with our reliability design and also avoids concerns of cluster-scale issues

K8s: Question 3 -- service discovery

K8s: Question 3 -- service discovery

- What?
  - How do we discover services (1) in-cluster (2) out-of-cluster (3) in-cluster
  - How do external services discover us?
- (High-level) Options?
  - K8s Services: accessible for all 3; requires all services to use this model
  - K8s SD: works in-cluster, can't register external SD into this
  - Consul: completely external SD mechanism, works for k8s and non-k8s
- Which?
  - Consul: We use consul for our other SD, works for all 3 modes, and less to support!

What is consul?

Consul is a distributed, highly available, and data center aware solution to connect and configure applications across dynamic, distributed infrastructure

- https://github.com/hashicorp/consul

Great, lets integrate k8s with consul!

K8s+consul v1: Sidecar consul-agent

- Why
  - Closest match to what we were doing outside of k8s
- What
  - Sidecar of consul-agent added to each pod

K8s+consul v1: Sidecar consul-agent

```
- command:
  - "/bin/consul"
  - agent
  image: consul:latest
  name: consul
  volumeMounts:
  - mountPath: "/etc/consul/secrets"
    name: consul-key
      - mountPath: "/etc/consul.d/"
    name: consul-config
```

Sidecar consul-agent problems

- Configuration
  - Consul secret in each namespace
  - Services/Tags need to be defined in a volume mounted to the sidecar
  - Even when templating manifests (e.g. jsonnet) this is a lot
- Consul vs k8s Checks
  - K8s itself has concepts of liveliness and readiness, keeping these in-sync with consul is operationally painful (different capabilities, options, and config)

Sidecar consul-agent problems

- Complexity
  - Enormous amounts of “nodes” in consul
    - 1 for the “node” + 1 per pod on the box
    - Consul nodes scale with non-host-network pod count; N+1
      - 10 pods per box means 11 consul nodes!
- Failure Modes
  - Thundering herd issues in consul failure

Sidecar consul-agent problems

- Noisy alerts
  - We use prometheus to monitor systems, prometheus uses consul's service discovery
  - Consul's deregistration defaults to 72h
  - The node still shows up in consul's service discovery until after the deregistration timeout

K8s+consul: Our requirements

K8s+consul: Our requirements

- Configuration through k8s annotations
- Readiness sync
- Highly Available with no Single Point of Failure (SPOF)

K8s+consul v2: consul-k8s

- Why
  - “First-class” option from hashicorp
- What
  - Configuration through k8s annotations
  - Syncs “readiness” of pod as health of consul entry

K8s+consul v2: consul-k8s

```
kind: Service
apiVersion: v1
metadata:
  name: my-service
  annotations:
    consul.hashicorp.com/service-name: my-consul-service
```

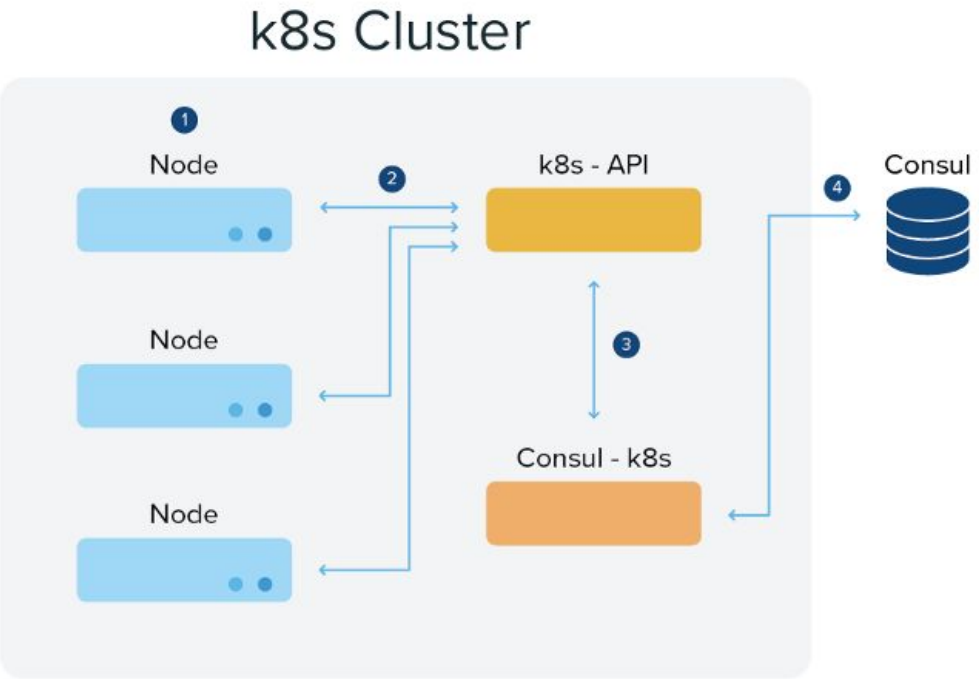
K8s+consul v2: consul-k8s

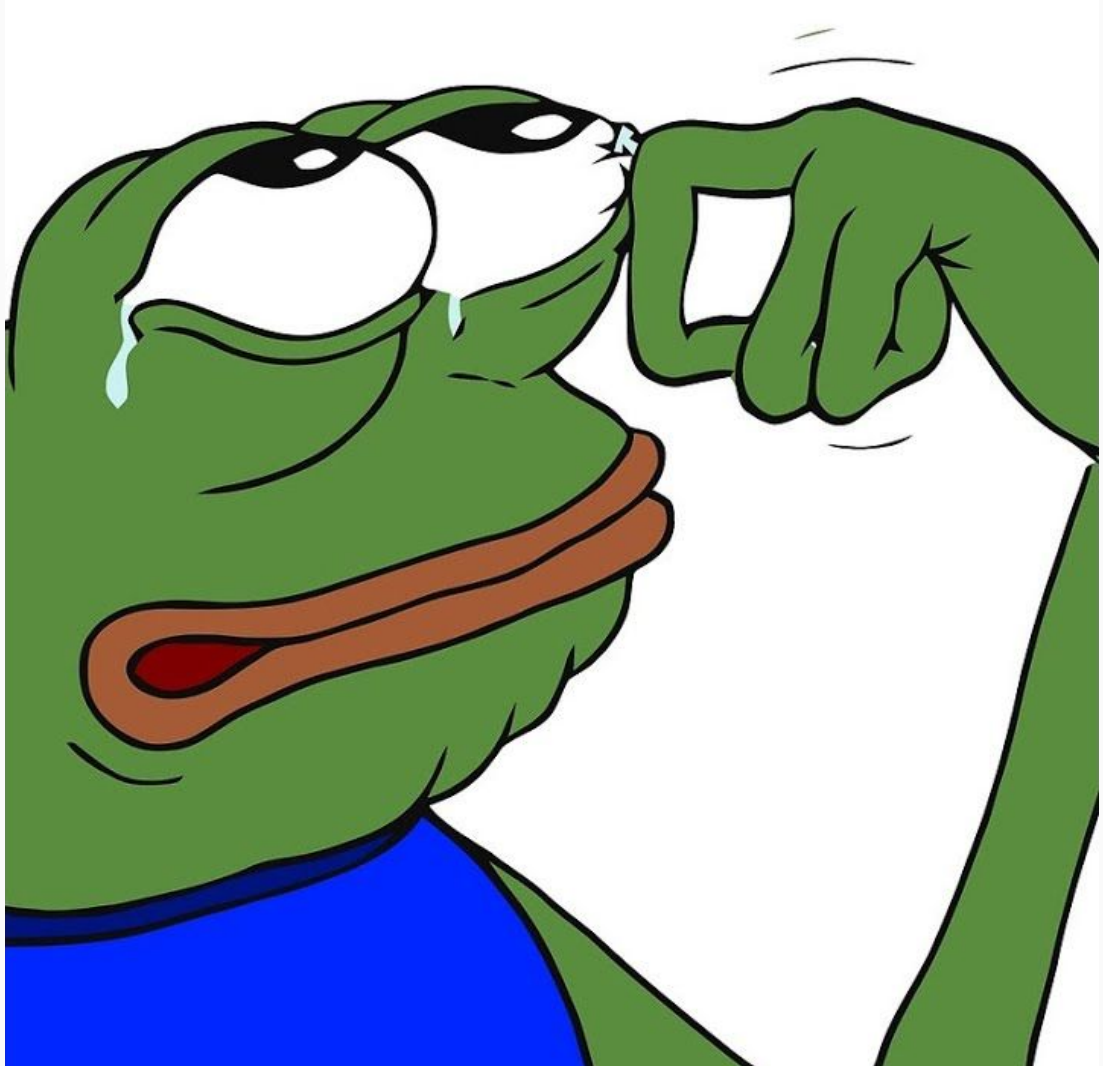
- Problems
  - Multi cluster support: https://github.com/hashicorp/consul-k8s/issues/42 (fixed)
  - Failure modes
    - No liveliness/readiness checks of the sync process (fixed): https://github.com/hashicorp/consul-k8s/issues/57
    - No mechanism to mitigate outage impact of consul-k8s: https://github.com/hashicorp/consul-k8s/issues/58
  - Not tied into readiness/deployment of pods/deployments
    - A requirement we didn't know we had!


NOT AWESOME
memeshappen.com

Consul-k8s sync process

1. Kubelet starts container on Node
2. Kubelet updates k8s API
3. Consul-k8s notices change in k8s-api
4. Consul-k8s pushes change to consul





K8s+consul Our requirements v2

- Configuration through k8s annotations
- Readiness sync
- HA with no SPOF
- **Ability to stop deploys from completing if not able to sync to consul**

K8s+consul v3

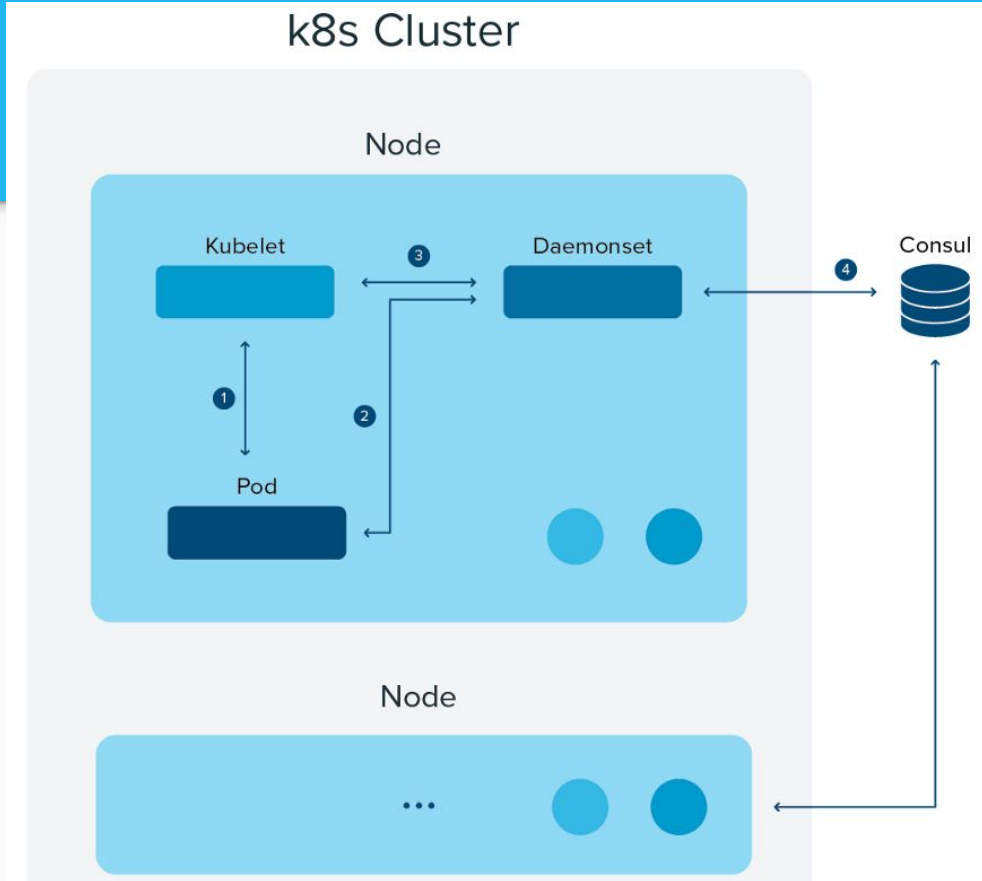
- Implementation Process
  - Poc -> testing -> failure testing
  - Local -> stage -> prod

K8s+consul v3: katalog-sync

- Design
  - Node-local sync daemonset
    - Sync services to consuls' Agent Services API
      - Agent-local services, health tied to consul-agent
      - All local syncing; No need for cluster-wide locking/coordination!
  - (optional) sidecar within pod to control deployment rollouts
  - Configuration through annotations

katalog-sync

1. Kubelet starts container on Node
2. (optional) katalog-sync-sidecar calls to katalog-sync-daemonset waiting until registration with consul is complete
3. Daemonset syncs changes from kubelet through the local kubelet API
4. Daemonset syncs changes to consul



K8s+consul v3: katalog-sync

```
apiVersion: v1
kind: Pod
metadata:
  annotations:
    katalog-sync.wish.com/service-names: my-service
    katalog-sync.wish.com/sidecar: katalog-sync-sidecar
```

Failure testing results

- Found during failure testing on pilot services in stage
  - Not all pods marked “ready” by the sidecars were in consul
- Saw errors in the consul-agent such as:

```
* Failed to join <IP>: Member ‘<US>’ has conflicting node ID
‘be688838-ca86-86e5-c906-89bf2ab585ce’ with member ‘<OTHER_MEMBER>’
```

Failure testing results

- Cause
  - Issue caused by an upgrade of consul-agent (fixed upstream now)
- Impact
  - Shows us that the local agent services API doesn’t consider syncing to the cluster
- Fix
  - Added a check for sidecar to ensure service is synced to the catalog API


EVERY THING IS
AWESOME
imgflip.com

Katalog-sync End-State

- Configuration
- Complexity
- Failure modes
- Noisy Alerts
- Consul checks vs k8s checks

Takeaways

- Kubernetes
  - Flexibility requires thought on how you'll deploy it
  - You don't need overlay networks to use k8s
  - Provides a great platform to integrate on top of
- Plan for and test failures
  - Can lead to finding unknown requirements
  - Saves you from a lot of pain in production

Questions?

Source: https://github.com/wish/katalog-sync/

Interested in this sort of thing? We're hiring! https://www.wish.com/careers