Building a Kubernetes on Bare-Metal Cluster to Serve Wikipedia

Alexandros Kosiaris
Giuseppe Lavagetto

WIKIMEDIA
FOUNDATION

Introduction

- The Wikimedia Foundation is the organization running the infrastructure supporting Wikipedia and other projects

- Monthly stats:
  - 17 Billion page views
  - 43 Million edits
  - 323 Thousands new registered users

Introduction (2)

- 2 Primary DCs (Ashburn, Dallas)
- 3 Caching DCs (San Francisco, Amsterdam, Singapore)
- ~1200 hardware machines
- ~100 VMs

- Size of engineering: ~160 people
- SRE team: ~ 20 people (4 dedicated to the application layer)
- We only write and use FLOSS

WIKIMEDIA
FOUNDATION

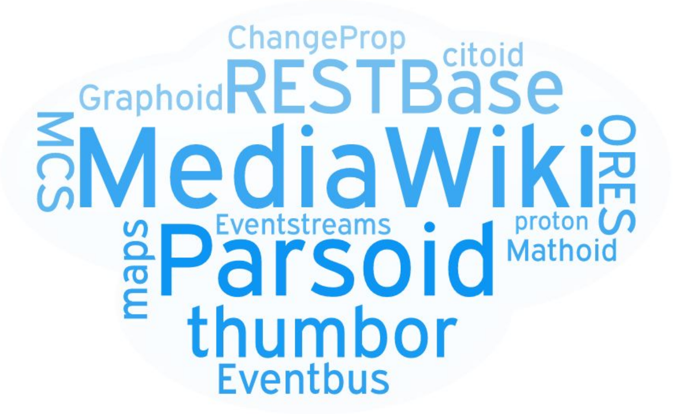
Reasons for introducing kubernetes

2014

MEDIAWIKI

Parsoid

2018



Also

- Elasticity
- Single-node failure management
- Containers
- Power to deployers!

But

- More moving parts
- New paradigm(™)
- Containers

WIKIMEDIA
FOUNDATION

Why on bare metal?

No public cloud

- User’s privacy guarantee
- Already maintaining our own infrastructure and CDN
- Costs

No private cloud

- We actually run Kubernetes on OpenStack, for a different project
- Reducing moving parts
- No practical advantages for production services

Cluster setup (1)

- We build our own Debian packages
  - Kubernetes 1.7 (on the way to 1.8 upgrade)
  - Calico 2.2.0
  - Etcd 2.2.1 (scheduled upgrade to 3.x)
- Configure all cluster components via Puppet
  - TLS included
  - But not the kubernetes resources!
- API servers are set up highly available (2 per cluster)
  - Kube-scheduler and kube-controller-manager run on the same hosts as apiserver

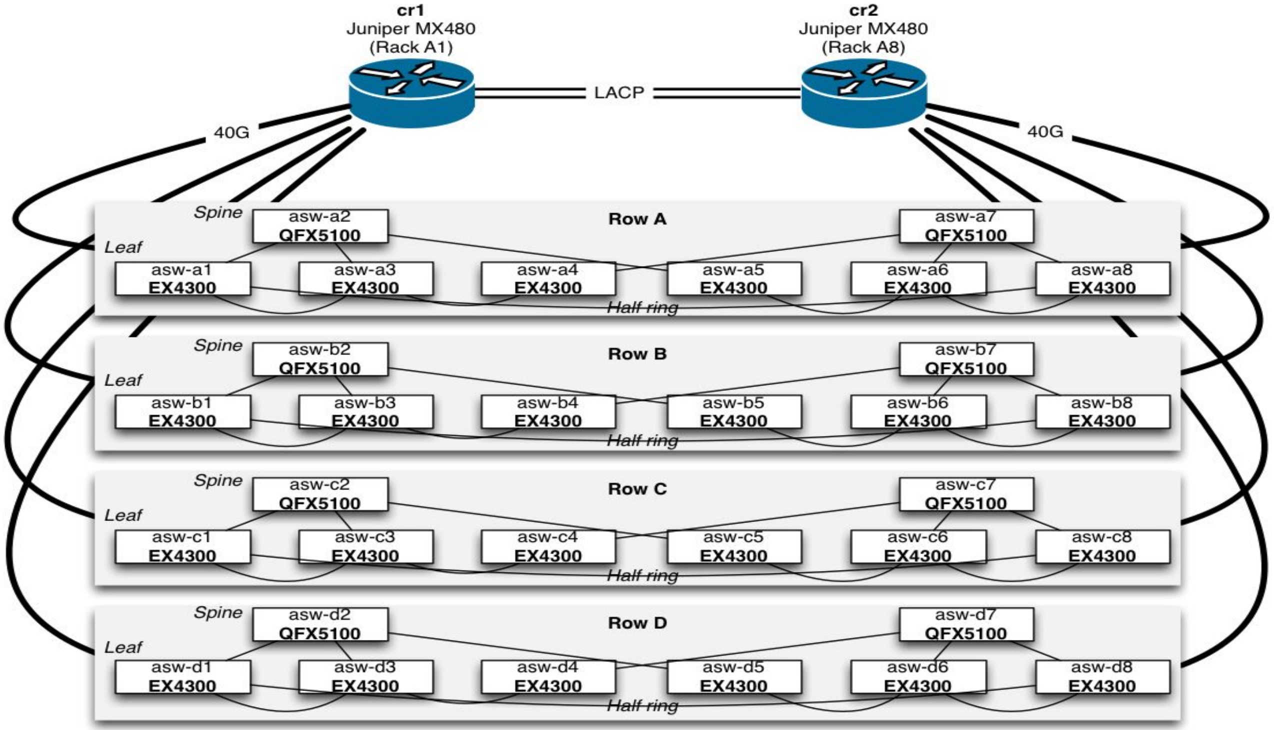
Cluster setup (2)

- 2 production clusters (1 per primary DC), 1 staging
- Separated Etcd clusters (3 servers, DC-local, on Ganeti VMs)
- Kube-proxy in iptables mode
- Docker as the runtime, but interested in rkt
- We host our own registry (and it’s the only allowed registry)
    - No image pulling from Dockerhub!
    - The backend of image storing is Openstack Swift
    - `docker pull docker-registry.wikimedia.org/wikimedia-stretch:latest`

Cluster setup (3)

- RBAC enabled since day #1 (we delayed rolling out a bit for that)
- 1 namespace per application/service
- Token-auth-file authentication
- Standard admission controllers for our version enabled
- Firewalling enabled on all nodes via ferm.
    - This works better than we feared!

Networking diagram



Networking

- Machines distributed per rack row for redundancy
- For backwards compatibility reasons we wanted to avoid overlay networking for pods
  - And calico fitted very nicely our networking model
- Calico BGP speakers on every node and the 2 routers
- No BGP Full mesh (cause the next hop is the router)
  - But will possibly have row specific full mesh
- RFC1918 10.x/8 address for the pods, but fully routable in our network.
- RFC1918 10.x/8 address for the Service IPs too, but:
  - Those are effectively just reservations
  - Are there to avoid surprises
- Pods have IPv6 address as well. Thank you Calico!
  - net.ipv6.conf.all.forwarding=1
  - net.ipv6.conf.eth0.accept_ra=2

WIKIMEDIA
FOUNDATION

Network Policies

- Kubernetes 1.7 does not support egress (but 1.8 does)
    - But Calico does
- Also does not allow changing a NetworkPolicy resource
- Alternative: We ‘ve patched calico-k8s-policy controller 0.6.0 (the python one)
    - Added reading a config file containing an enforced standard egress policy
    - Populate the file using a ConfigMap
    - Patch is minimal: 14 LoC in total
    - But also already deprecated. Next version is in Go

WIKIMEDIA
FOUNDATION

Ingress

- What about Ingress?
  - Evaluated it and decided to hold on it for now. We don’t even need the niceties yet.

- Use in-house python daemon running on load balancers (PyBal)
  - We do NodePort with externalIPs
  - And PyBal manages LVS DR entries on the load balancers

- A lot of expertise in house regarding PyBal, reusing it sounded the best approach
- Open Source: https://github.com/wikimedia/PyBal

Metrics collection

- Infrastructure (Prometheus)
  - The discovery mechanisms rule
  - Polling all API servers
  - Polling all kubelets
  - Polling kubelet cAdvisors as well (hello kubernetes 1.7.3!!!)

- Applications (Prometheus, yes that too!)
  - Applications historically used statsd
  - prometheus_statsd_exporter in a sidecar container, applications talk to localhost :-)
  - Prometheus discovers and polls all pods

- https://grafana.wikimedia.org/dashboard/db/kubernetes

Alerting

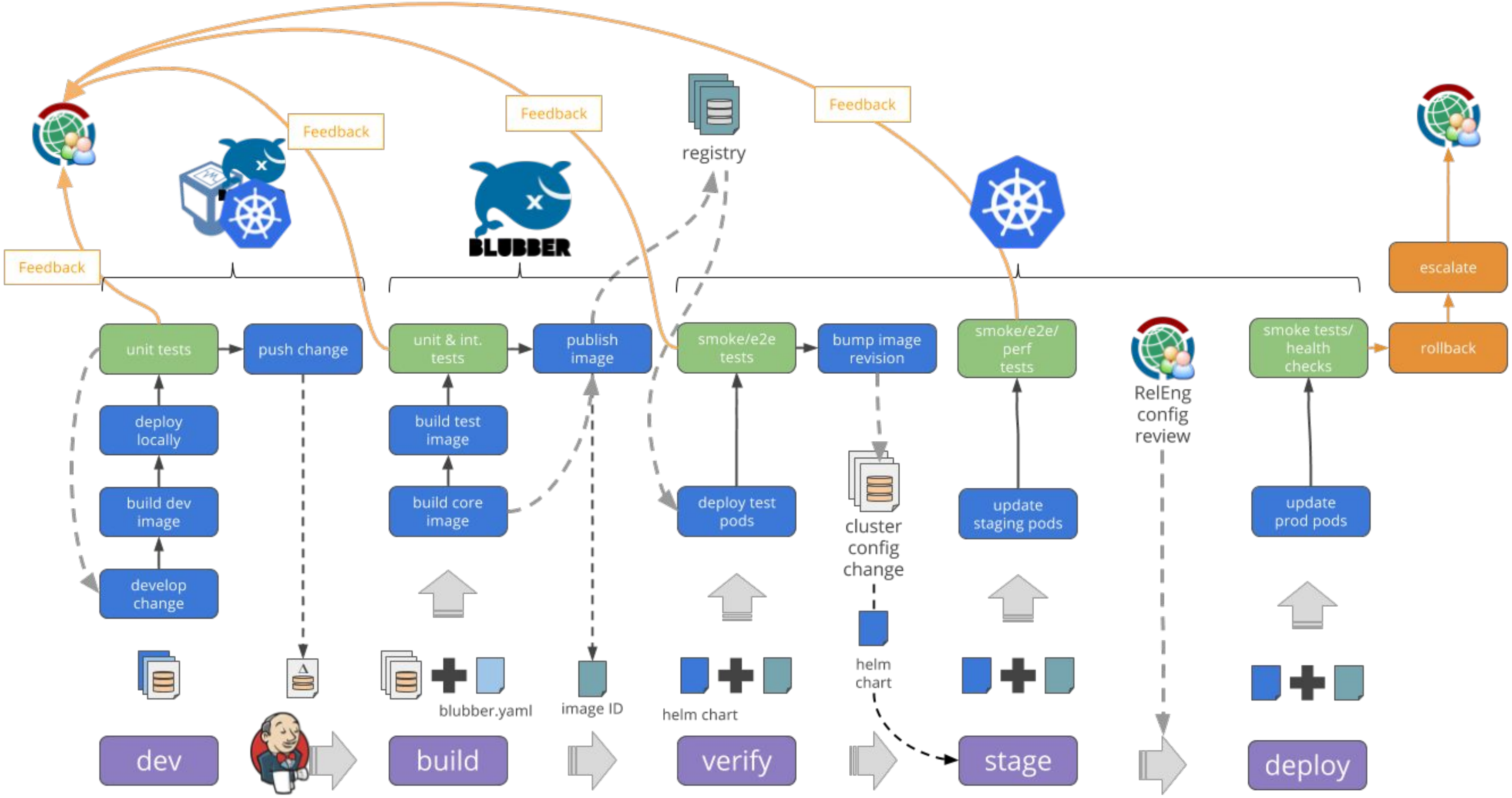
- Prometheus again! Albeit only partly
- Still on icinga 1.11.6
- Started by using check_prometheus_metric.sh
  - And it did not support floats
  - Rewrote the whole thing in python
- Add in Puppet and the checks can be a bit intimidating to look at

  -
    ```
    query => "scalar(\
                sum(rate(kubelet_runtime_operations_latency_microseconds_sum{\
                job=\"k8s-node\", instance=\"${::fqdn}\"}[5m]))/ \
                sum(rate(kubelet_runtime_operations_latency_microseconds_count{\
                job=\"k8s-node\", instance=\"${::fqdn}\"}[5m])))",
    ```
- Swagger spec based monitoring to instrument checks
  - https://github.com/wikimedia/operations-software-service-checker

WIKIMEDIA
FOUNDATION

Streamlined Service Delivery

We re-thought the whole software lifecycle for things that will run on Kubernetes, from development to deployment

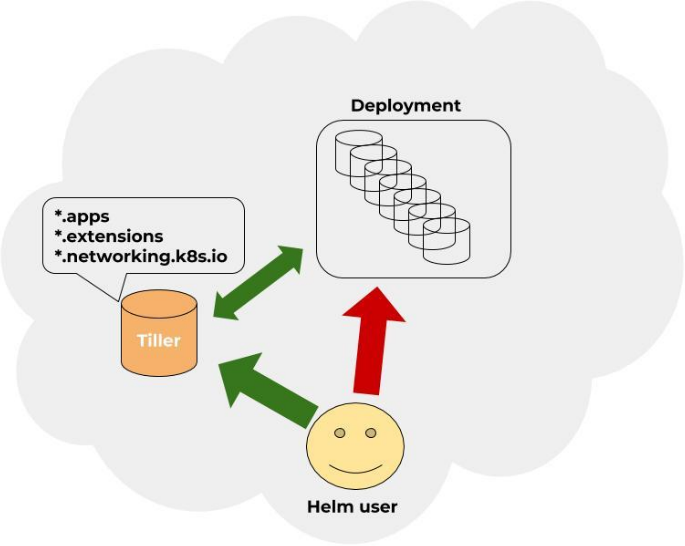


WIKIMEDIA
FOUNDATION

Deployment

Helm setup:

- Tiller resides on the namespace of the application and has specific RBAC rights
- A “deploy” user is only granted rights to talk to Tiller
- Unix users with rights to deploy to a namespace get access to the corresponding credentials
- A simple wrapper around helm ensures the correct credentials are used



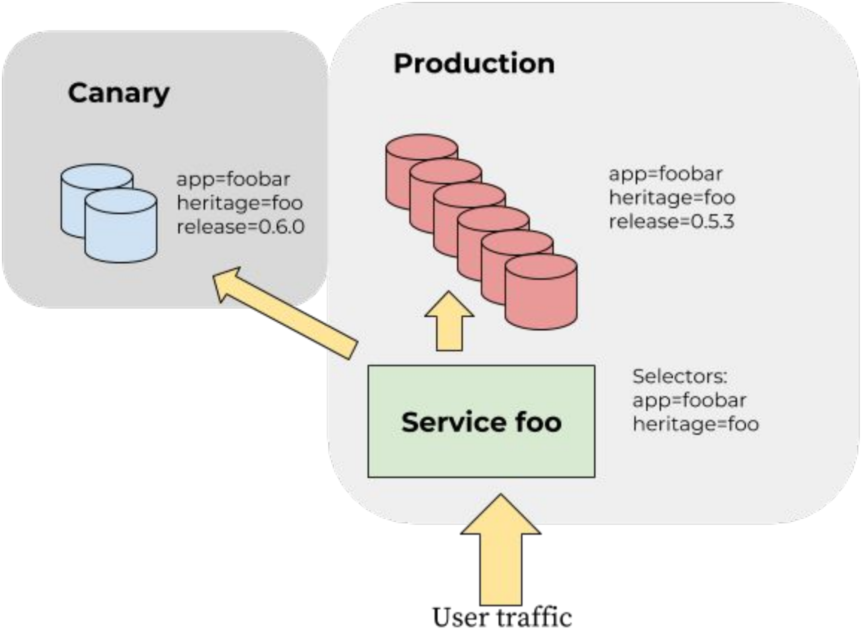
WIKIMEDIA FOUNDATION

Deployment

Also:

- Deploy to both datacenters
- We (will) support canary deployments
  - 2 helm releases, “canary” and “production”
  - Only production declares a Service, which selects all pods from both releases
  - New release of the software goes to “canary”, once all tests and metrics are green, it gets deployed to “production”
- Goal is to rewrite the wrapper as helm plugins

Our own (very new!) helm repo: https://releases.wikimedia.org/charts/



THANK YOU

WIKIMEDIA
FOUNDATION

SRE team is hiring!

https://jobs.wikimedia.org

https://github.com/wikimedia/PyBal
https://grafana.wikimedia.org/dashboard/db/kubernetes
https://github.com/wikimedia/operations-software-service-checker
https://releases.wikimedia.org/charts/
https://docker-registry.wikimedia.org

WIKIMEDIA
FOUNDATION

Questions?

WIKIMEDIA
FOUNDATION