A Customer-Focused SLA for a Kubernetes-Based PaaS

Shrenik Dedhia, Sr. Staff Engineer/Tech Lead Manager

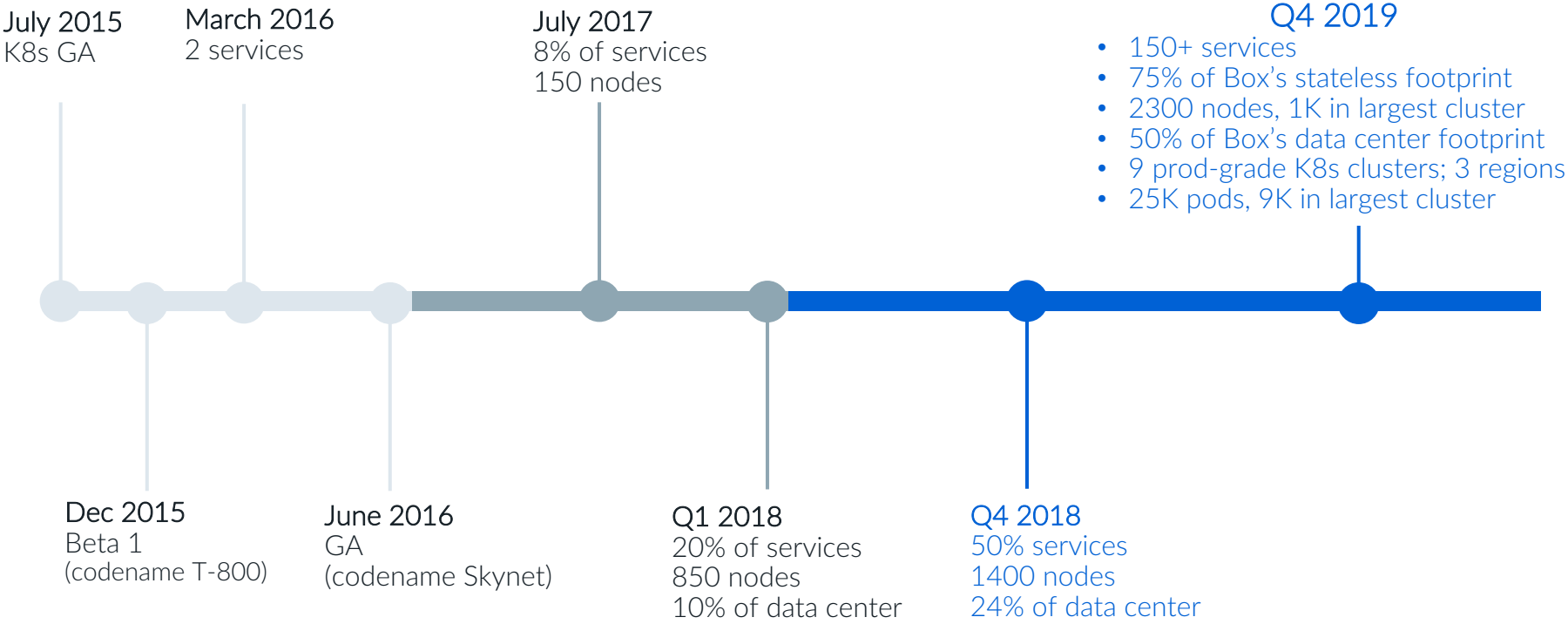
Kubecon 2019

Agenda

1. Kubernetes @ Box
2. The Problem
3. Exploration
4. Principles
5. Path Taken

Kubernetes @ Box

Under The Hood



Kubernetes @ Box

Over The Hood

Platform As A Service @ Box

- Mission: Run Box apps securely, reliably, efficiently, in any region (*of the world*)
- Built on K8s w/ few abstractions
- Declarative config monorepo in Git
- Jsonnet templating
- *Kube-applier* to apply configuration
  - https://github.com/box/kube-applier (Box donated to open source)

Kubernetes @ Box

Over The Hood

Platform As A Service @ Box

- Cross-cutting control plane integrations:
  - PKI – *Box Internal CA*
  - Secrets – *Hashicorp Vault*
  - Network policies/IPAM – *Calico*
  - Service discovery – *SmartStack*
  - Image management – *Artifactory*
  - Pipelines – *Jenkins*

- Multi-tenancy using K8s namespaces and RBAC policies

The Problem

Conundrum

How do we measure platform health?

- What is our service level agreement (SLA)?
- Which key performance indicators (KPI's)?
  - *e.g. LATEST – Latency, Availability, Throughput, Error Rate, Saturation, Traffic*
- What service level objective (SLO)?
- How do we measure the service level indicator (SLI)?

The Problem

Conundrum

How do we measure platform health?

- Breadth: control plane / data plane / other?
- Depth: cluster / availability zone / region?
- What does the industry think? "Liberal" examples:
  - *"Unavailable" and "Unavailability" mean that **all** connection requests to an endpoint for the applicable EKS Cluster fail during a 5-minute interval* (source)
  - *Loss of external connectivity and/or Kubernetes API access to **all** running clusters with the inability to launch replacement clusters in any zone* (source)

Exploration

Start Simple

Initial solutions explored

- Minutes of Box.com degradations caused by PaaS
  - Simple but very coarse
  - Not all PaaS degradations count
- % of 5xx's from K8s API server
  - i.e. control plane availability and uptime
  - Doesn't account for the data plane
- Synthetic test app exercising CI/CD
  - i.e. data plane availability (schedule/create/start/evict pods)
  - Not "real"; not representative of customers

Platform Health

What do customers care about?

How do we leverage Kubernetes?

Principles

Customer-focused

What do customers care about?

- Care about
  - Ease of use of the platform
  - Availability/uptime of their own services
  - Consistently serve 100% of peak traffic

- Don't care about (for most part)
  - Control plane availability
  - Kubernetes API nitty gritty
  - Protections against h/w and s/w faults, bin-packing inefficiency, maintenance, etc.

Principles

Leverage Kubernetes

What pod availability protections does Kubernetes offer?

- Liveness/readiness probes
  - Accounts for downstream dependency degradations
- Rolling updates
  - Blocks propagation of bad changes
- PodDisruptionBudget (PDB)
  - Protects service from the cluster administrator disruptions
- QoS for pods (guaranteed, burstable, best effort)
  - Enables scheduler make decisions for scheduling/evicting pods
  - Prefer “guaranteed” to scale horizontally (more pods) vs vertically (more cpu/mem per pod)

Note: There may be others which we haven’t explored.

Path Taken

Critical Replica Availability “CRA”

How does it work?

- Principle
  - Build platform to provide HA for services
  - If platform doesn’t meet HA needs ➔ must impact platform’s KPI
  - i.e. data-driven accountability and effective prioritization
- Concept
  - Customer defines critical replicas threshold, i.e. minimum healthy replicas for availability
  - Auto-configure K8s protections to serve critical replicas
  - Calculate replica overhead for protections

Path Taken

Critical Replica Availability “CRA”

Example (values are suggestive only)

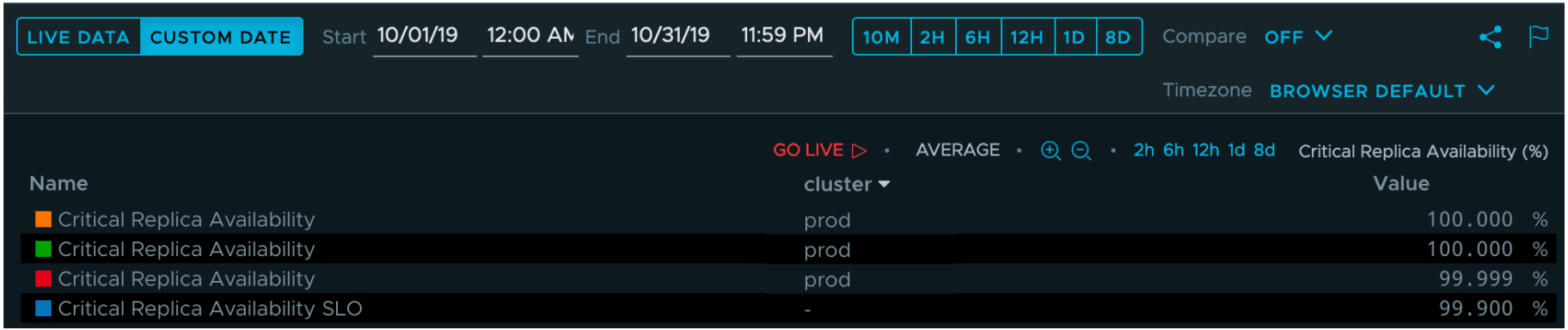
- Customer
  - Define critical replica threshold (CRT) = 100 replicas
  - Configure liveness/readiness probes
  - Use “guaranteed” QoS
- Tooling auto-adds
  - 5% h/w fault buffer = ceil(100 * 105%) = 105 pods
  - 5% PDB buffer = 111 pods
  - 5% rolling update (surge) = 117 pods ← replicas given to the service
  - 5% maintenance buffer = 123 pods ← worth of physical capacity allocated

Path Taken

Critical Replica Availability “CRA”

Measuring the SLI

- Critical Replica Threshold = minimum # replicas required for HA
- Available replicas = min(total available pods, CRT)
- Replica availability = (available replicas / CRT) * 100
- Critical Replica Availability = mean(replica availability of each service)



Path Taken

Critical Replica Availability “CRA”

Next Steps

- Gaps
  - Massive system failures will impact SLI regardless of method
  - Treats all services equally; no tiering
  - Subject to service owner induced errors, e.g. lots of crash-looping pods, or incorrectly configured liveness/readiness probes, etc.
- Future Work (For GA)
  - Leverage newer Kubernetes features, e.g. Pod Priority and Preemption
  - Additional tooling for improving HA for services, e.g. canaries, chaos, etc.
  - Stronger service owner accountability and policies, e.g. alerts targeting service owners, exclusion from SLI for misbehaving services, etc.

Questions?

sdedhia@box.com