Optimizing Mesos Utilization at OpenTable

JAY CHIN
INFRASTRUCTURE ENGINEERING
1.4 Billion Online Reservations
2.3 Million Diners per Month
58 Million verified reviews
Around 2013

Opentable Codebase

- Search
- Reviews
- Emails
- Reservations
- Photo Service
- Restaurant profiles
- Availability Service
- Menu API
- White Label
- External API
- Person API
- Feedback API
- Etc
- Etc
- Etc
Virtualization
Let’s Scale!

VM
Search

VM
Reviews

VM
Photos

VM
Restaurant profiles

VM
Availability Service

VM
External API

VM
Person API

VM
Menu API

VM
White Label

VM
Feedback API

VM
Emails

VM
Datacentre

VM
Restaurant profiles

VM
External API

VM
Person API

VM
Menu API

VM
White Label

VM
Feedback API

VM
Emails

VM
Datacentre

VM
Restaurant profiles

VM
External API

VM
Person API

VM
Menu API

VM
White Label

VM
Feedback API

VM
Emails

VM
Datacentre

VM
Restaurant profiles

VM
External API

VM
Person API

VM
Menu API

VM
White Label

VM
Feedback API

VM
Emails

VM
Datacentre

VM
Restaurant profiles

VM
External API

VM
Person API

VM
Menu API

VM
White Label

VM
Feedback API

VM
Emails

VM
Datacentre

VM
Restaurant profiles

VM
External API

VM
Person API

VM
Menu API

VM
White Label

VM
Feedback API

VM
Emails
Around 2014
Explore Mesos
Local Build

- Code
- Local Docker Testing
- Push to Docker Repo

Provision

- Deploy Service to Mesos Cluster
- Deploy service to other Mesos Cluster

Metrics

- Code integration with Statsd/Graphite
- Write Puppet Code
- Infrastructure Team pushes Puppet code
- Build Grafana Dashboards

Monitoring

- Identify Metrics or emit metrics
- Runbooks and escalation policies
- Write Puppet Code
- Infrastructure Team pushes Puppet code
Metrics Pipeline

- Mesos Task
- Singularity API
- Mesos API
- Carbon Format Publisher
- Kafka
- Carbon Format Consumer
- Carbon-c relay
- Graphite Cluster
- Grafana

https://github.com/opentable/mesos_stats
https://github.com/weaveworks/grafanalyzer
Auto-generated Grafana Dashboard

Every Service running in Mesos will have an auto-generated dashboard

Help text explaining the graphs and what they mean

Shows cluster-wide Usage and Instance Usage
Right-sizing Resource Usage = $$$ Saved

Mesos Cluster

Singularity Task

resources: {}
  cpus: 0.1
  memoryMb: 256
  numPorts: 1
  diskMb: 0

Shows that memory is over-provisioned for this service
**Local Build**
- Code
- Local Docker Testing
- Push to Docker Repo

**Provision**
- Deploy Service to Mesos Cluster
- Deploy service to other Mesos Cluster

**Metrics**
- Only application specific metrics
- Create application specific dashboards

**Optional**

**Monitoring**
- Identify Metrics or emit metrics
- Runbooks and escalation policies
- Write Puppet Code
- Infrastructure Team pushes Puppet code
Sous
https://github.com/opentable/sous

- Manifest Change
- Sous Deploy
- Global Deployment Manifest
- Sous Service
- Container Repository
- Sous Build
- Code
- Mesos Cluster QA
- Mesos Cluster Prod (London)
- Mesos Cluster Prod (US-West2)
Source: github.com/opentable/xxxx

Owners:
- marcopolo@opentable.com
- james_bond@opentable.com

Kind: http-service

Deployments:

Prod-London:

Resources:
- cpus: "0.15"
- memory: "256"
- ports: "1"

Env:
- OT_AUTOCHECK_RUNBOOK: https://wiki.otcorp.opentable.com/myfabulousservice
- OT_AUTOCHECK_SENSU_TEAM: the_A_team
- SERVICE_TYPE: super-server

NumInstances: 3

Volumes: []

Startup:
  Timeout: 188
  CheckReadyURIPath: /servers
  CheckReadyURITimeout: 1

Version: 0.5.45

QA-uswest2:

Resources:
- cpus: "0.15"
- memory: "256"
- ports: "1"

Env:
- OT_AUTOCHECK_RUNBOOK: https://wiki.otcorp.opentable.com/myfabulousservice
- OT_AUTOCHECK_SENSUTEAM: the_A_team
- SERVICE_TYPE: super-server

NumInstances: 6

Volumes: []

Startup:
  Timeout: 188
  CheckReadyURIPath: /servers
  CheckReadyURITimeout: 1

Version: 0.5.46
Local Build

- Code
- Local Docker Testing
- Sous Deploy

Provision

- Updated Global Deployment Manifest

Metrics

- Only application specific metrics
- Create application specific dashboards

Optional

Monitoring

- Identify Metrics and Thresholds
- Runbooks and escalation policies
- Updated Global Deployment Manifest
Logging

Restaurant_id == RID == ResID == Res_ID

Global RequestID
https://github.com/opentable/request-timeline
Timeline Demo
Key Takeaways

- Map out developer workflow and constantly look for opportunities to standardise, automate and enhance.
- Make metrics and monitoring part and parcel of the Mesos service.
- Engineers don’t always make the best choice when deciding resource usage - help them make an informed choice.
- Have a common deployment pipeline across the organisation that facilitates production readiness*
- Having a global data model for logging allows us to make more sense of logging data across the various Mesos tasks.