Helping your Dev teams succeed at Ops, post-Kubernetes

MOO

Intro

Kubernetes goes live at MOO

- Breaking up the old monoliths
- Shift towards micro-services to achieve this
- Remove the Platform Ops team from the critical path of setting up and deploying new services

Additional responsibilities for our crews

- Redundancy (replicas)
- Monitoring
- Alerting
- Resource allocation
- ...

Helping our teams to help themselves...

Techniques we used, the good bits and the bad.

Template app

Generated with a small amount of user input, gave our crews a starting point and some sensible defaults

- Pipeline that went all the way to prod cluster
- A set of default annotations
- Make targets to deploy to minikube locally

Template app

But it wasn’t that easy

- Made certain assumptions
- Difficult get updates out to the crews
- The opsy stuff was a bit heavyweight
- 2 distinct opinions on direction of the project

Knowledge share

Spreading the k8s love

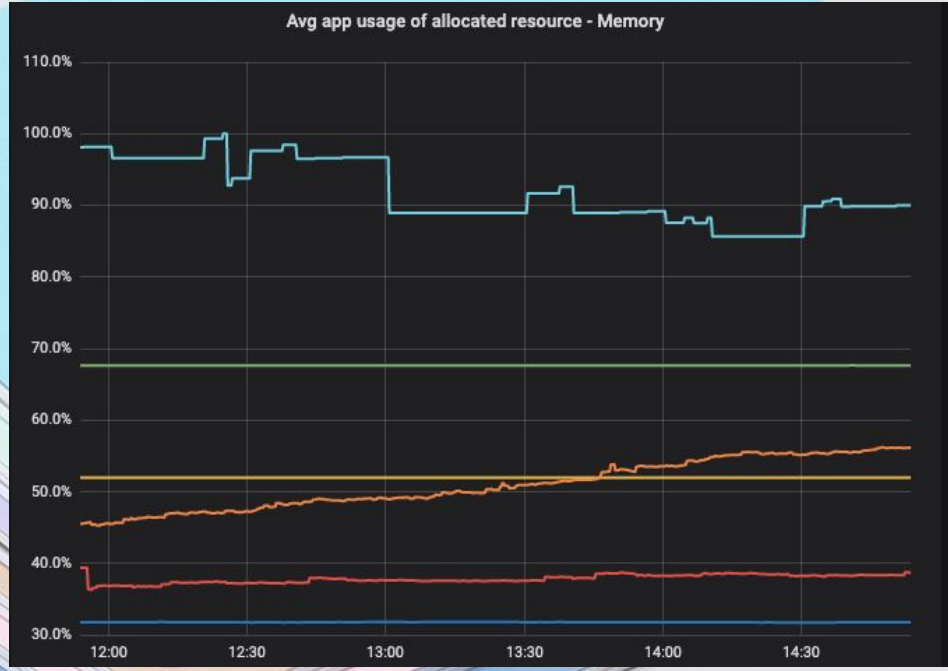
Took a number of approaches

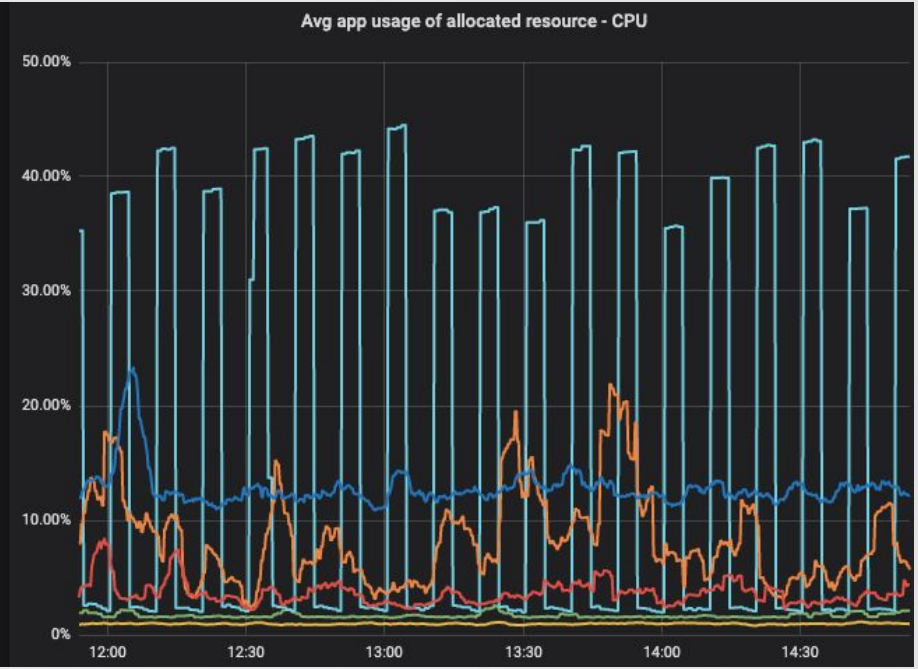
- Ran “k8s 101” workshop sessions for all teams
- Created a **#kubernetes** slack channel
- Embedded Ops engineers with crews

Quotas

Namespace based resource limits

- Generous
- Reaching limits provoked conversation
- Ensured apps were tuned accordingly
- Not great for HPA!


Avg app usage of allocated resource - Memory
110.0%
100.0%
90.0%
80.0%
70.0%
60.0%
50.0%
40.0%
30.0%
12:00
12:30
13:00
13:30
14:00
14:30


Avg app usage of allocated resource - CPU
50.00%
40.00%
30.00%
20.00%
10.00%
0%
12:00
12:30
13:00
13:30
14:00
14:30

Self-serve prometheus alerts

We built it like code

- Tests
- Pipelined
- Per crew alert channels

alert_source_files/sandbox/auto_anomaly_detection.alerts

+11 −0

```
+ - name: sandbox
+   rules:
+   - alert: p95LatencyExceeded
+     expr: app:latency:p95 > (app:latency:avg_offset_p95 + (app:latency:stddev_offset_p95 * 3))
+     for: 15m
+     labels:
+       severity: sandbox
+     annotations:
+       summary: 'Endpoint p95 latency exc
+       description: 'The endpoint {{ $lab
https:/[redacted]/d/bAeUOc
env=Thanos-Prod&var-percentile=95&var-se
$labels.method }}&var-endpoint_name={{ $
```

tests/sandbox/sandbox.tests

+24 −0

```
+ rule_files:
+   - alerts_concat/alerts.yaml
+ evaluation_interval: 1m
+ tests:
+   - interval: 1m
+     input_series:
+       - series: 'app:latency:stddev_offset_p95{app="foo",endpoint_name="/test",method="GET"}'
+         values: '1 1+0.1x11 2.2-0.1x11 1+0x10'
+       - series: 'app:latency:avg_offset_p95{app="foo",endpoint_name="/test",method="GET"}'
+         values: '1+0x35'
+       - series: 'app:latency:p95{app="foo",endpoint_name="/test",method="GET"}'
+         values: '3+0x10 4+0.5x10 9+0x4 9-0.5x9'
+     alert_rule_test:
+       - alertname: p95LatencyExceeded
+         eval_time: 32m
+         exp_alerts:
+           - exp_labels:
+               severity: sandbox
+               app: foo
+               endpoint_name: /test
+               method: GET
+             exp_annotations:
+               summary: 'Endpoint p95 latency exceeding normal duration : foo'
+               description: 'The endpoint /test appears to be operating slowly. Check
https:/[redacted]/d/bAeUOciWk/k8s-service-kpis-template?orgId=1&from=now-1h&to=now&var-
env=Thanos-Prod&var-percentile=95&var-service=foo&var-stddev_multiplier=3&var-method=GET&var-
endpoint_name=/test'
```

Solving problems from an abstract perspective...

Data driven alerts

Provide the means to make informed choices about thresholds

- Standard metrics
- Recording rules
- Dashboards to inform
- Sandbox alerts channel to try things out.

HTTP request response time histograms

See: https://prometheus.io/docs/concepts/metric_types/#histogram

Metric name: **http_requests_seconds_bucket**
Labels: endpoint_name
method
response_code <optional>
role and/or job
le

Example:

```
http_requests_seconds_bucket{endpoint_name="/project",instance="foo_services.prod.eu-west-1-10.112.2.176
",app="foo",le="0.025",method="POST",response_code="2xx",role="foo_services"}
```

Remember when generating the endpoint_name to sanitise any values unique to the request e.g.
endpoint_name="/project/<project_id>/foo/<process>"

http_requests_seconds_buckets

HTTP request volume

See: https://prometheus.io/docs/concepts/metric_types/#counter

Metric name: **http_requests_seconds_count**

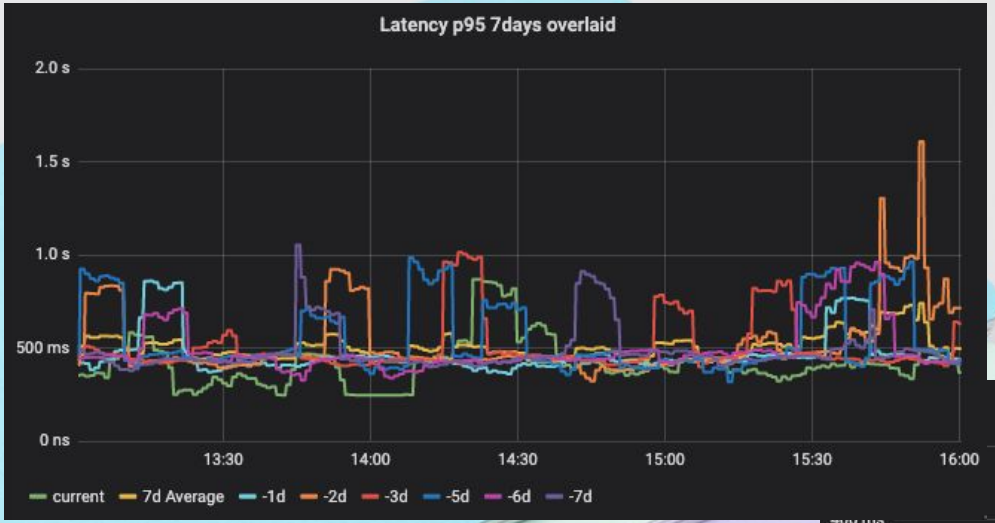
Minimum Labels required:
- endpoint_name
- method
- response_code : *either grouped (1xx, 2xx, 3xx…) or explicit (200, 503 etc.) but not both.*
- role and/or job : *so we have something to distinguish your service*

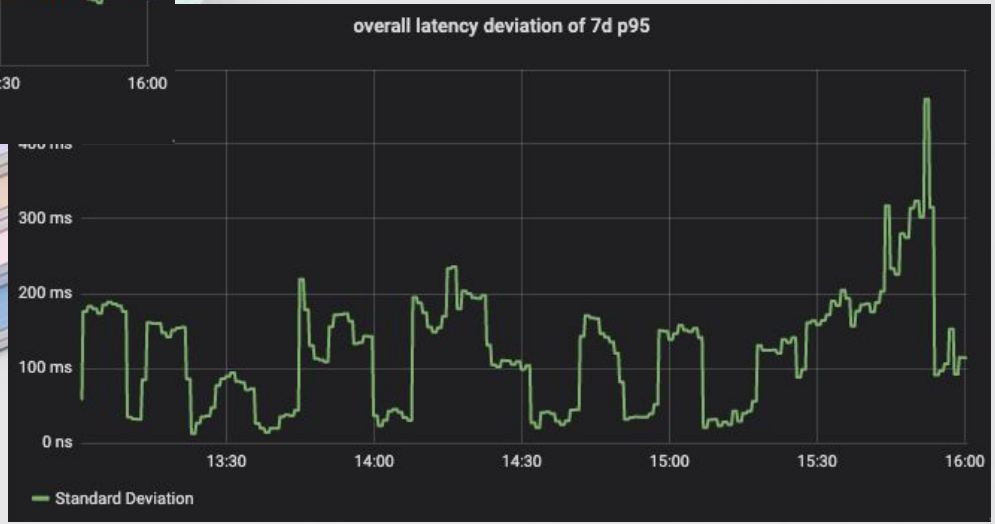
Example:

```
http_requests_seconds_count{endpoint_name="/project",instance="foo_services.prod.eu-west-1-10.112.65.42"
,app="foo",method="POST",response_code="2xx",role="foo_services"}
```

http_requests_seconds_count

```yaml
rules:
  - record: app:latency:rate10m
    expr: sum(rate(http_requests_seconds_bucket[10m])) without (instance, node, build)
  - record: app:latency:p95
    expr: histogram_quantile(0.95, app:latency:rate10m)
  - record: app:latency:offset_p95
    expr: app:latency:p95 offset 1d
    labels:
      offset: 1d
  - record: app:latency:offset_p95
    expr: app:latency:p95 offset 2d
    labels:
      offset: 2d
  - record: app:latency:offset_p95
    expr: app:latency:p95 offset 3d
    labels:
      offset: 3d
   <-------- SNIP! --------->
  - record: app:latency:offset_p95
    expr: app:latency:p95 offset 7d
    labels:
      offset: 7d
  - record: app:latency:avg_offset_p95
    expr: avg without (offset) (app:latency:offset_p95)
  - record: app:latency:stddev_offset_p95
    expr: stddev without (offset) (app:latency:offset_p95)
  - record: app:latency:normalised_offset_p95
    expr: (app:latency:offset_p95 - app:latency:avg_offset_p95) / app:latency:stddev_offset_p95
```


Latency p95 7days overlaid
2.0 s
1.5 s
1.0 s
500 ms
0 ns
13:30
14:00
14:30
15:00
15:30
16:00
current
7d Average
-1d
-2d
-3d
-5d
-6d
-7d


overall latency deviation of 7d p95
300 ms
200 ms
100 ms
0 ns
13:30
14:00
14:30
15:00
15:30
16:00
Standard Deviation

alert_source_files/sandbox/auto_anomaly_detection.alerts

+11 −0

```
+ - name: sandbox
+   rules:
+   - alert: p95LatencyExceeded
+     expr: app:latency:p95 > (app:latency:avg_offset_p95 + (app:latency:stddev_offset_p95 * 3))
+     for: 15m
+     labels:
+       severity: sandbox
+     annotations:
+       summary: 'Endpoint p95 latency exceeding normal duration : {{ $labels.app }}'
+       description: 'The endpoint {{ $labels.endpoint_name }} appears to be operating slowly. Check
https:/[redacted]/d/bAeUOciWk/k8s-service-kpis-template?orgId=1&from=now-1h&to=now&var-
env=Thanos-Prod&var-percentile=95&var-service={{ $labels.app }}&var-stddev_multiplier=3&var-method={{
$labels.method }}&var-endpoint_name={{ $labels.endpoint_name }}'
```

Ongoing engagement...

Evolving your k8s

Involve representative from the crews

- Observability
- Service Meshes
- Secrets management
- And so on....

In summary

Giving Ops to your crews:

- There's a lot to do and it's complicated but you can...
- Set sensible defaults and limits
- Mimic the development workflow where possible
- Enable data to inform decisions
- Stay engaged with your crews
- And involve them in evolving your platform

Thank you

MOO