Helping your Dev teams succeed at Ops, post-Kubernetes
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
Handing over
Setting the teams up for success

Additional responsibilities for our crews

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

MOC
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
But it wasn’t that easy

- Made certain assumptions
- Difficult to get updates out to the crews
- Theopsy 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!
Self-serve prometheus alerts

We built it like code

- Tests
- Pipelined
- Per crew alert channels
```yaml
# alert_source_files/sandbox/auto_anomaly_detection.alerts

- 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 exceeded normal duration
      description: The endpoint {{ $labels.method }} var-endpoint_name={{ $labels.method }}

# tests/sandbox/sandbox.tests

- rule_files:
  - alerting_concat/alerts.yaml
  - evaluation_interval: 1m
  - tests:
    - 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+0x80'
      - series: 'app:latency:avg_offset_p95(app="foo",endpoint_name="/test",method="GET")'
        values: '1+9x35'
      - series: 'app:latency:p95(app="foo",endpoint_name="/test",method="GET")'
        values: '3+9x10 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
          description: The endpoint /test appears to be operating slowly. Check
          values: 'http://d/baev0c1wk/k8s-service-kpis-template?orgId=16from-now-1h&to=now&var-env=Thanos-Prod&var-percentile=95&var-service=foo&var-stddev_multiplier=36&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](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](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
```

**MOC**
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:normalized_offset_p95
  expr: (app:latency:offset_p95 - app:latency:avg_offset_p95) / app:latency:stddev_offset_p95
- 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://d/bAeU0ciWk/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