frustrating deployment and update process
container orchestration solutions with a bazillion features
monitoring dynamically changing services
observability in a dynamically scheduled world
leveraging Prometheus and Alertmanager for cluster monitoring and alerting
about me

software engineer @DigitalOcean
former delivery, currently observability kubernetes, prometheus
the plan:

- the “olden” days vs. container orchestration
- **doxx** at DigitalOcean
- **prometheus** + **alertmanager** and **doxx**
- alerting in action: examples
- potential pitfalls
- next steps
the “olden” days:

1. service owners write an application
2. provision a server with chef or ansible
3. use a CI/CD pipeline, bash scripts, or other tools to deploy and update application on VM
the “olden” days:

4. use nagios + various plugins to monitor
5. use collectd + statsd + graphite/carbon
the “olden” days:

- longer to provision than write actual service
- hard to set up monitoring services
- blackbox monitoring **NOT** insightful
- whitebox monitoring services **NOT** easily **queryable**
Digital Ocean Command Center

a tool for deploying containerized, stateless applications
post-docc:

abstraction layer on top of kubernetes deployments and updates take minutes, not hours

easy-to-use CLI
post-docc:

1. service owners write an application
2. service owner dockerizes application
3. describe application in json manifest file
4. deploy!
post-docc:

view running applications

get application logs

easily **scale**, **update**, or **restart** applications
But what about **monitoring**?
Let’s use `prometheus + alertmanager`
Why use **prometheus** and **alertmanager**?
digitalocean.com

easy to deploy
flexible and extensible
labelling works well with kubernetes primitives
complementary **kubernetes** service discovery
low-level metrics leveraged via a strong query language
counters
gauges
histograms
summaries
alertmanager:

easily deployed alongside prometheus

dedupes alerts

high availability configuration

multiple receiver options
any downsides?

**push vs. pull** model

service owner must *instrument* application
putting it all together
1. **instrument** your application
   
   use prometheus golang client
   
   expose **metrics endpoint**
specify **metrics, ports, alerts** in your manifest file

Which **metrics endpoint** should be scraped?

Which container **port** needs to be exposed?

Specify **alerting rule**, **duration** interval, and **channel**.
use **docc CLI** to deploy your application

```bash
$ docc deploy manifest.json
```
docc

daemonset → pods

service

promconfig

alertmanager

alertconfig
**prometheus** talks to the **kubernetes api** and grabs the metrics endpoint and port information.
promconfig grabs alert information and rewrites Prometheus rules file
alertconfig grabs alert routes and rewrites alertmanager configuration file
Some stats
300+ production applications
1.5 million+ timeseries
100+ prometheus alerts
What should we monitor?
counters: cumulative, increasing metric

gauges: single metric that goes up or down

histograms: samples and buckets observations

summaries: also samples observations but can calculate things like quantiles
latency - *histogram + summaries*

traffic - *counters + rate()*

error - *counters + rate()*

saturations - *gauge*
R - request rate
E - error rate
D - duration
U - utilization
S - saturation
E - error rate
real-life examples

cluster CPU reservation
node memory utilization
loadbalancer connection error rate
service http request duration
How should we alert?
State-based alerts

Is there a divergence between expected state and actual state of a service?
State-based alerts

Is my service up and/or scrapeable?

absent(up{kubernetes_name="doccserver"}) or sum(up{kubernetes_name="doccserver"}) == 0

Do I have the # of loadbalancers I expect?

sum(up{kubernetes_name="loadbalancer"}) < 3
Threshold alerts

Do any of our measured metrics exceed a lower or upper bound?
Threshold alerts

Is our loadbalancer at 50% capacity in terms of sessions?

\[
\text{max}\left(\frac{\text{haproxy\_frontend\_current\_sessions}}{\text{haproxy\_frontend\_limit\_sessions}}\right) \times 100 > 50
\]

Are 50 percent of tests taking longer than 10 minutes?

\[
\text{max}(\text{test\_duration\_seconds}\{\text{quantile}="0.5", \text{result}="pass"\}) > 600
\]
Common pitfalls
Pitfall #1: Alerting fatigue

Sneha Inguva 📣 3:58 PM
okay so i think i may have confused
myself on how pagerduty is
deduplicating stuff so gonna page
myself a bunch for our testservice

Sneha Inguva 📣 6:15 PM
Crap! Did i just miss a pagerduty
page? What happened
Solution: **Slack and/or Pagerduty**

send only the most **urgent, production alerts** to pagerduty

try out different **promQL queries** to have less **spikey** metrics
Solution: **Dedupe and group alerts**

```plaintext
group_by:
- alertname
- cluster
- service_name
group_wait: 30s
group_interval: 5m
repeat_interval: 3h
```
Pitfall #2: Confused service owners

Sneha Inguva 🧑‍💻 2:33 PM
btw this is where i read about the irate vs. rate
https://www.digitalocean.com/community/tutorials/how-to-query-prometheus-on-ubuntu-14-04-part-1

Sneha Inguva 🧑‍💻 5:22 PM
yep
i hope this irate vs rate thing at least stops some of our flappy alerts

Dan Norris 🧑‍💻 5:23 PM

Sneha Inguva 🧑‍💻 1:45 PM
actually not that useful but haha also of note - rate is better than irate for alerts

Brian Knox ( #observability, RVA ) 🧑‍💻 1:52 PM
alerts always make me irate
Solution: **Docs and suggested alerts**

extensive *documentation* and *tutorials*, accessible from CLI

*prometheus slack channel* for real-time help

standard alert *examples*
Pitfall #3: Who owns what?

Sneha Inguva 🧵 8:53 PM
well that was quite the spike

8:53  ✋ what is neptune, actually, btw?

Mac Browning 🐝 8:56 PM
sweet
wasnt sure, just did a [docc show](#) to find the maintainer
Solution: opinionated **manifest** file

services owner **must** include **maintainer** information

alerts themselves include **descriptions** and **summaries** with several labels

alerts must include **team-specific** receivers
Pitfall #4: Monitoring the monitors

- **Mitchell Anicas** 10:49 AM
  is frog's prometheus down?

- **Joonas Bergius** 10:50 AM
  let's see

- **Sneha Inguva** 10:51 AM
  hey @manicas taking a look!

- **Tommy Murphy** 10:58 AM
  ha and now im taking over
  yeah it sure does look to be down!
Solution: Duplicate promethei and HA alertmanager
Solution: Deadman’s switch

ALERT JustKeepSwimming
IF vector(1)
#1: Automated alerts

utilize user-defined memory and cpu limits for threshold alerts

automatic state-based alerts
#2: Leverage metrics for autopilot

user trusts in our custom controllers and schedulers
collect metrics and build model about resource usage over time
accordingly adjust limits and alerts
#3: **Leverage metrics for autoscaling**

- services based on resource usage, # connections, etc.
- loadbalancers based on # of frontend and backend connections
- # of **worker nodes** based on memory and cpu capacity metrics
a brave new world of **container orchestration**

**OSS whitebox monitoring**

**extensibility**
sources

- The best prometheus tutorials you will ever read, Julius Volz
- Actual Prometheus Website, Julien Friedman
- Kubernetes Project