elastic | 中国开发者大会

Real world examples of using Elastic at Elastic

Michael Russell
Elastic


elastic
Real world examples of using Elastic at Elastic
Michael Russell
Software Engineer - Infrastructure

Me

Facts

- Is: Australian
- Lives: In the Netherlands
- Works: At Elastic
- Responds to: Michael, Mick, Micky, Mike, Mikey, Crazybus and Rusty
- Likes: Food, travelling, gaming, music

We should use X because it will solve our Y problem

Everyone

We should use Kubernetes because it will solve all our problems

A lot of people

Some questions I ask when solving any problem

- When did the problem actually start?
- What problem are you actually trying to solve?
- Does the proposed solution to the problem fix more things than it breaks?
- Are there more important problems we should be solving first?
- Has someone else already solved this problem?

elastic

Has someone else already solved this problem?

Normally yes...

elastic

Does the proposed solution to the problem fix more things than it breaks?

This doesn’t always matter now

elastic

So what changed?

Infra team goals

1. Make sure developers have everything they need to create software

1. Make sure users have everything they need to use the software

1. While doing this, use our software as much as possible
    a. Even when it doesn't make sense
    b. Even when it wasn't designed to do that

elastic

Even when it doesn't make sense

- If it’s possible, users will try to do it
- If it isn’t possible we will know and can either:
  - Stop others from making the same mistake
  - Improve the software to make it possible
  - Write a cool blog post about how to do it
- Bare minimum is to just stuff all of the logs and metrics into Elasticsearch and plan to do something with it later
- Result: Around 50 Elasticsearch clusters with separate use cases

elastic

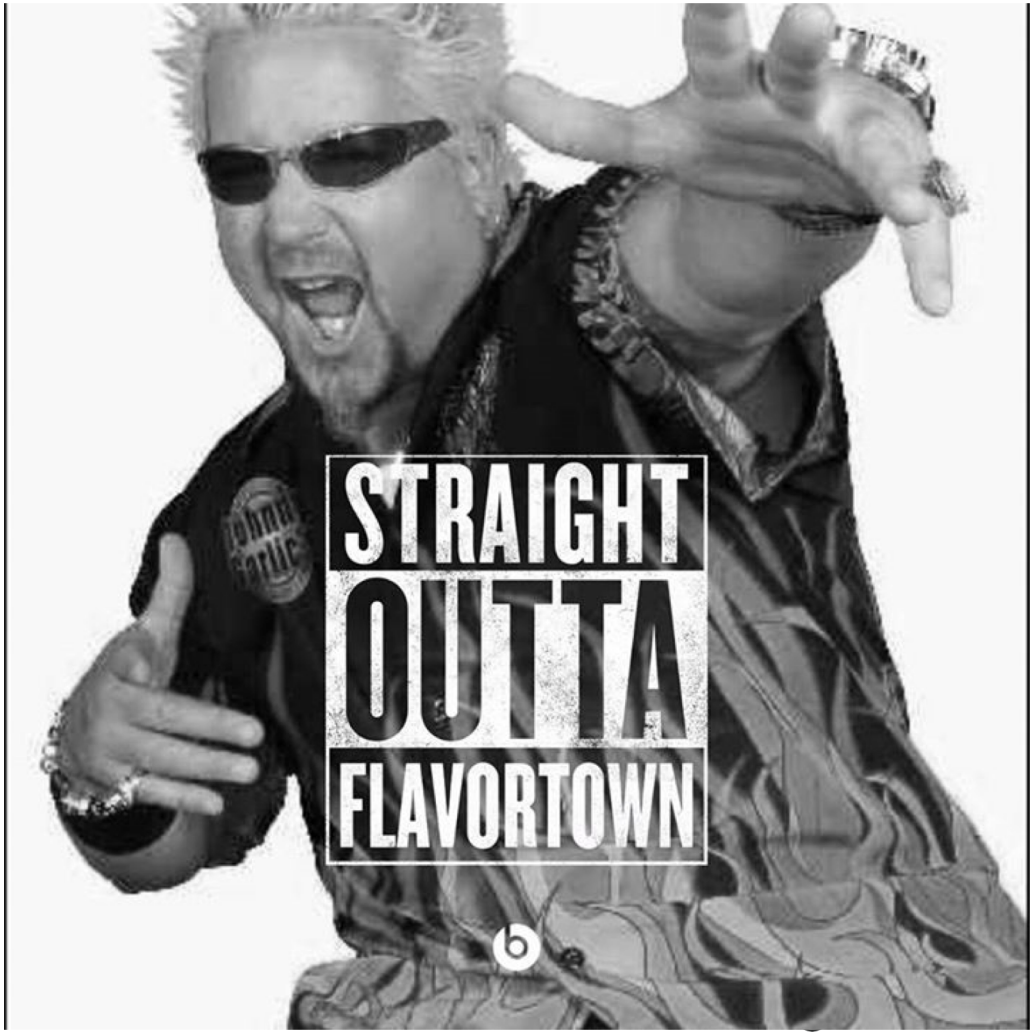
Common Infra use cases

- CI Build statistics
- Kubernetes metrics and logging
- “Normal” Server metrics and logging
- Website uptime and availability
- Having an easy way to use our stack without having to reinvent the wheel

elastic

Things that Infra likes

- Automation
- Having a monorepo
- Good documentation
- Good monitoring
- Having a dashboard for everything
- Guy Fieri



Things we don’t like

- Doing the same thing twice
- Manually running commands
- Clicking in a UI to make a change
- Finding out we don’t have a dashboard for something
- Doing the same thing twice

elastic

Heartbeat - the plan

- Replace external monitoring service
- Use heartbeat
- Use watcher for alerting
- Done???

elastic

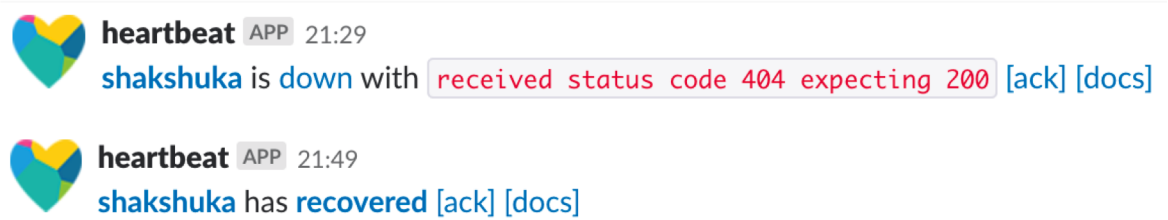
Heartbeat - the reality

- Couldn’t use a single watch
- Couldn’t use a single watch per team
- Couldn’t use a regex pattern to check content of a website
- Couldn’t do recovery alerts
- No way to allow the user to configure where the alert went

elastic

Heartbeat - the solution

- Small python application to generate a seperate watch per heartbeat monitor
- Heartbeat monitor configuration being used to configure the watch
- Automatically generate recovery alerts
- Added upstream support for checking regex patterns



elastic

Heartbeat - the reaction

- Why are you doing that?
- You are creating how many watches?
- Watcher really wasn’t designed for this!

elastic

Heartbeat - the configuration

```
heartbeat.monitors:
  - name: shakshuka
    type: http
    schedule: '@every 1m'
    check.response.status: 200
    urls:
      - 'https://shakshuka.app.example.com'
    fields.watcher:
      docs: 'https://github.com/repo/docs/shakshuka.md'
      alerts:
        slack:
          - '@michael.russell'
```

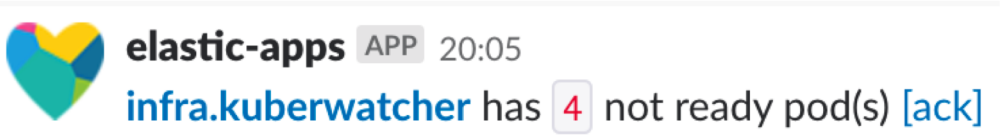
Kubernetes

- We run a managed internal Elastic Kubernetes service
- It has many integrations including metricbeat and filebeat using the Kubernetes module
- We had all the data but weren't doing anything with it.
- Kubernetes is all about abstractions

elastic

Kuberwatcher

- https://github.com/elastic/kuberwatcher
- Inspired by the heartbeat work
- Generates watcher alerts for groups of pods (deployments, daemonsets, cronjobs)
- Configuration can be overridden at the namespace or pod level



elastic

Kuberwatcher configuration

```yaml
apiVersion: v1
kind: Namespace
metadata:
  name: michael
  labels:
    watcher: enabled
  annotations:
    watcher.alerts.slack: '@michael.russell'
    watcher.docs: https://github.com/elastic/kuberwatcher/blob/master/README.md
```

Testing

- We make a lot of software at Elastic
- We test a lot of software at Elastic
- We have a lot of Jenkins jobs
- Even more Jenkins workers

elastic

Reality check

- Our software has to run everywhere
- We support different versions
- That’s a lot of combinations

Product and Operating System

	CentOS/RHEL 6.x/7.x	Oracle Enterprise Linux 6/7 with RHEL Kernel only	Ubuntu 14.04	Ubuntu 16.04	SLES 11 SP4**	SLES 12	openSUSE Leap 42	Windows Server 2012/R2	Windows Server 2016	Debian 7	Debian 8	Debian 9	Solaris/ SmartOS	Ama Linu
Elasticsearch 2.4.x	✓	✓	✓	✗	✓	✓	✓	✓	✗	✓	✓	✗	✗	✓
Elasticsearch 5.0.x	✓	✓	✓	✓	✓	✓	✓	✓	✗	✓	✓	✗	✗	✓
Elasticsearch 5.1.x	✓	✓	✓	✓	✓	✓	✓	✓	✗	✓	✓	✗	✗	✓
Elasticsearch 5.2.x	✓	✓	✓	✓	✓	✓	✓	✓	✗	✓	✓	✗	✗	✓
Elasticsearch 5.3.x	✓	✓	✓	✓	✓	✓	✓	✓	✗	✓	✓	✗	✗	✓
Elasticsearch 5.4.x	✓	✓	✓	✓	✓	✓	✓	✓	✗	✓	✓	✗	✗	✓
Elasticsearch 5.5.x	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓	✗	✓
Elasticsearch 5.6.x	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓	✗	✓
Elasticsearch 6.0.x	✓	✓	✓	✓	✗	✓	✓	✓	✓	✗	✓	✓	✗	✓
Elasticsearch 6.1.x	✓	✓	✓	✓	✗	✓	✓	✓	✓	✗	✓	✓	✗	✓
Elasticsearch 6.2.x	✓	✓	✓	✓	✗	✓	✓	✓	✓	✗	✓	✓	✗	✓
Elasticsearch 6.3.x	✓	✓	✓	✓	✗	✓	✓	✓	✓	✗	✓	✓	✗	✓
Elasticsearch 6.4.x	✓	✓	✓	✓	✗	✓	✓	✓	✓	✗	✓	✓	✗	✓

elastic

Problem

- Build queue was too large which delayed testing
- Builds were failing because other builds didn’t clean up properly
- Infra team had to constantly fix bad workers
- Workers doing nothing during off peak times is expensive
- Sometimes we couldn’t replace a worker because the automation had broken since it was last used

elastic

Attempt 1 with plugins

- Found a GCP dynamic worker plugin
- Initial testing went great
- Basic workflows worked perfect
- On a real world cluster it just didn't work
- I was even warned not to use a plugin!

I solve problems, sometimes the only way to do that is by writing code

Me

Custom Code

- Even after several upstream fixes to the plugin things weren't even close
- Last resort was to write some custom code
- Custom code that stops productivity for all of engineering if it doesn't work
- It will need to have really really good logging and monitoring if we want to be able to trust it

elastic

Logging

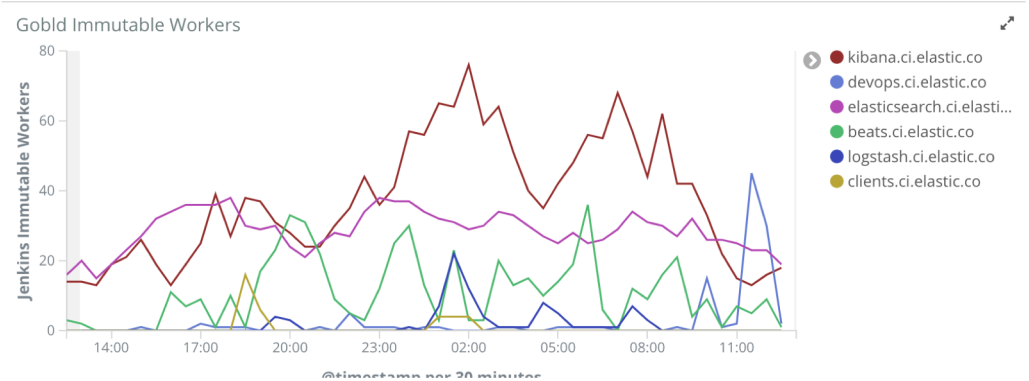
- We chose to deploy it into our Kubernetes platform
- This gave us filebeat logging out of the box
- Logging in JSON meant things were automagically parsed for us!
- The Kubernetes integration meant important metadata like cluster name was already included with the logs!

#	json.age	10
◑	json.idle	true
t	json.level	info
t	json.message	Jenkins worker succesfully destroyed and deleted from Jenkins
◑	json.old	true
◑	json.online	false
t	json.time	2018-11-08T05:28:10Z
t	json.worker	devops-ci-immutable-ubuntu-1541654281777635216
t	kubernetes.container.name	gobld
t	kubernetes.labels.app	devops.ci.elastic.co
t	kubernetes.labels.logtype	json
t	kubernetes.labels.pod-template-hash	828547348
t	kubernetes.labels.release	gobld-gobld-devops.ci.elastic.co
t	kubernetes.namespace	gobld
t	kubernetes.node.name	gke-apps-n1-standard-4-397ca7a1-gw8k
t	kubernetes.pod.name	devops.ci.elastic.co-d6d98c78d-l9pmk

elastic

Metrics

- We now had lots of useful logs
- But we were missing a dashboard!
- Luckily we had structured logging already so we could just log whatever fields we wanted!



elastic

Alerting

- We needed to make sure that we got alerts when something was going wrong
- We had lots of ideas about ways we could build this into the application
- Once again we first tried to use what we already had
- Simple watcher alert looking for error logs worked way better than expected

elastic

Watcher alert

```
"filter": [
  {
    "match": {
      "kubernetes.namespace": "gobld"
    }
  },
  {
    "match": {
      "json.level": "error"
    }
  }
```

```
"condition": {
  "compare": {
    "ctx.payload.hits.total": {
      "gt": 20
    }
  }
}
```

Another day another problem

- Jenkins was not building pull requests
- After some investigation it turned out GitHub has API rate limiting
- We were hitting these limits but didn’t know how fast (no dashboard!)
- Had previously discussed some ideas on writing something custom to get this information into Elasticsearch
- The previous work I had done with Gobld gave me a better (faster) idea!

elastic

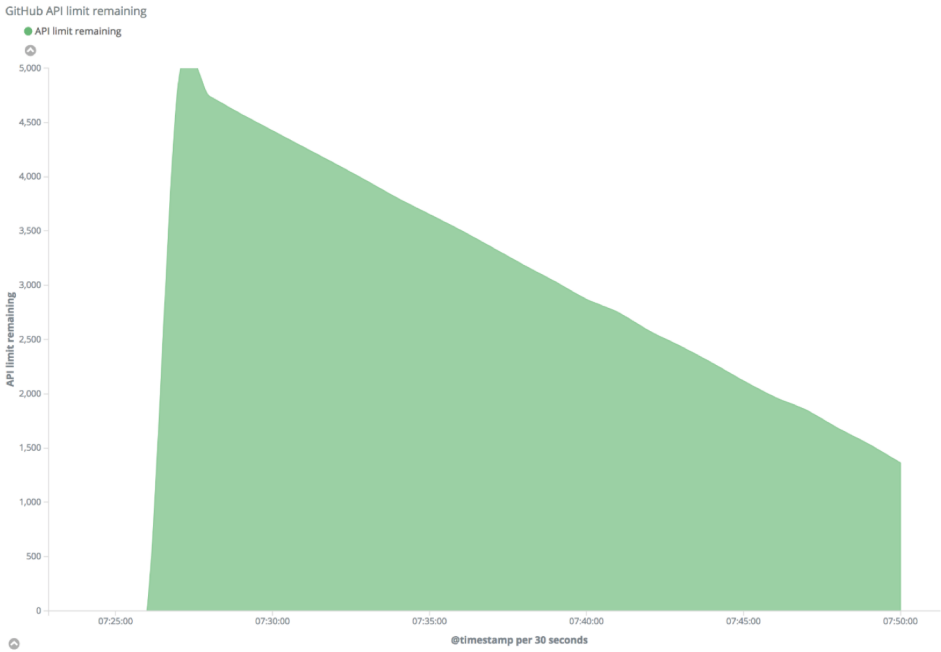
What was old is new again!

```
#!/usr/bin/env bash -e
curl --silent --fail \
  -u ${GITHUB_TOKEN} \
  https://api.github.com/rate_limit | tr -d '\n'
```

At the limit

- Yay we have a graph now
- Wow! It’s much worse than we thought
- Lets try changing things and see if the graph changes!
- One problem is we don’t know what the graph normally looks like :(

Here is how the graph looks at time of writing.



elastic

Could it be that new Gobld thing?

- Me: No way! It doesn't use GitHub
- Me: Ok, I'll turn it off for 5 minutes to prove it isn't Gobld
- *5 minutes later*
- Me: Oh no

elastic

Thank you!

elastic
中文社区

专业、垂直、纯粹的 Elastic 开源技术交流社区
https://elasticsearch.cn/

elastic