Deploying Prometheus at Edinburgh

aka: Making my life as a sysadmin easier.

What is Prometheus?

Officially:

`Prometheus is an open-source systems monitoring and alerting toolkit with an active ecosystem.`

My take:

Prometheus is an easy to use, centralised, stateful, pull-based modular monitoring system with several mature supporting projects.

Why Prometheus? (Some Backstory)

At Edinburgh we used to use a ganglia based monitoring system. This was slightly painful, was set up in a hurry and didn't really tell us much of anything in any great detail. Also (imho) it was ugly.

We were exploring potential alternatives when Gareth gave a talk on Prometheus/Borg at HEPSYSMAN Glasgow January 2018. This peaked our interest.

Prometheus.pdf  University of Glasgow

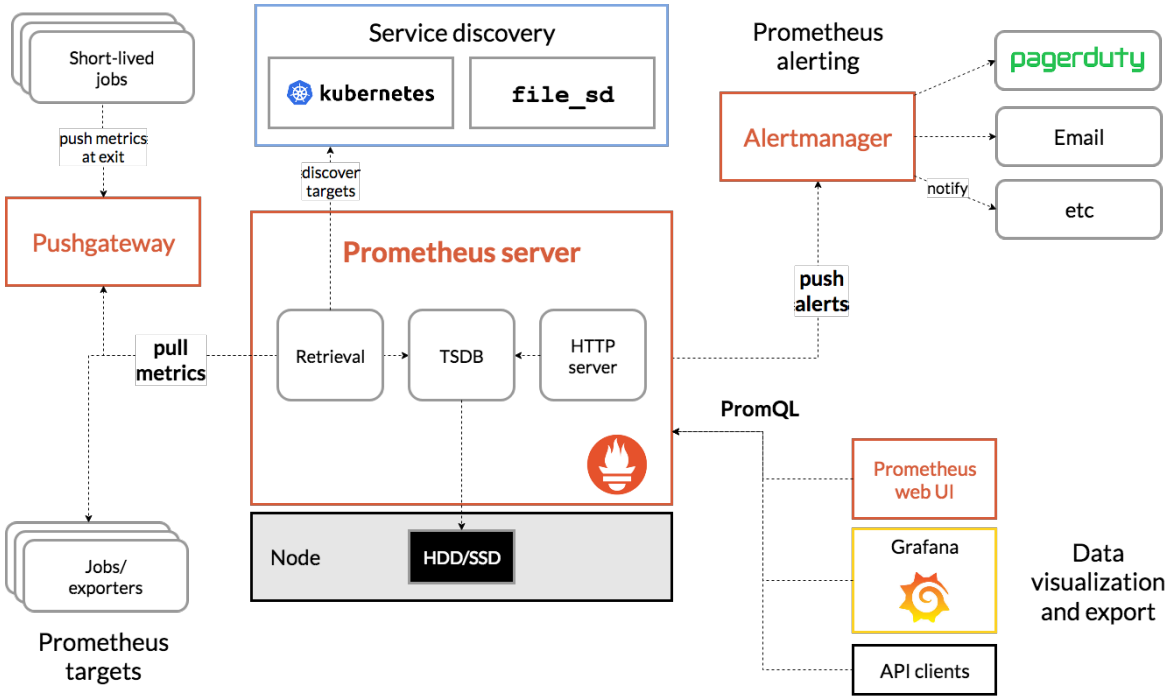
Since then I've seen numerous examples of other places in industry and academia which have adopted the Prometheus + Grafana stack for their service/system monitoring.

How to install prometheus

Simplest Install:

```
wget http://tiny.cc/oa7yaz
tar -xf prometheus-2.11.1.linux-amd64.tar.gz
./prometheus
```

A more complete installation:



Installing

What components are **needed** to monitor a site with prometheus?

- node_exporter: node-github
  This runs on the various machines being monitored collecting system data.

- Prometheus: Prometheus-github
  This is the main tool which collects the monitoring data, stores it and acts on it.

- Grafana: Grafana-github
  This is the plotting/dashboard tool which gets all of the focus when being run day to day.

Installing (Cont)

For *node_exporter* and *Prometheus* all you need to do is grab the binaries and run them. *(and configure your firewall)*.

The configuration of all 3 parts is mostly handled by .yml files or point and click web interfaces.

Bonus points: *node_exporter* can run as an unprivileged user.

A key advantage of these applications being written in go is that everything is statically shipped with the executable.
No 3rd party libraries, no yum install.

But this means no **.service** file or system integration.

Using this monitoring

With node_exporter we are able to monitor the following:

- Traffic sent/received for IPv4 vs IPv6
- CPU/Memory availability/usage
- Number of open files/sockets
- Storage usage/availability
- Available system entropy
- Active login sessions
- Context switching
- Numa statistics
- (not tested) nVidia GPU statistics

What else can prometheus track?

- Background radiation levels from geiger-counter
- How much sunlight our plants get each day
- Temperature from met-office
- Seek time for disk access
- Anything that can exported as numerical data via http

How can this data be accessed

Data is accessed from Prometheus via it's own query language PromQL.

This query language supports useful manipulations of the data that has been collected.

- Maths operations (BODMAS, quantiles, basic logic)
- Pattern matching to select sub-sets of species
- Can return vectors or scalars

For lots more info:
https://prometheus.io/docs/prometheus/latest/querying/

How to display all of this data?

Prometheus has it's own web-ui which allows you to query the data is stores via it's PromQL language.
(There are even external projects to extend this.)

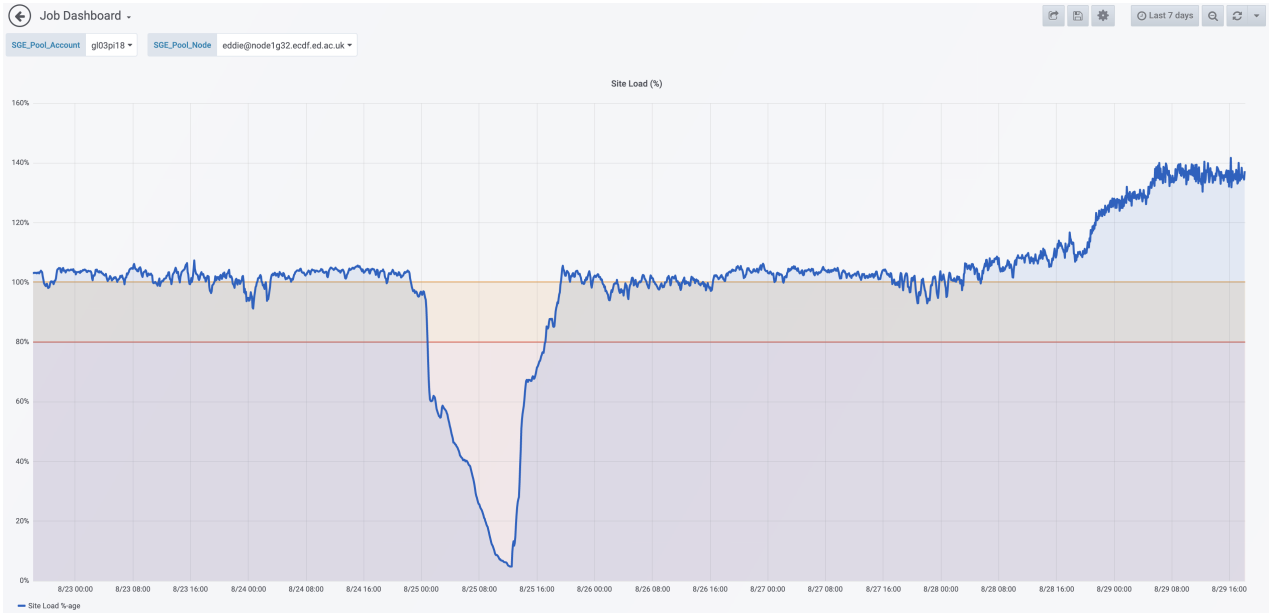
The prometheus UI is very useful for testing that your monitoring is working as you expect or to test/develop complex queries.

Why Grafana?

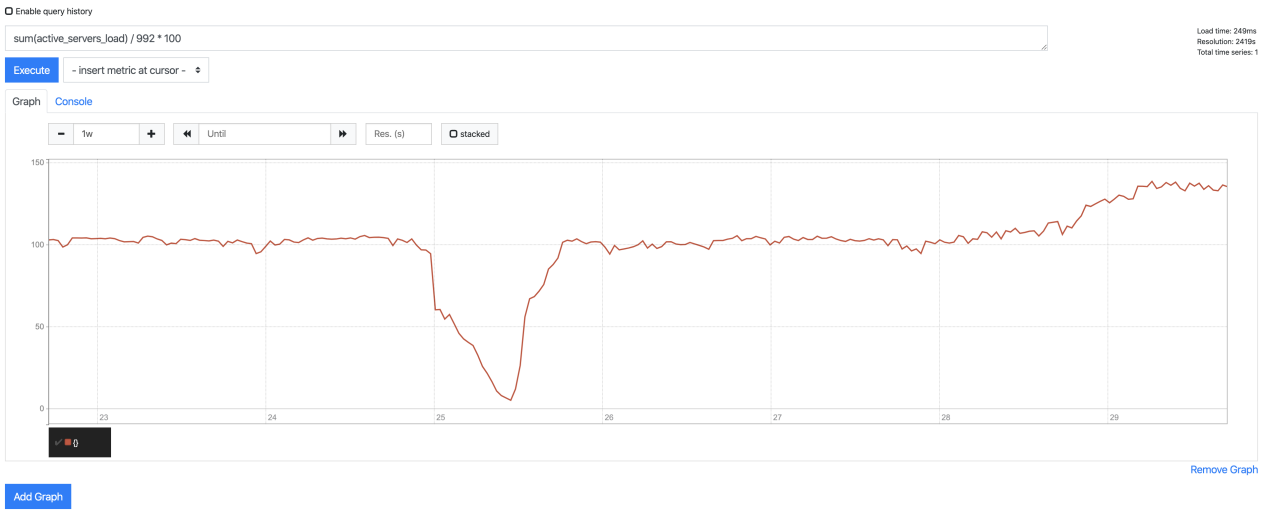
Grafana does a better job of presenting the result of multiple queries as well as building and saving dashboards,

Plotting Data

Grafana:



Prometheus:



Displaying the data

Grafana has a lot of user generated dashboards available which you can download and modify based on your interests.

As a result if you've got a resource you want to monitor chances are someone has probably done most of the legwork for you.

Minor(?) caveat:
There was a major shift in **node_exporter** version 0.16.0
This change renamed and standardised the metrics collected for prometheus. This means a lot of older dashboards are not directly forward compatible.

What can Prometheus allow us to do?

Expanding our installation at Edinburgh, we now have:

- Realtime monitoring of multiple servers
- Realtime monitoring/alerting of WLCG Tier2 services *
- Granular monitoring of containers and services *
- Full SGE exporter *
- Tiered Historical data
- Monitoring of jails in fail2ban

* More details to follow

Monitoring Tier2 Services (1/2)

We want to closely monitor the Tier-2 services at our site.
This means we can be pro-active rather than re-active in dealing with issues.

To do this I see 2 potential options:

1. Monitor/export the logs from services on the host and build a complex system to check for failures.
2. Check to see if a service is actively listening for new remote connections.

We went for for option 2.

Monitoring Tier2 Services (2/2)

Monitoring the Tier2 services at our site at the network level.

This is achieved through deploying a **blackbox_exporter** service on our monitoring box. This attempts to open a tcp/udp connection against a target port number for a host.

Pros:

- Easy to setup (but verbose)
- Easy to understand/test/develop
- Can understand http responses
- Can be run remotely
- Can understand http responses

Cons:

- Floods some logs with un-authorized connection attempts
- Can potentially look like unusual network activity

Alerting from the Tier2 (1/2)

In addition to everything else mentioned here Edinburgh is subscribed to a scotgrid team on the CERN mattermost instance. (This could be slack)

In order to get notified when anything happens of merit we have setup an instance of the **alert_manager** service.

When a (PromQL) condition is satisfied within prometheus it fires an alert which is handled by this additional service. It then manages these alerts and sends a notification to mattermost.

Some excellent examples of things to alert on:
https://awesome-prometheus-alerts.grep.to/rules.html

Alerting from the Tier2 (2/2)

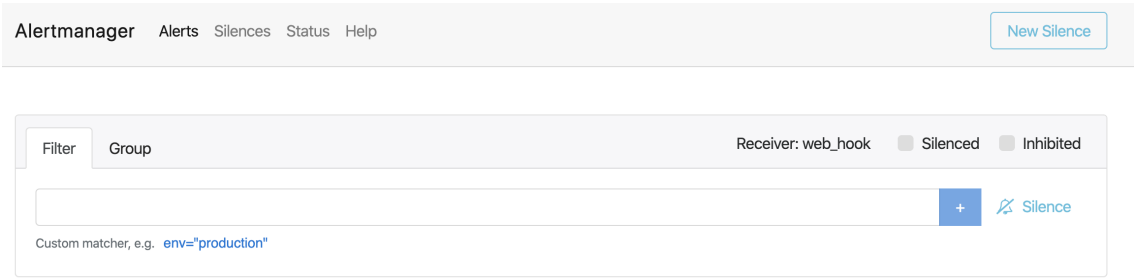
Using node+prometheus+blackbox+alertmanager we get mattermost notifications for:

- Services which stop listening for new connections
- Disks reaching $>$ 90% capacity
- Compute nodes dropping below 50% usage
- Entropy dropping too low*

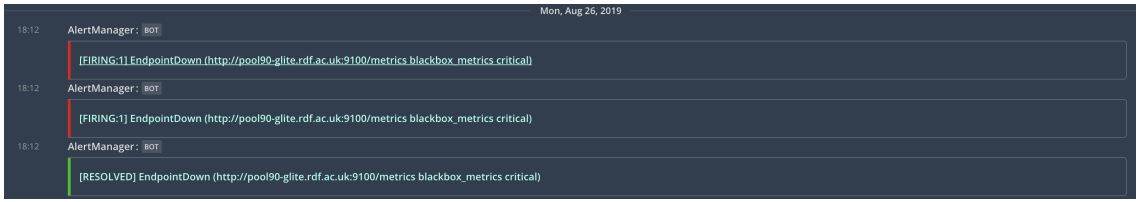
* We're mitigating this now, ask me offline if you want to know more

Being alerted

1. **blackbox_exporter**: Service has stopped listening on a port
2. **Prometheus**: Send an alert to alert_manager.
3. **alert_manager**: Send event to mattermost:



4. **mattermost**: Inform sysadmin (phone buzzes):



5. Clickable links back up the chain allow us to quickly explore via web-ui what fell over.

Monitoring Services/Containers

Google has another project called **cAdvisor**. This service records and exports metrics associated with containers and cgroups.

Although primarily designed to work with docker this can (with a few configuration changes*) be used to export metrics associated with *normal* systemd services.

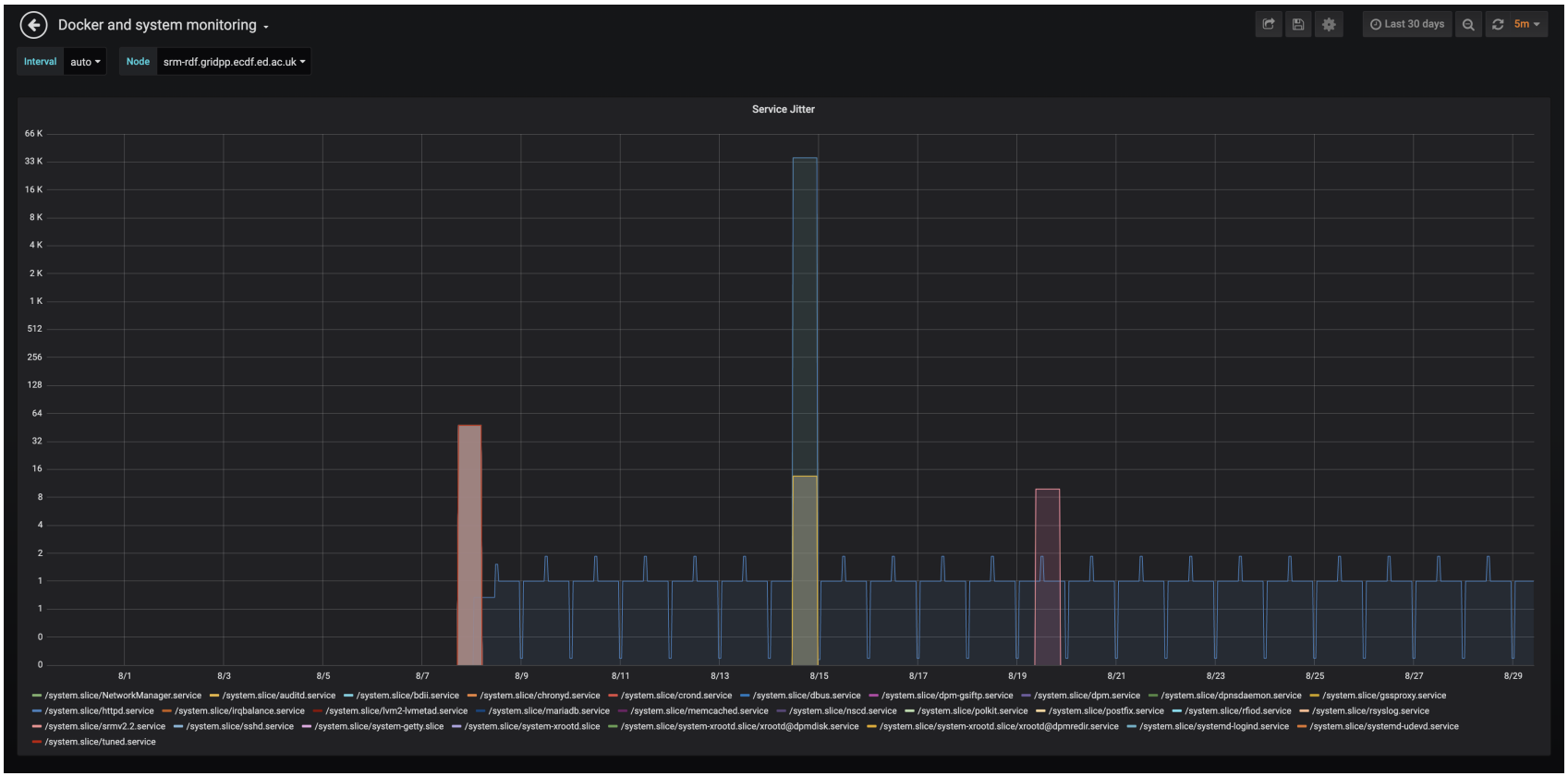
This allows us to get a fine grained view of Memory/CPU usage on a per-application level within a server.

* make sure you configure systemd to monitor usage of cgroups.

For the interested.

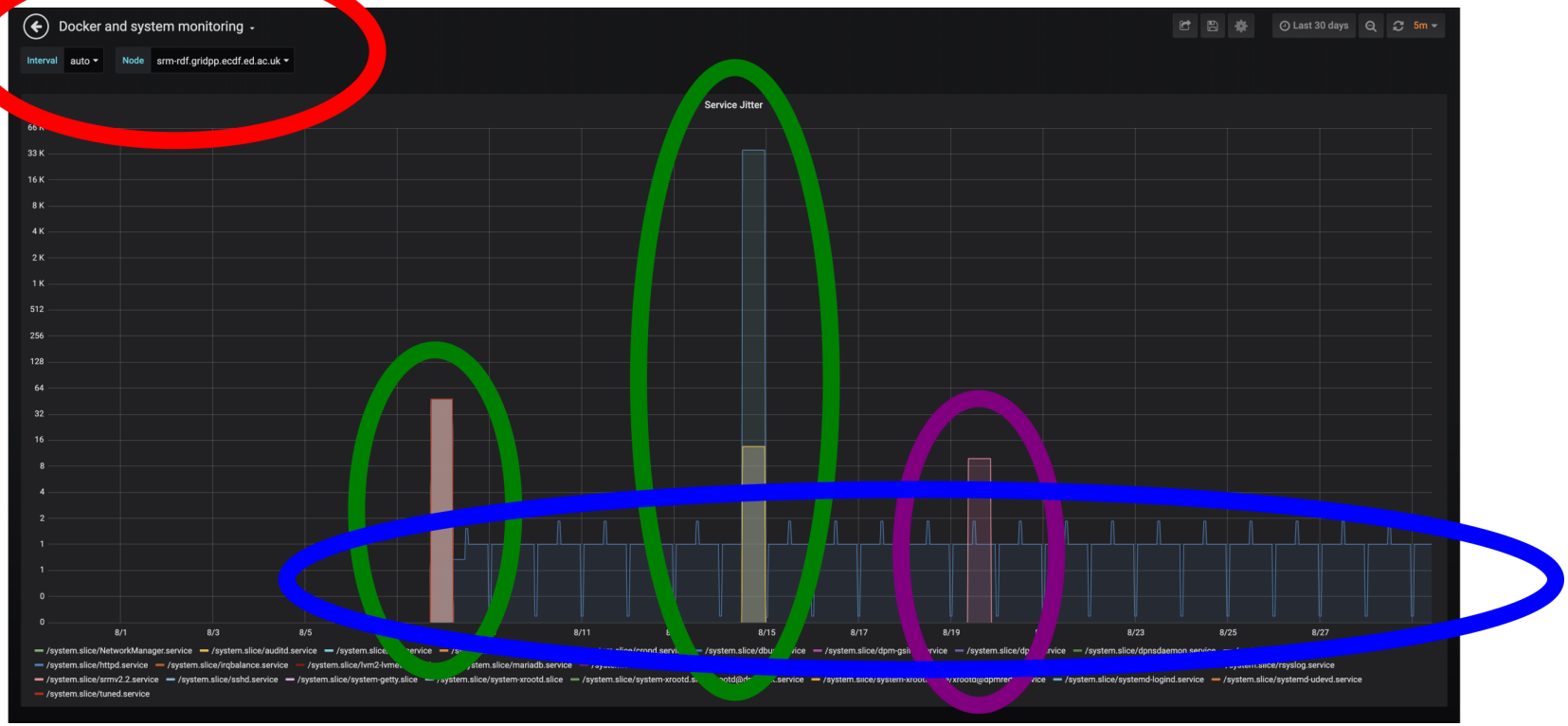
Monitoring Services (1/3)

Can monitor for a service restarting using systemd:



Monitoring Services (2/3)

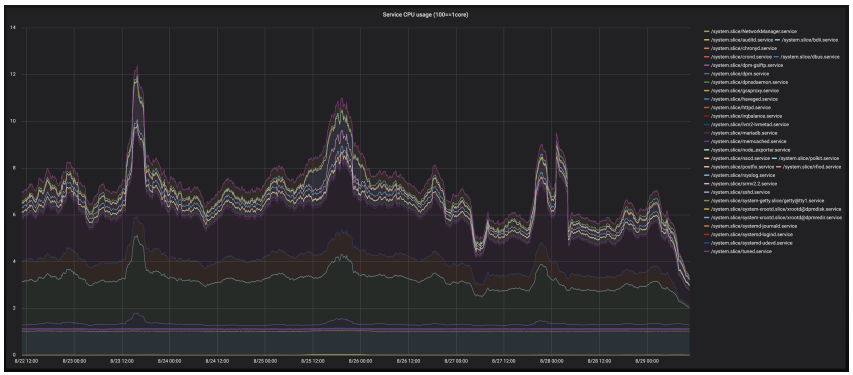
Selected Storage Server, System Reboot,
Storage Service Restart (Intervention),
Apache being regularly restarted.



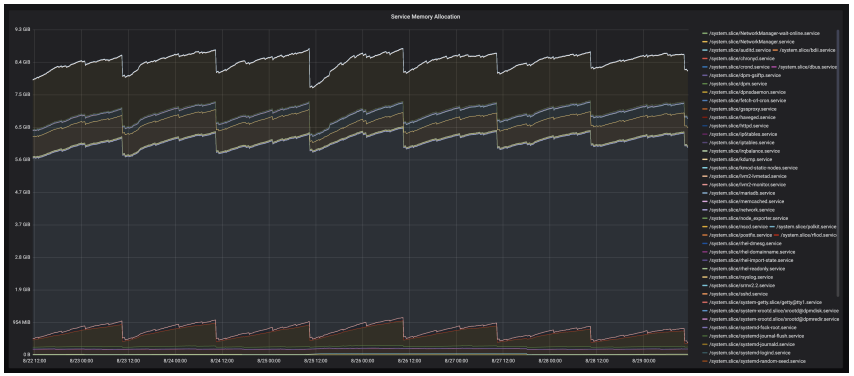
Monitoring Services (3/3)

Using cgroups and cAdvisor we can now monitor the CPU/Memory usage for individual services on a box:

CPU



Memory



Using this we can potentially monitor for individual services running into problems such as too high/low CPU/Memory usage.

SGE monitoring

Why develop this?

Edinburgh is unusual in that we don't directly control our SGE.
Previously had no insight in to what was going on.
Solutions to monitor this had been tried in the past and failed.
...

What can we do now?

- Track issues in realtime supporting our site admins.
- Realtime view of site activity.
- Quickly spot problems in site/jobs.
- Show the site is behaving in a reliable/reproducible way.

SGE monitoring

What else can we do with an SGE exporter?

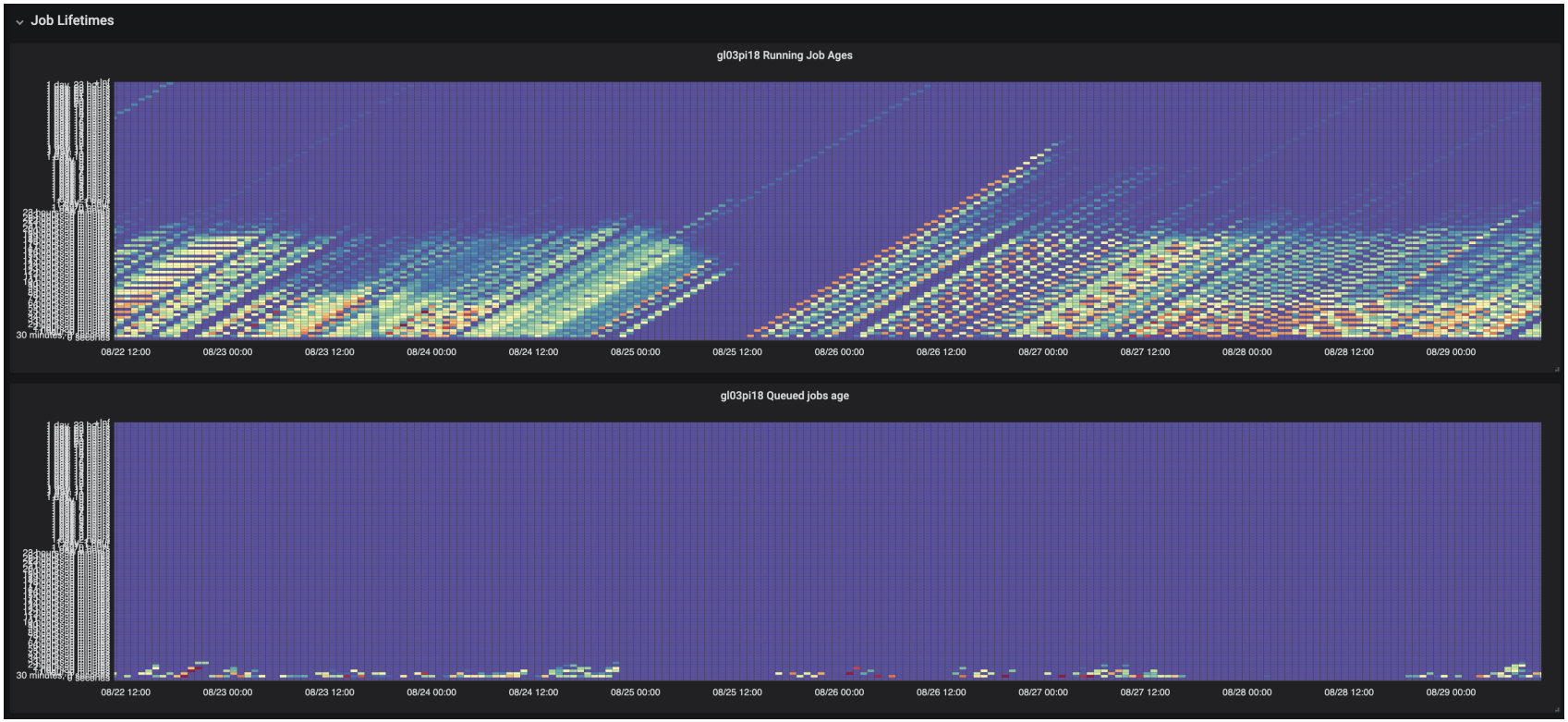
- Realtime data on average lifetime of running jobs
- Realtime data on where jobs are running
- See how long jobs have been queued for
- See patterns in how jobs are submitted for different VOs
- Spotting patterns in job submission

Realtime Job Dashboard

Default landing page for Edinburgh Tier2 monitoring

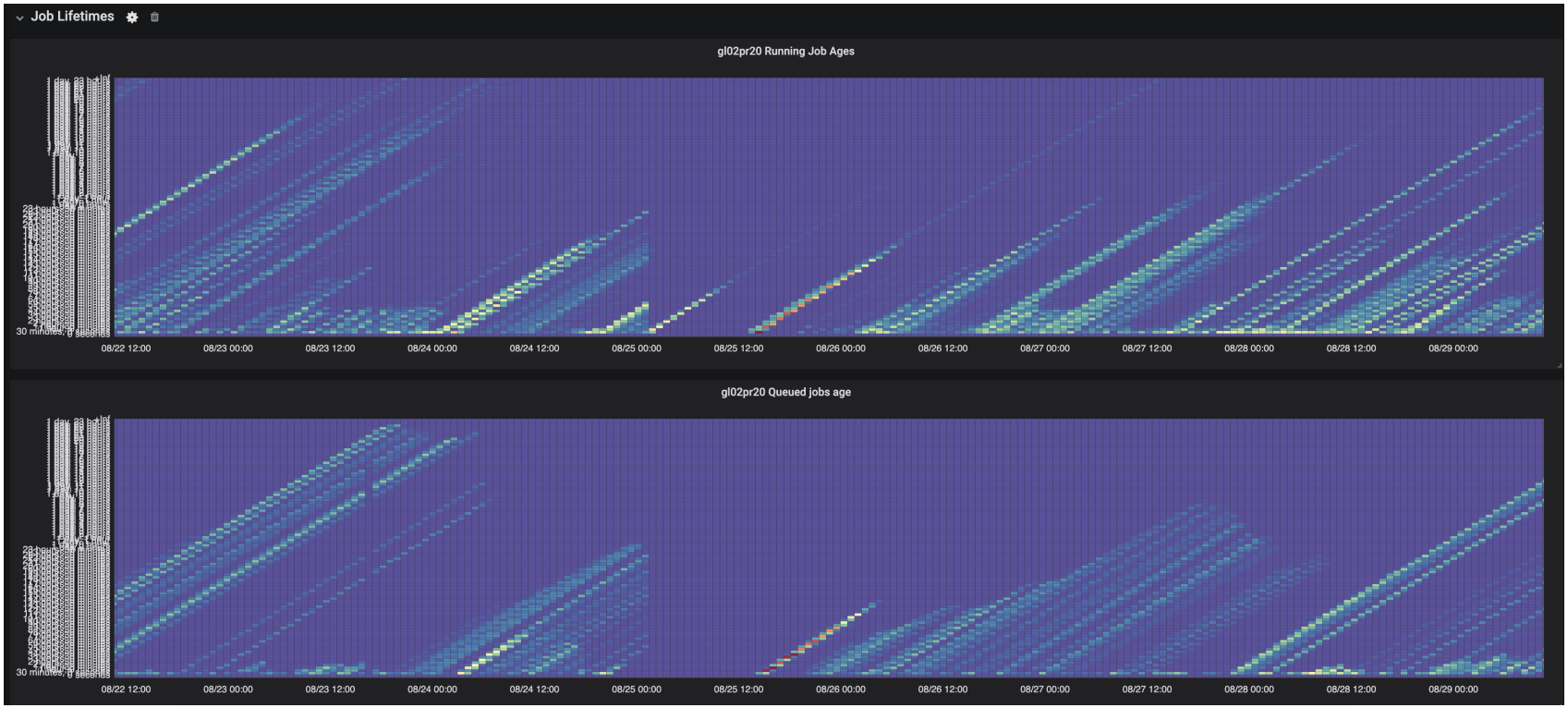
Job lifetimes (1/3)

Running and Queued job lifetimes for a HEP experiment



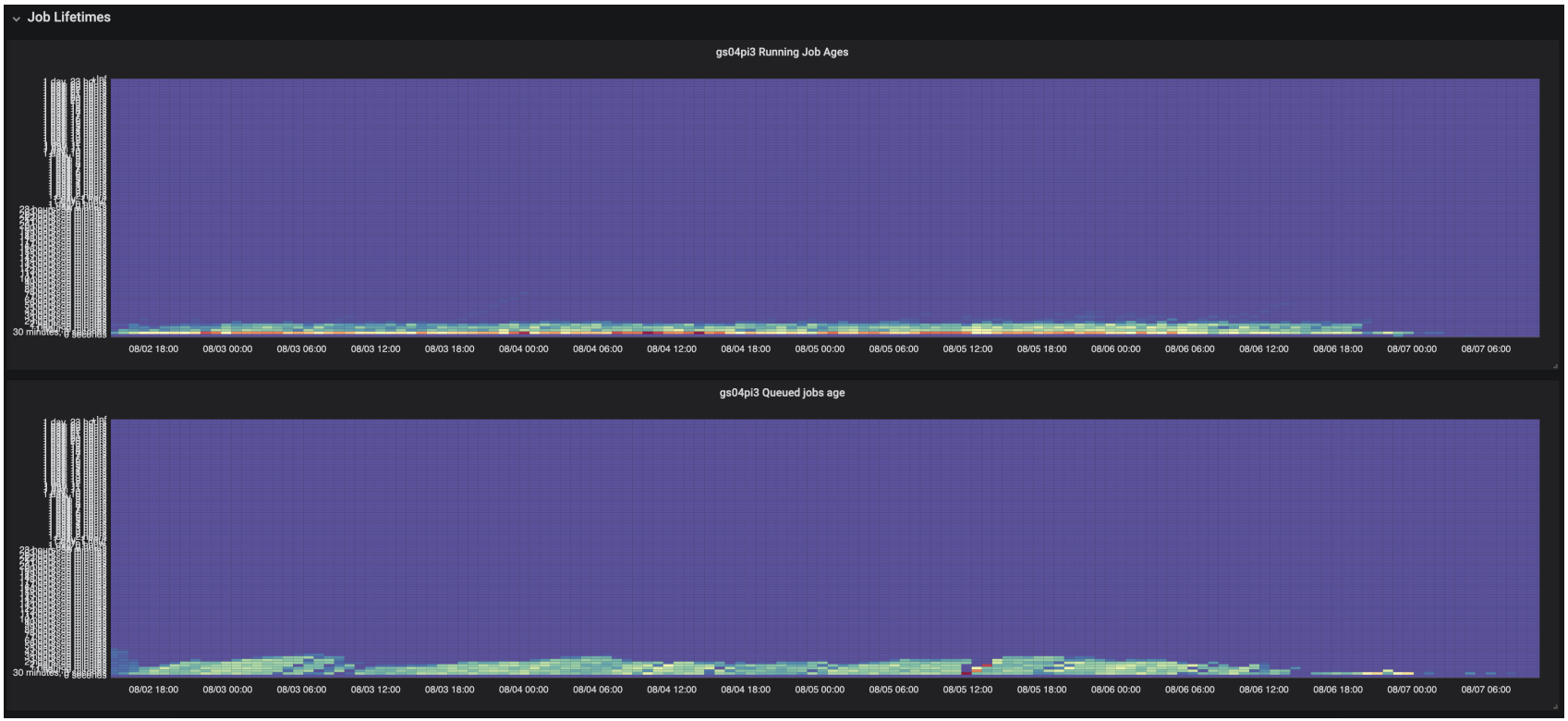
Job lifetimes (2/3)

Running and Queued job lifetimes for another HEP experiment



Job lifetimes (3/3)

Running and Queued job lifetimes for an IRIS experiment



Summary

From our experience at Edinburgh Prometheus is:

- Easy to use
- Fun to play with
- Simple to extend/configure
- Easy to install/maintain using Docker
- Extremely flexible/modular

I think the last of these is the real reason that Prometheus seems to be taking off in industry.

Conclusion

I would strongly encourage people looking to do monitoring to look at Prometheus.

It's a powerful standalone tool and can work well as a component in a more complete monitoring solution.

If I've peaked your interest, or convinced you your site needs this there are some useful links on the remaining slides.

Further Reading

If you want to read up on configuring the exporter/prometheus/grafana stack I recommend the following articles:

https://devconnected.com/complete-node-exporter-mastery-with-prometheus/

https://prometheus.io/docs/guides/cadvisor/

https://medium.com/@wbassler23/getting-started-with-prometheus-pt-1-8f95eef417ed

Installing extra components

These are some **optional** components which would make your Prometheus monitoring even better.

- blackbox_exporter: blackbox-github
  This service logs attempts to connect to open ports on remote machines.
- alertmanager: alertmanager-github
  This service manages alerts.

- victoria_metrics: VictoraMetrics-github
  An excellent even more powerful tool to manipulate/store/search data from prometheus.
- cadvisor: cAdvisor-github
  Excellent tool from Google which is designed to monitor and record metrics relating to docker/containers.
- influxdb: influxdb-docs-prometheus
  Excellent tool for recording large amounts of data long term with complex user-defined retention rules.

BACKUPS

Security

It's worth noting Prometheus and most other projects (except grafana) do not come with multiple-user or authentication(login) support.

They do however support TLS. Setting this up however takes time and effort but gives the usual benefits of https over http. Not *currently* used in Edinburgh but I'm exploring how to roll this out using ansible.

However, you're still running a small standalone web browser on each of your boxes in the system.

In Edinburgh we try and stay secure by:

- Only allow access from a trusted IP (or range).
- Keep components updated regularly.
- Be careful about what goes into public monitoring dashboards.