A hybrid system for monitoring & automated recovery @ Glasgow Tier-2 cluster

Motivation

We have been looking at how an increased level of automation can allow significantly more CPU-power to be delivered with the existing staff-effort.

Most sites already use all sort of automation tools, from system provisioning, remote management, monitoring and issue tracking. This allows WLCG sites to run many instances with relatively little manpower.

For this project we have developed an automated recovery system that can resolve the most common issues, such as a disk clean-up or a system reboot, creating a Tier-2 cluster that is more resilient, more efficient and requires less human intervention.

Outline

- Glasgow Tier2 site description
- Cluster Monitoring & Logging
- Alerts & Automation
- Visualization
- Conclusions & Outlook

ScotGrid Glasgow:

Emanuele Simili, Gareth Roy (†), Gordon Stewart, Samuel Skipsey, David Britton

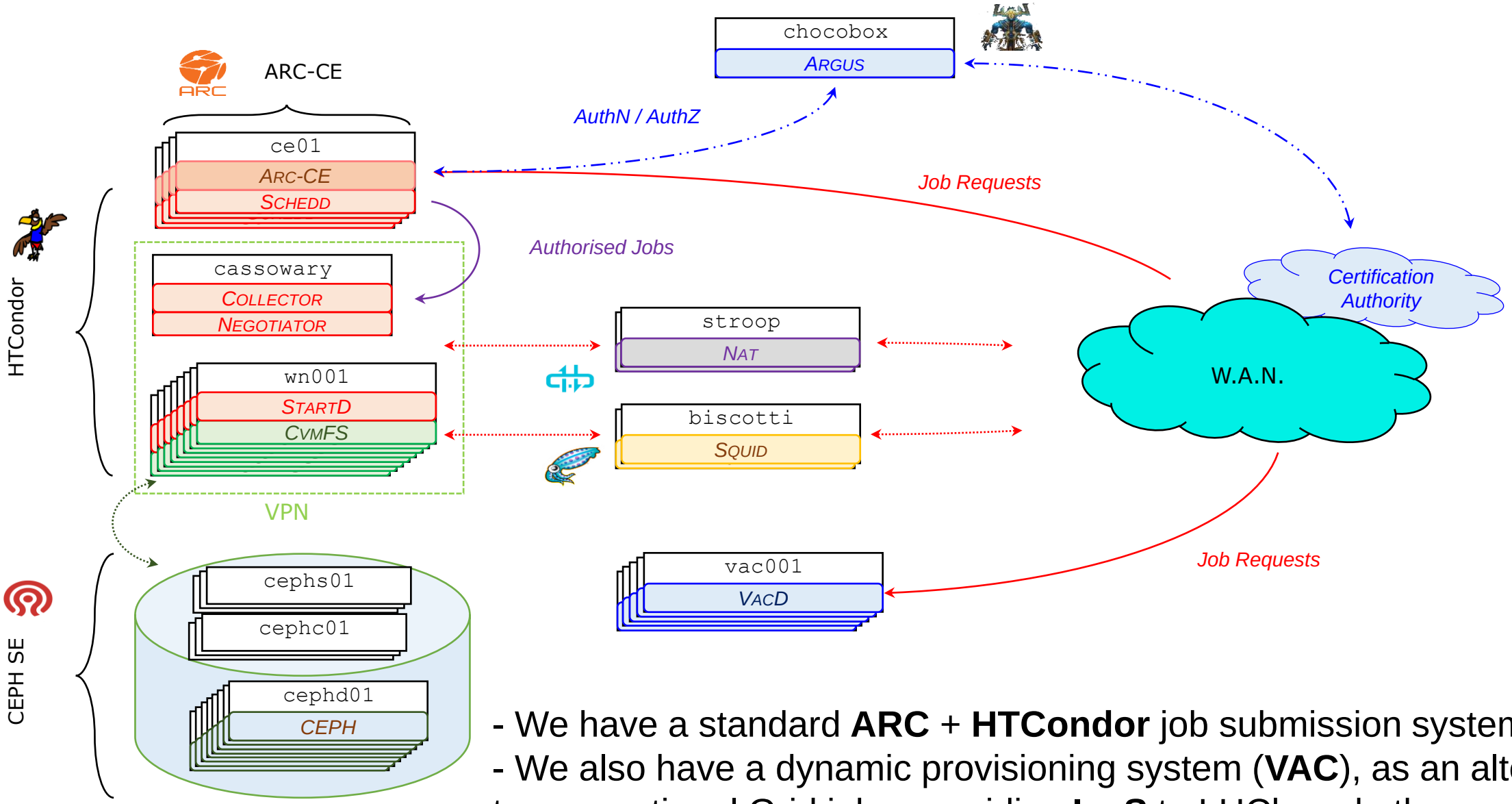
ScotGrid Glasgow Site

At present, ScotGrid Glasgow consists of:

- ~6000 CPU cores
- 4.9 PB storage (3.2 PB CEPH + 1.7 PB DPM) *
- 160 Gb/s internal network bandwidth

For a rough total of about 61 KHS06

Cluster Map

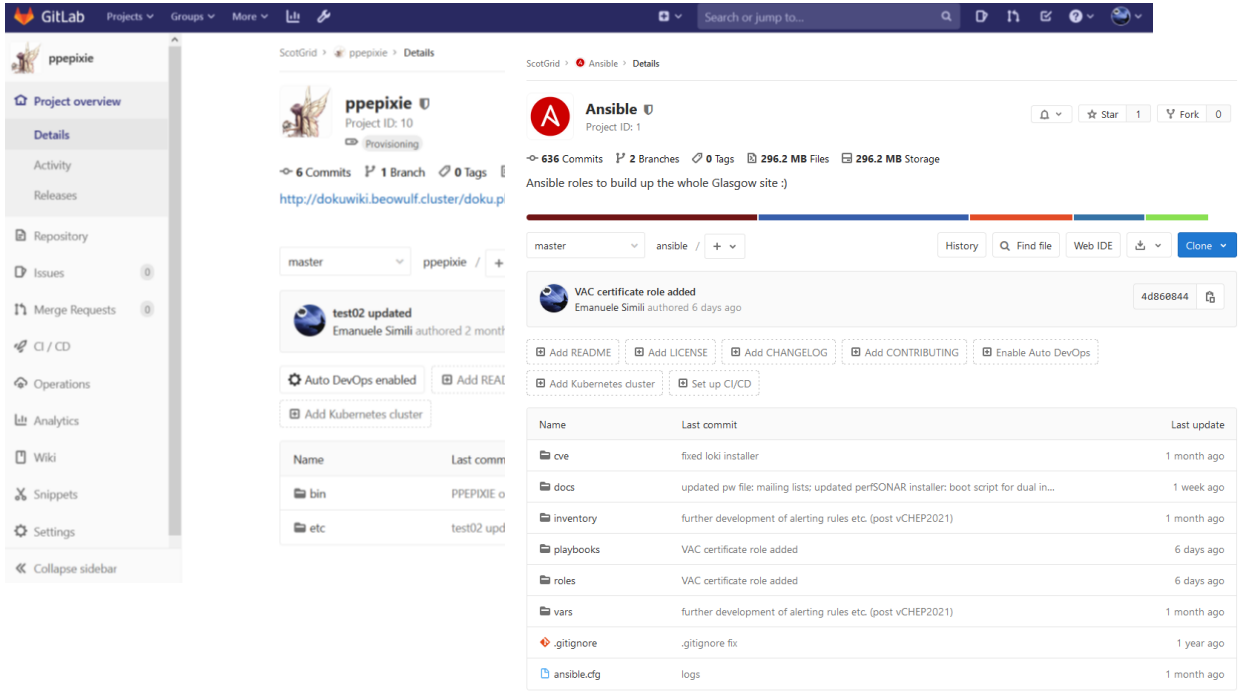


- We have a standard **ARC** + **HTCondor** job submission system
- We also have a dynamic provisioning system (**VAC**), as an alternative to conventional Grid jobs, providing **IaaS** to LHCb and other experiments

Automation Tools

As a site of significant size, we use many sys-admin empowering tools:

- A network provisioning system (**PPE PiXiE**)
- A remote management tool (**Ansible**)
- A central monitoring & logging system (**PLG**)



And we have invested quite some time in preparing such environments…

Example Procedures

Create a new workernodes (assuming node **00x** is cabled and plugged in):

- Fill in MAC, IP and HostName (in Ansible), then run the play-book to update the network
  nedit roles/dhcp/files/hosts.conf roles/dnsmasq/files/10.0.0.0_8
  ansible-playbook update_networking.yml
- Create a PXE system profile, then boot the machine to install OS
  nedit systems/nodes/wn00x.yml
  systemctl restart pxecli
- Finally run ansible to install the software and configure the workernode
  ansible-playbook -l wn00x workernodes.yml

Add a new VO to the standard job submission system:

- Fill in the VO coordinates in a proper Ansible role, then run it
  cp -r roles/vo-template roles/vo-xxx
  nedit roles/vo-xxx/files/xxx.grid-mapfile
  ansible-playbook add_vo.yml -extra-vars “vo=xxx”

Clean CVMFS cache and reboot node **00x**:

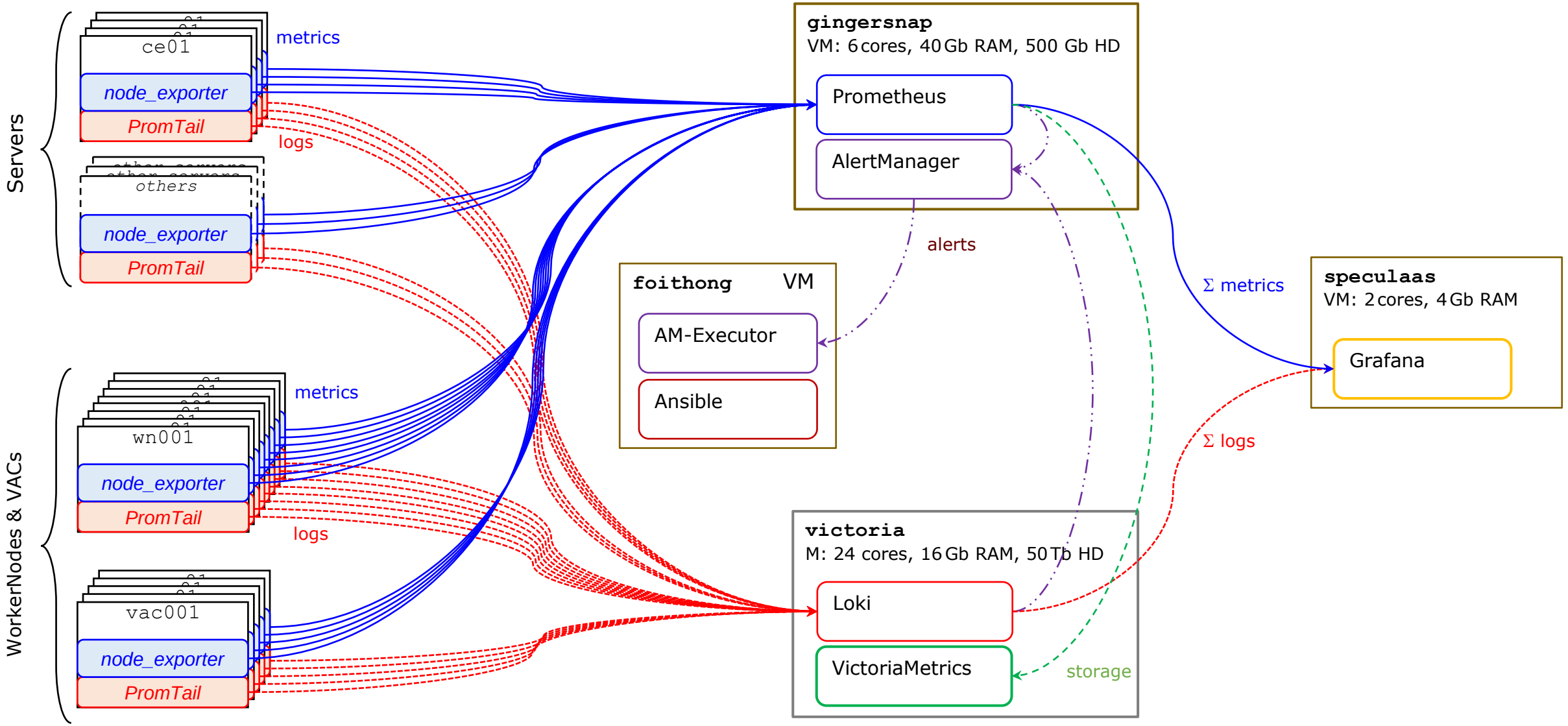
- Just run the ansible play-book
  ansible-playbook -l wn00x cleanCVMFS.yml

Monitoring & Logging (1)

Our system is built on **Prometheus**, **Loki** and **Grafana** (**PLG**)

- Metrics are exported by **node_exporter** (installed on nodes) and collected by a central **Prometheus** instance
- Logs are exported by **PromTail** (nodes) and collected by **Loki** (central)
- Data visualization and querying is done by **Grafana**, which pulls data from both **Prometheus** and **Loki**
- Alert are sent out by **AlertManager**, installed on the Prometheus server, and received by **AM-Executor**, installed on the configuration management server that also runs **Ansible**
- Long term storage of metrics is done by **VictoriaMetrics**, installed on the same server as **Loki** to take full advantage of the large storage

Monitoring & Logging (2)



Exported Metrics

node_exporter produces by default a large set of metrics (e.g., CPU load, network activity, disk I/O, …)

And it can be further customised with ad-hoc metrics:

Metric	Purpose	Tool
Temperature HD	check for overheat	S.M.A.R.T. tools
Temperature CPU	check for overheat	IPMI tools
HTCondor jobs stats	check the status and ownership of running jobs	condor_status
VAC jobs stats	check the number and status of virtual machines	check-vacd
Running processes	runtime and resource usage	ps + grep
Number of Reboots	static counter to keep track of unexpected restarts	cron job
Static info	check for unexpected changes in hardware (e.g., memory bank fail)	lscpu, vmstat , lsblk

Exported Logs

PromTail tails the content of local logs and sends it to a central **Loki**.

We have chosen this tool (instead than the widely spread **ElasticStearch**) because:

- it well integrates with Grafana
- it is very easy to set-up …

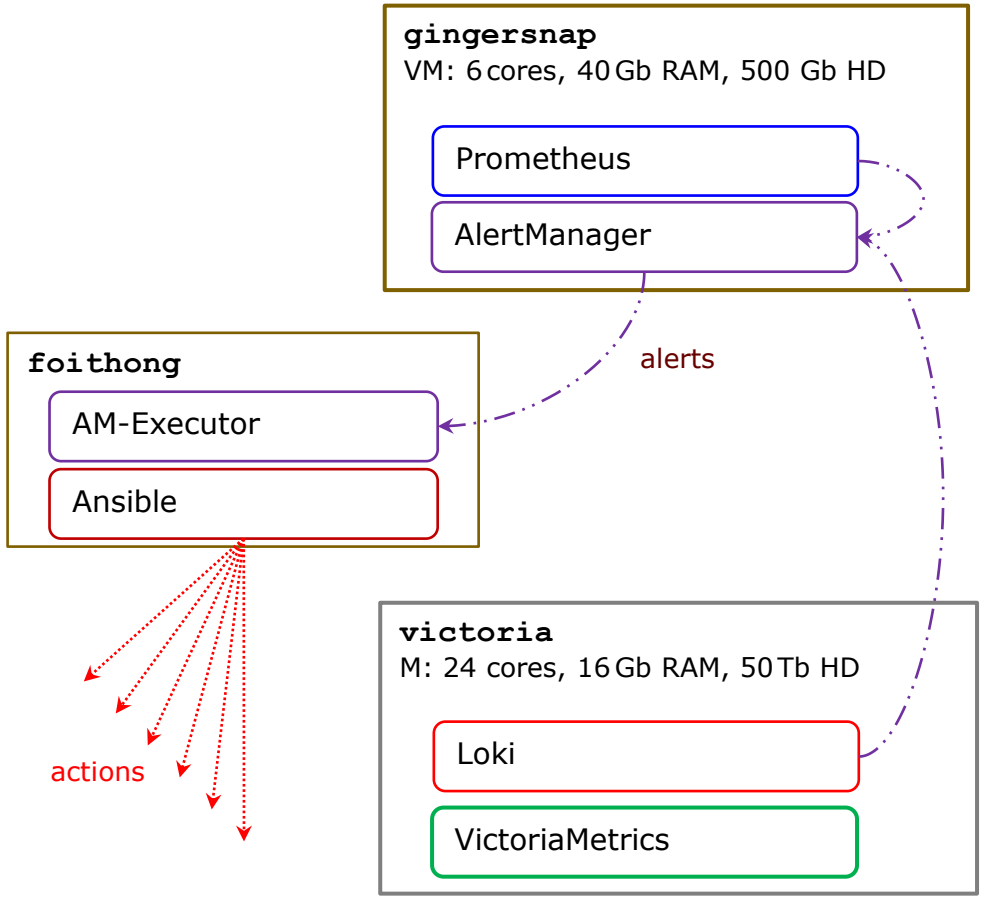
Logs to scrapes are specified in the configuration (we scrape default system logs* and specific service logs)

** we do not export the journal because it clogs the system*

Service	Exported Logs
all	/var/log/messages /var/log/secure /var/log/cron /var/log/yum.log
HTCondor	/var/log/condor/
ARC-CE	/var/log/arc/ /var/log/arc/bdii/
ARGUS	/var/log/argus/pap/ /var/log/argus/pdp/ /var/log/argus/pepd/
Squid	/var/log/squid/
Apache	/var/log/httpd/
Ansible	/var/log/ansible/ /var/log/alerts/

Alerts & Automation (1)

Alerts are generated according to the outcome of conditional tests involving metrics or queries against certain criteria. Both **Prometheus** and **Loki*** can be configured to trigger alerts when a custom set of conditions is met.

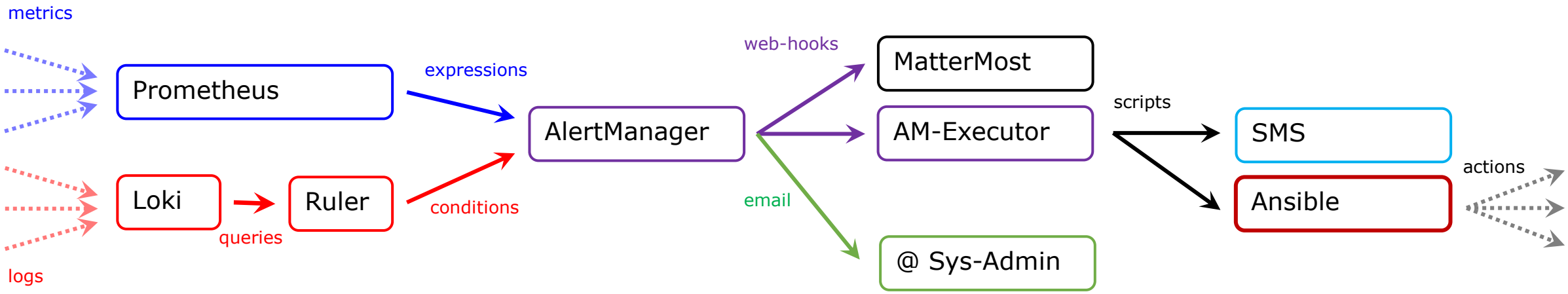


Alerts are handled and sent out by the **AlertManager** …

… and are received by the **AM-Executor**, which performs some pre-encoded actions.

* Loki's alert system is new in v2.0 and uses a specific plug-in (`ruler`), which in turns calls `AlertManager` …

Alerts & Automation (2)



In summary, these are the stages of an alert :

0) **Prometheus** or **Loki** verifies a test condition and triggers **AlertManager**

1) An alert messages is generated by **AlertManager** and sent as a web-hook

2) **AM-Executor** is an HTTP server that listens for alerts, parses their content, and triggers custom commands matching the alert type and source

3) By running **AM-Executor** on the same management server that runs **Ansible**, we can harness the full power of Ansible remote management, with a variety of play-books prepared beforehand

AM-Executor

AM-Executor is configured with a custom script that defines the action to be taken based on the alert:

- Actions implemented in a case-switch
- Multiple alerts handled in a for loop
- The script can call an Ansible play-book or trigger another script (SMS)
- Privileges for Ansible remote management are set by a password-less SSH key *

* This can raise security concerns, and we'll use a better solution in the future

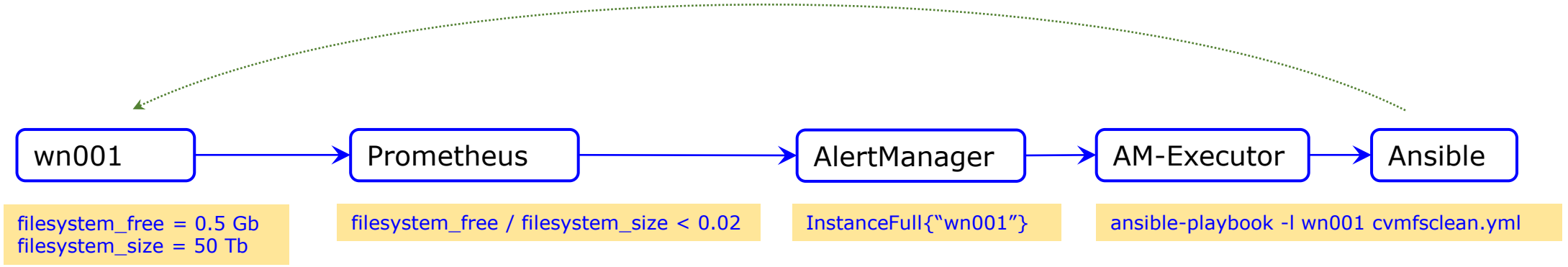
```
# case-switch to assign an action
case $AM_ALERTNAME in
    InstanceFull) ansible-playbook -l $AM_NODE /etc/ansible/playbooks/do_cvmfsclean.yml -b
        echo "> ansible-playbook -l $AM_NODE do_cvmfsclean.yml -b" >> $AM_LOGFILE
        ;;

    CpuHot)    ansible-playbook -l $AM_NODE /etc/ansible/playbooks/do_shutdown.yml -b
        echo "> ansible-playbook -l $AM_NODE do_shutdown.yml -b" >> $AM_LOGFILE
        ;;

    HdHot)     ansible-playbook -l $AM_NODE /etc/ansible/playbooks/do_shutdown.yml -b
        echo "> ansible-playbook -l $AM_NODE do_shutdown.yml -b" >> $AM_LOGFILE
        ;;

    NodeLazy)  ansible-playbook -l $AM_NODE /etc/ansible/playbooks/do_clean-enable.yml -b
        echo "> ansible-playbook -l $AM_NODE do_clean-enable.yml -b" >> $AM_LOGFILE
        ;;

    VacLazy)   ansible $AM_NODE -m reboot -b
        echo "> ansible -l $AM_NODE -m reboot -b" >> $AM_LOGFILE
        ;;

    WayTooHot) ansible-playbook /etc/ansible/playbooks/do_shutdown.yml -b
        echo "> ansible-playbook do_shutdown.yml -b" >> $AM_LOGFILE
        ;;

    TooHot)    . /etc/am-executor/send_sms_alert.sh
        echo "> . /etc/am-executor/send_sms_alert.sh" >> $AM_LOGFILE
        ;;

    *) echo "Alert not recognised:  $AM_ALERTNAME  !" >> $AM_LOGFILE
        ;;
esac
```

Alert Conditions

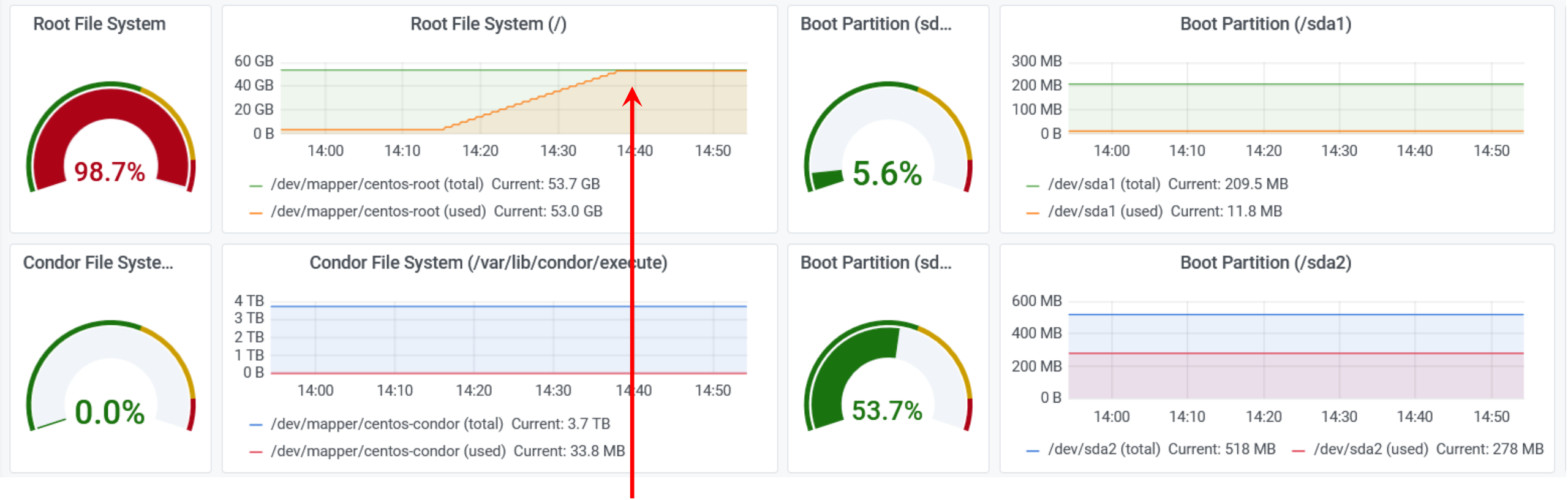
So far, only a limited number of alert-cases have been implemented:

Alert condition	Alert Name	Action	Mail Chat	SMS
$\text{avg}(\text{node\_sensors\_ambient}\{all\}) > 25$	TooHot	/	✓	✓
$\text{avg}(\text{node\_sensors\_ambient}\{all\}) \geq 30$	WayTooHot	shutdown$\{all\}$	✓	
$\text{node\_hwmon\_temp}\{node\} > 80$ $\vee$ $\text{node\_sensors\_hddtemp}\{node\} > 35$	CpuHot HdHot	shutdown$\{node\}$	✓	
$\frac{\text{node\_filesystem\_free}\{node\}}{\text{node\_filesystem\_size}\{node\}} < 0.02$	InstanceFull	clean$\{node\}$	✓	
$\text{avg}(\text{node\_condor\_cpu}\{node\})[1h] \leq 1$ $\wedge$ $\text{avg}(\text{node\_condor\_cpu})[1h] > 10$	NodeLazy	reboot$\{node\}$ + enable$\{node\}$	✓	
$\text{rate}(\{node\} \mid \sim \text{“[Ee]rror”}[1h]) > 0.1$	ManyErrors	disable$\{node\}$	✓	

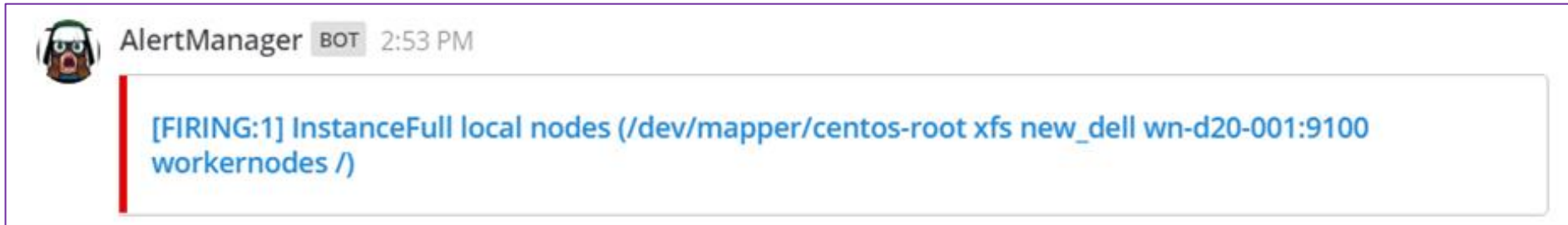
Example (instance full):



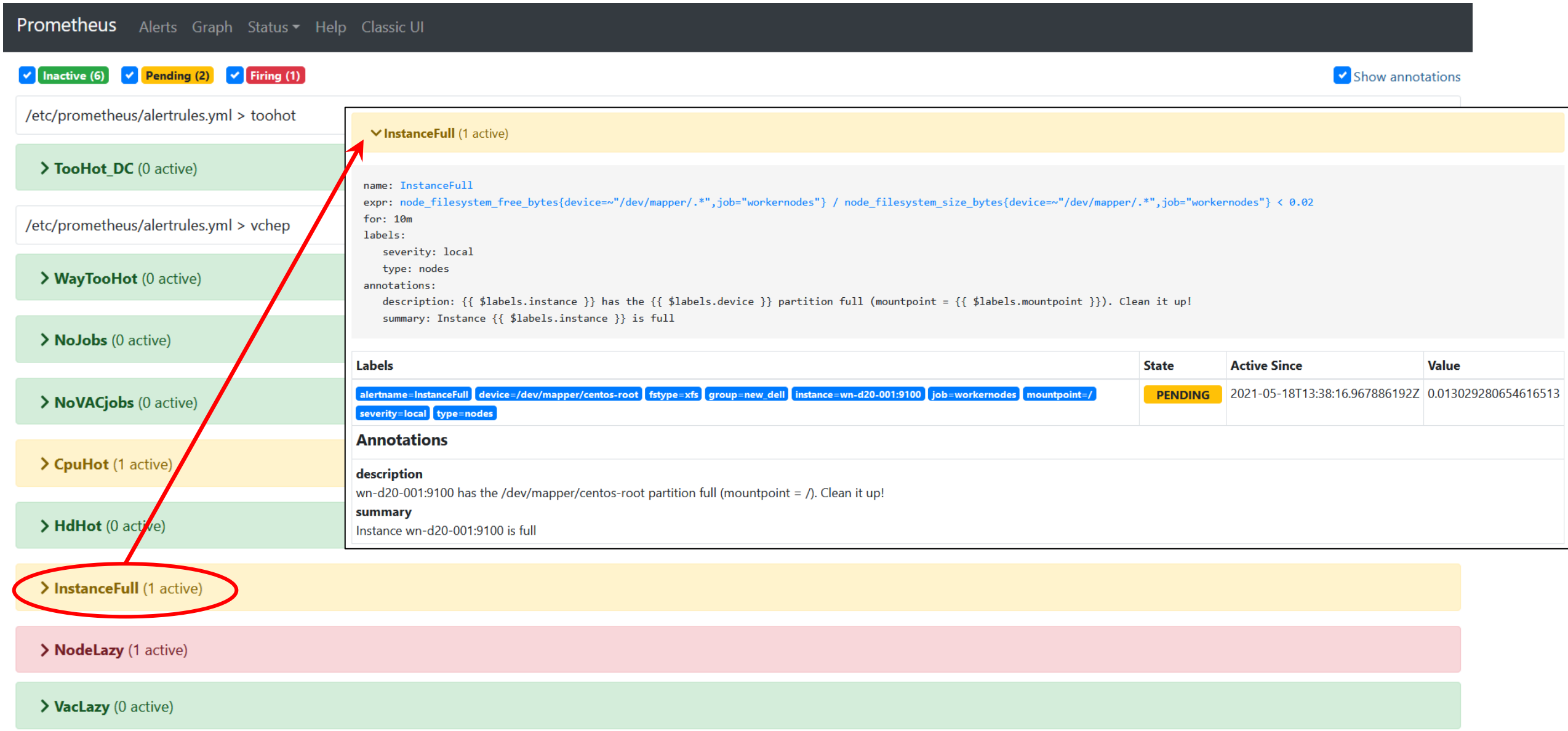
Example Alert



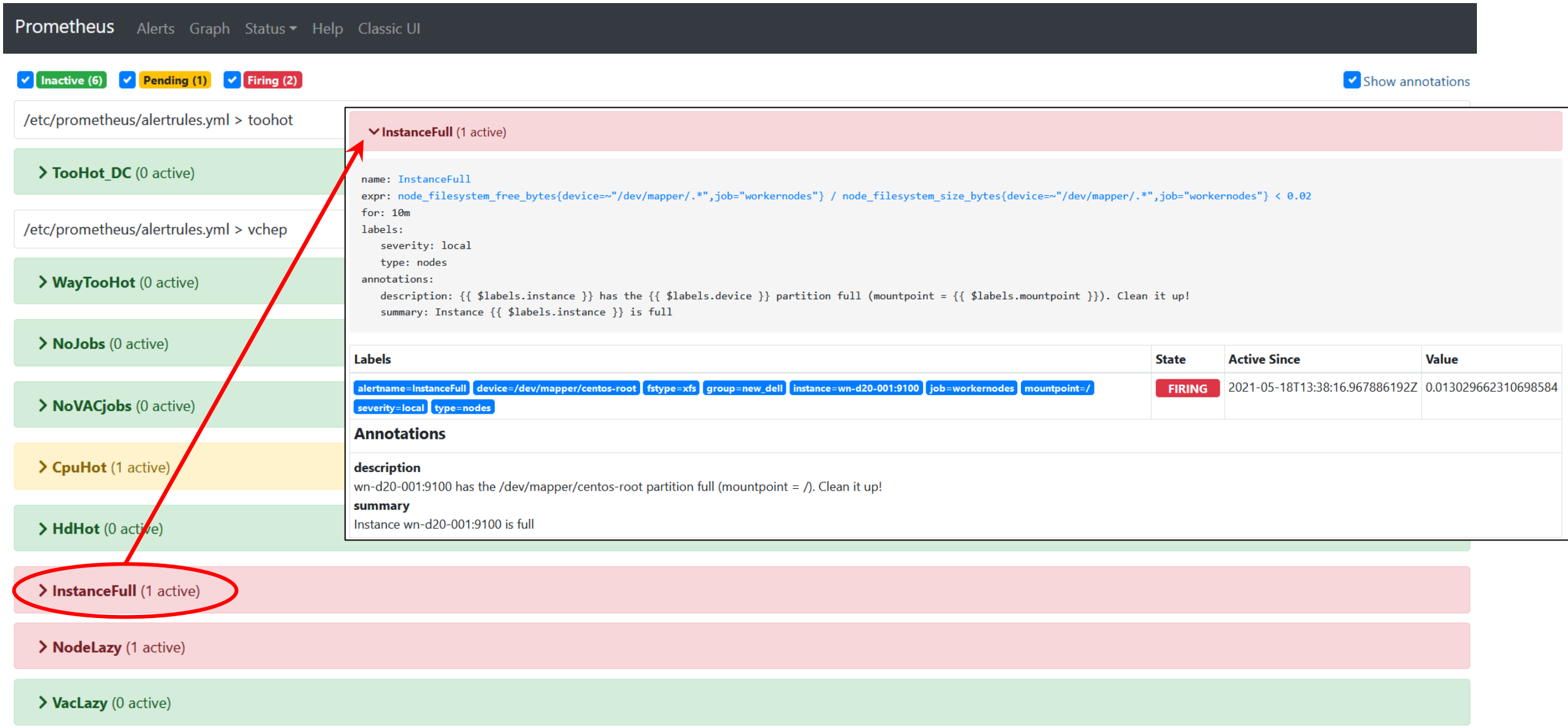
One of the disk has filled-up and an ‘Alert’ message is generated (after 10 min):



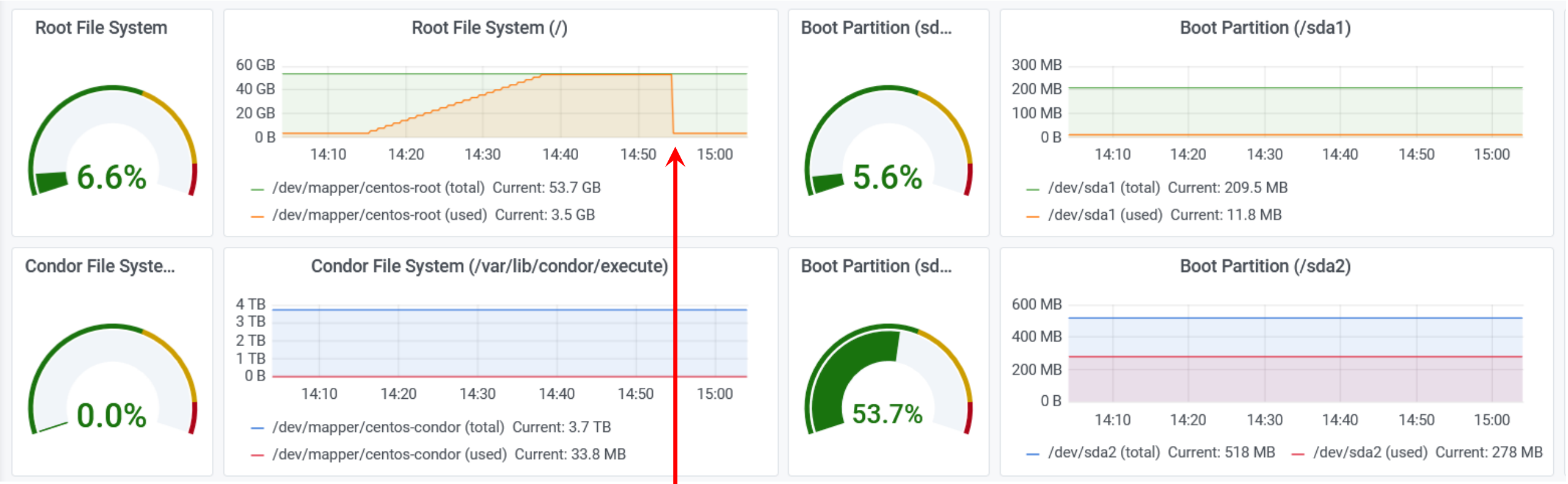
Alert Pending



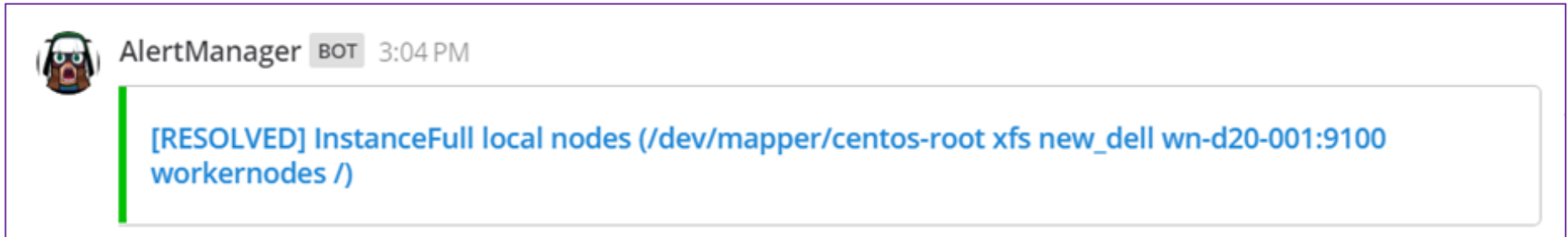
Alert Firing



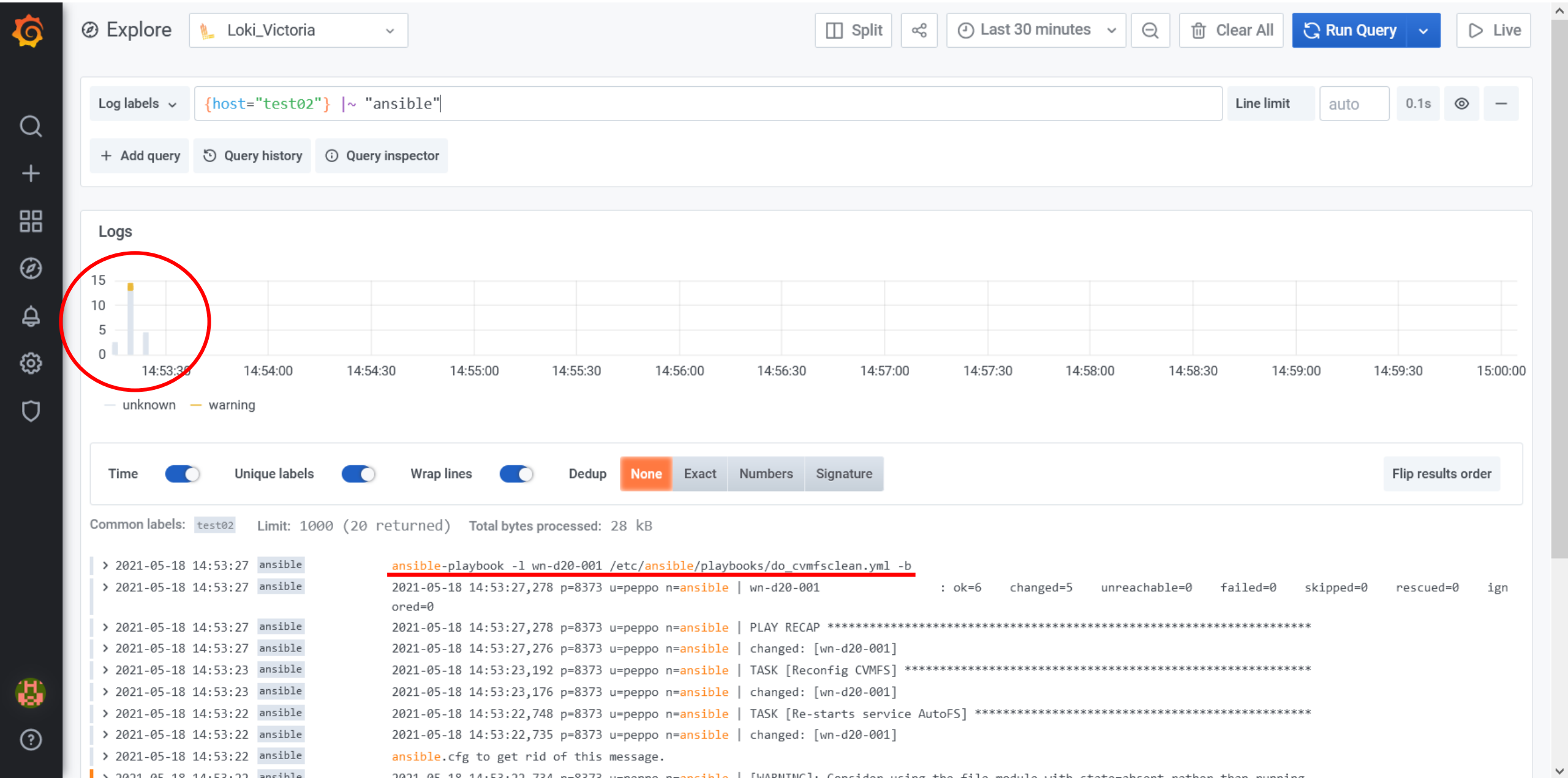
Alert Solution



The disk has been cleaned and a ‘Resolved Alert’ message is sent (after 10 min):



Ansible Logs



Conclusions & Outlook

This system is relatively new (first working prototype in January 2021) and it is still under critical evaluation*:

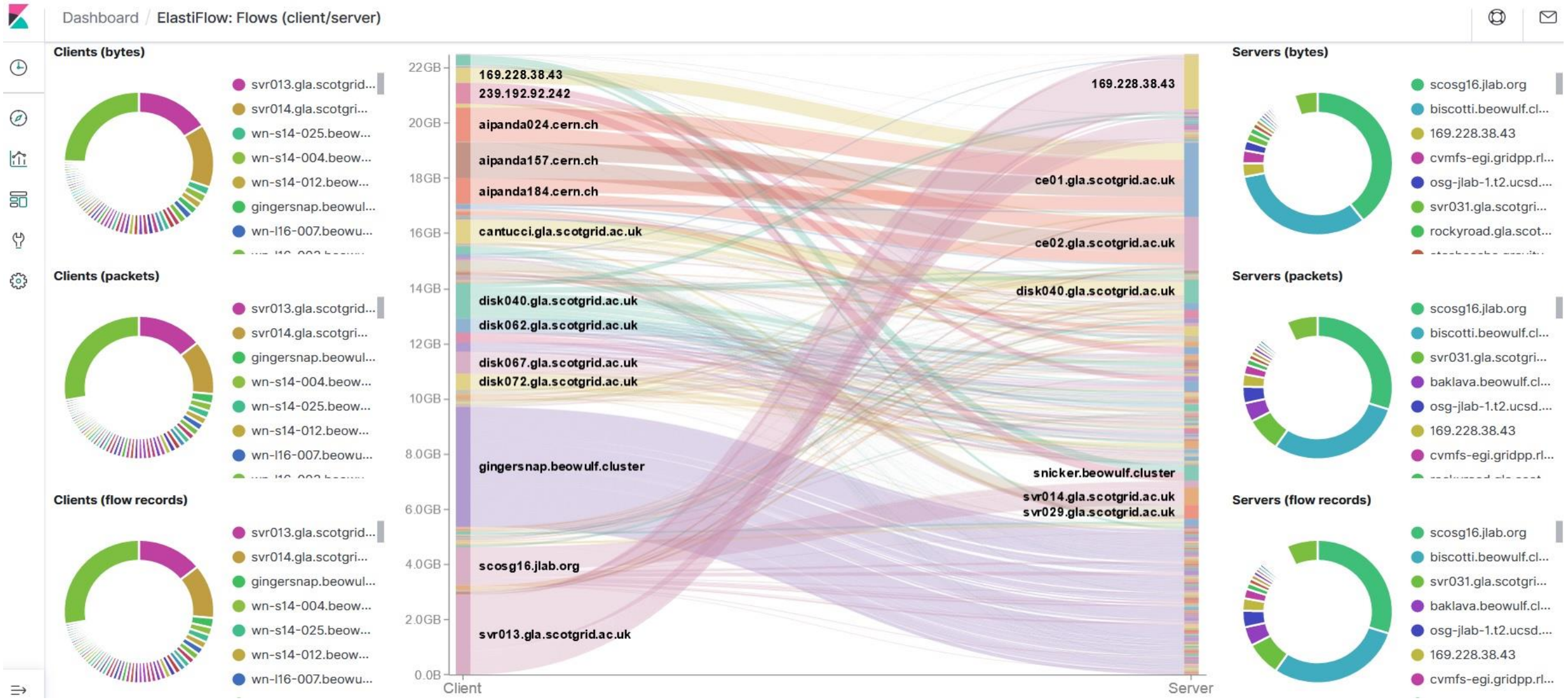
- We have an alerting system that can send emails/SMS in case of emergency, and few simple recovery actions that run automatically (clean HD, reboot, ...)
- Next, we want to identify more complex alerts based on combinations of metrics and define multi-step recovery actions, such as re-provision a node
- Also, it would be nice to extend the monitoring actions to network stats
- Eventually, we would like to experiment with ML Anomaly Detection systems
- We plan to engage other GridPP sites to come up with the best practices for monitoring and automated fault recovery, as well as get involved with the CERN Operational Intelligence group

* It is well known that giving full control to the machines without proper evaluation will eventually lead to *Skynet*

Thanks for listening.

ElasticFlow

We did some experimenting with **ElasticSearch** modules to analyze network flows from



… but this is another story.

Example Logging

As an example, these are the bits in the **PromTail** configuration (`promtail.yml`) that export general and HTCondor logs:

```yaml
# PromTail Configuration defines the rules for scraping local logs

server:
  http_listen_port: 9080
  grpc_listen_port: 0

# Positions
positions:
  filename: /tmp/positions.yaml

# Loki Server URL
clients:
  - url: http://{{ central_log_server }}:3100/loki/api/v1/push

scrape_configs:
  ## Common Logs
  - job_name: messages        # log messages (/var/log/messages)
    static_configs:
      - targets:
          - "{{ inventory_hostname }}:9080"
        labels:
          job: messages
          host: "{{ inventory_hostname }}"
          __path__: /var/log/messages
  # ...
```

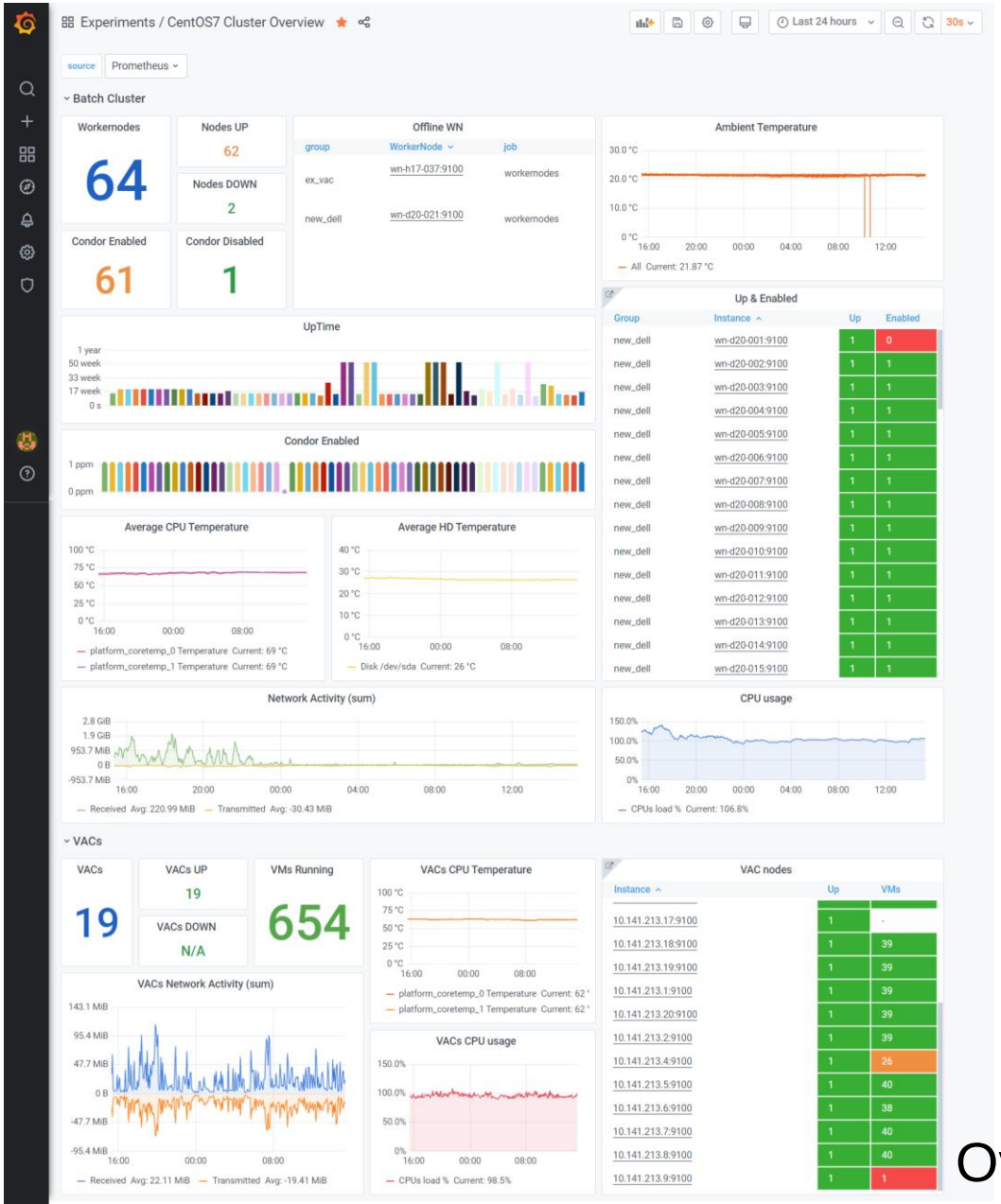
General Logs (system messages)

```yaml
  ## Service Logs
  - job_name: condor
    static_configs:
      - targets:
          - "{{ inventory_hostname }}:9080"
        labels:
          job: condor
          host: "{{ inventory_hostname }}"
          __path__: /var/log/condor/*og
```

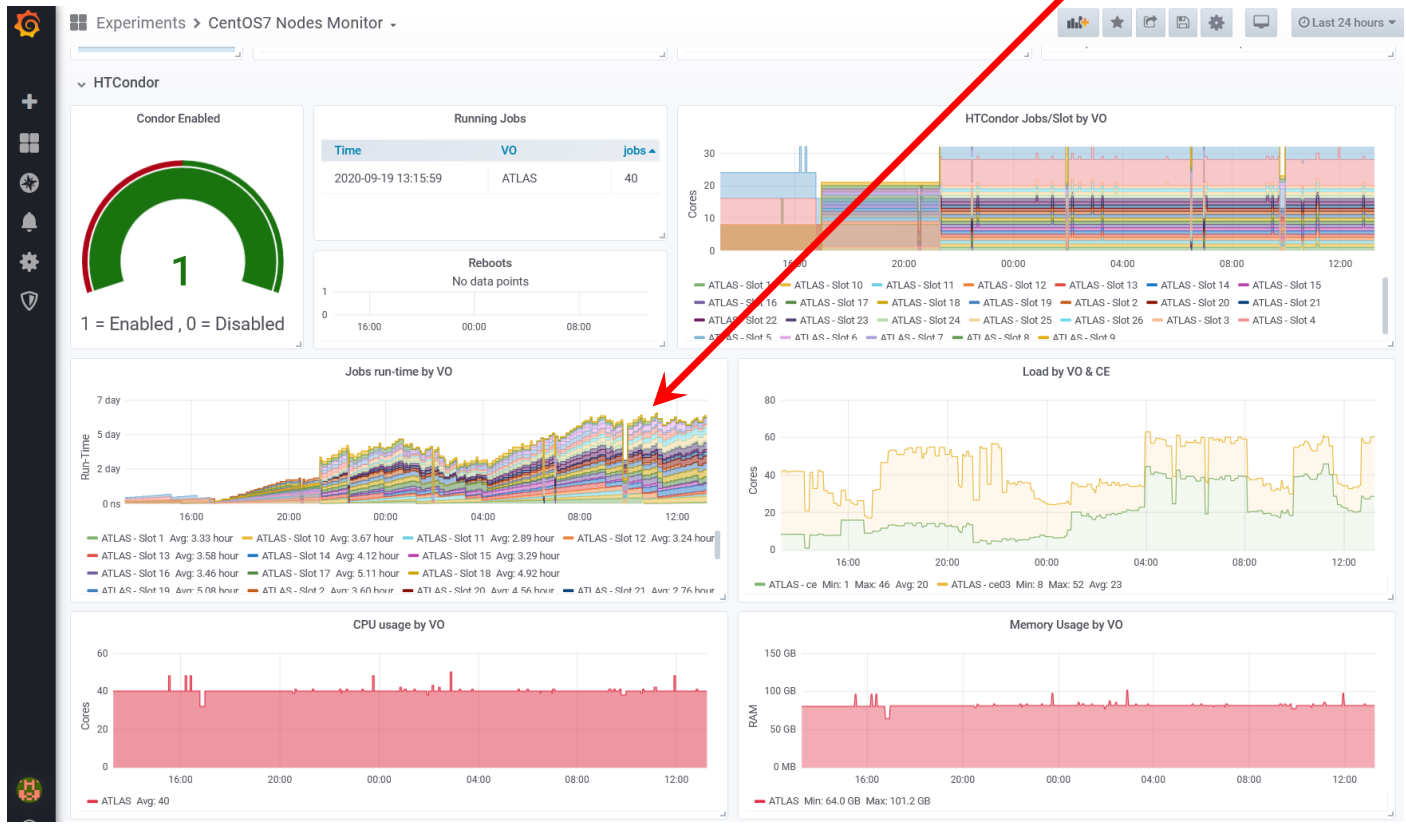
HTCondor logs:

`/var/log/condor/`

ClusterView Dashboards

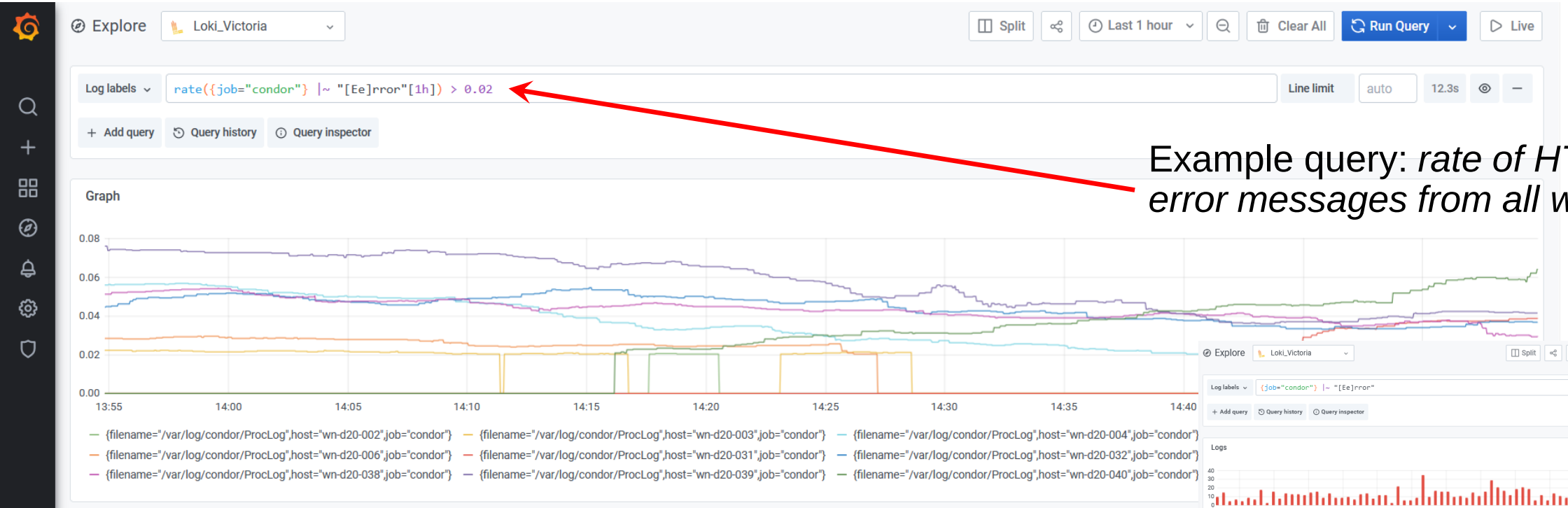


Running jobs per VO



Over 50 graphs about CPU, Memory, Disks, Networking ...

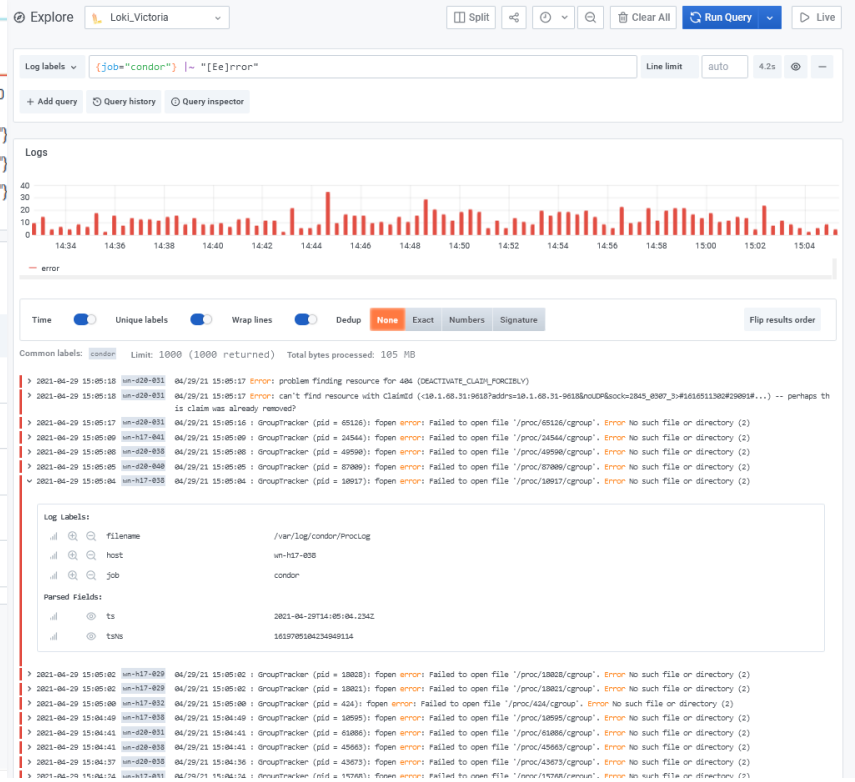
Log Queries



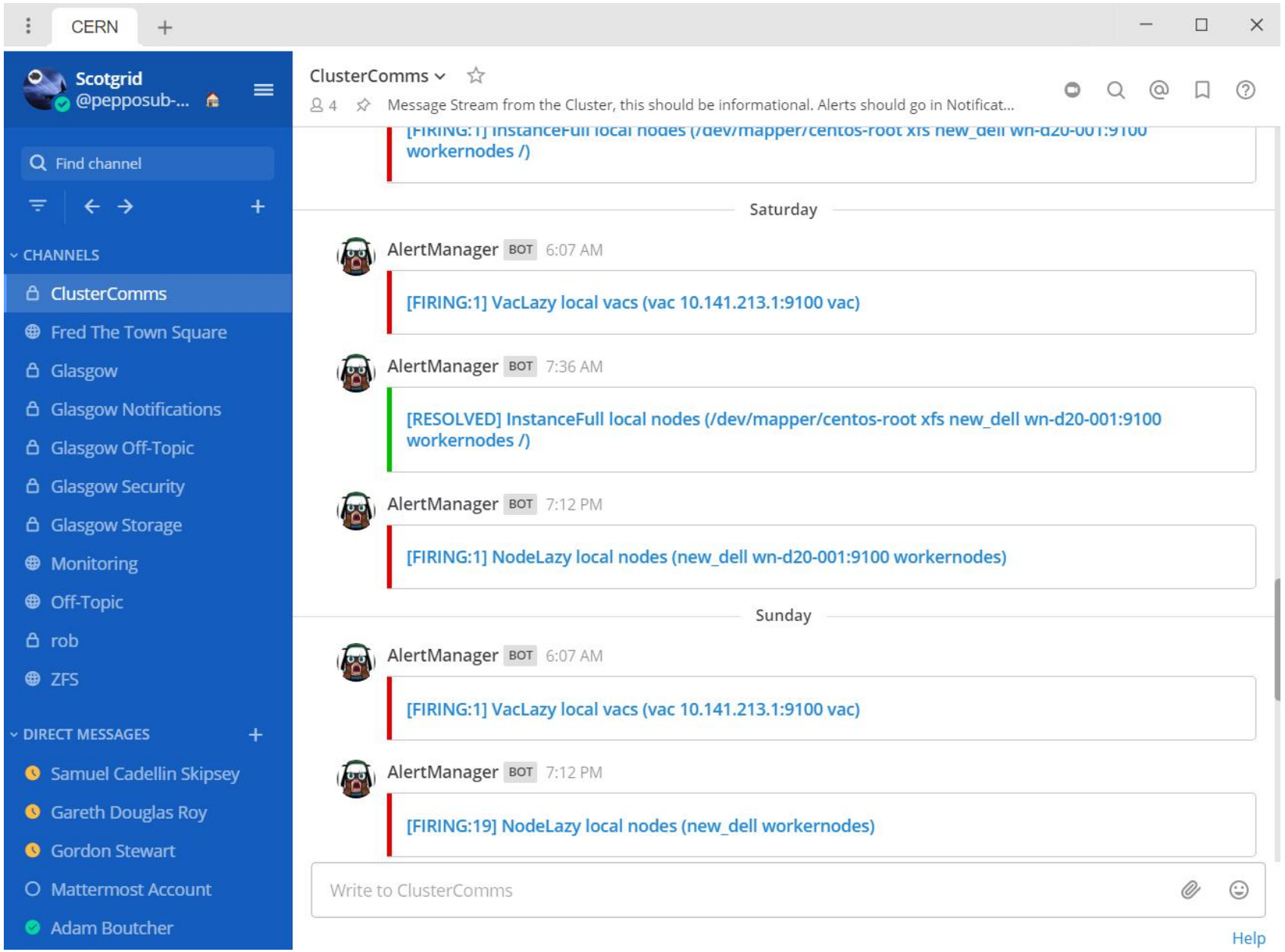
Example query: *rate of HTCondor error messages from all workernodes*

Table

Time	filename	host	job
2021-04-29 14:54:50	/var/log/condor/ProcLog	wn-d20-031	condor
2021-04-29 14:54:50	/var/log/condor/ProcLog	wn-d20-032	condor
2021-04-29 14:54:50	/var/log/condor/ProcLog	wn-d20-038	condor
2021-04-29 14:54:50	/var/log/condor/ProcLog	wn-d20-039	condor
2021-04-29 14:54:50	/var/log/condor/ProcLog	wn-d20-040	condor

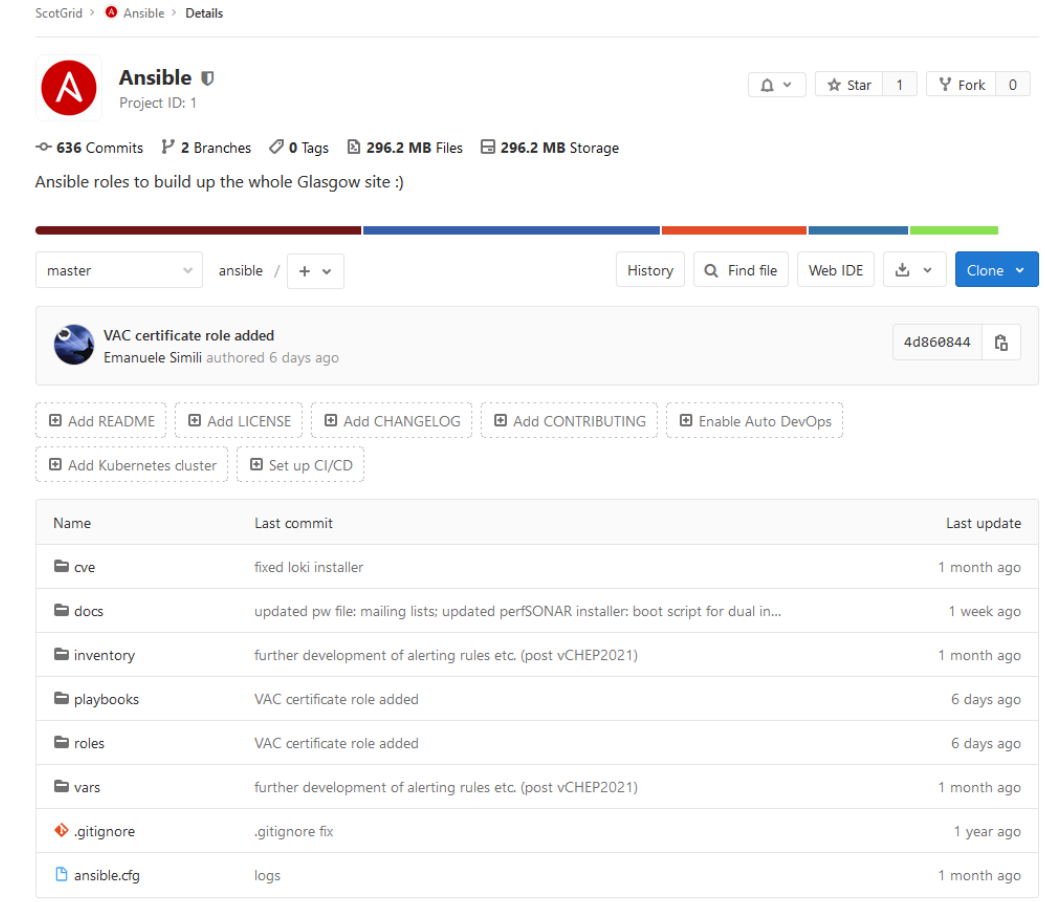


MatterMost Channel

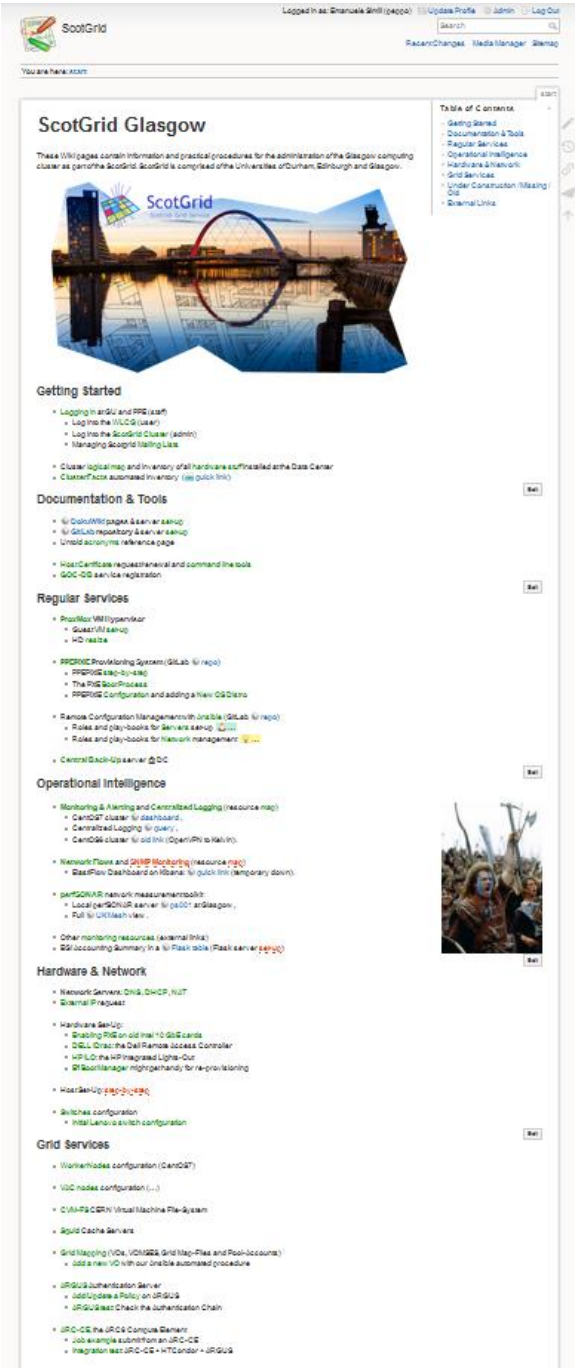


Git & Wiki

Ansible repository on our internal Git:
~70 Ansible roles
~50 Ansible play-books (incl. maintenance)



Internal wiki with detailed procedures and code snippets for cluster built:
~50 pages



Glasgow Tier-2 Cluster

Submission system consists of a few hundred (*) nodes managed by **NorduGrid ARC**, **ARGUS** and **HTCondor**.
It provides computing power to LHC experiments (ATLAS, CMS) and non-LHC experiments (LIGO, CLAS12, …).

- We have 4 ARC-CE server for job submission: **ce01**, **ce02**, ...
- We have ~300 * physical workernodes (~80 connected): **wn00x**
- W have 1 ARGUS for user authentication and authorization
- We have 1 HTCondor manager which collects scheduled jobs
- We have 2 Squids acting as a cache for CVMFS
- We have 2 NAT servers for communicating with the outside world
- We have ~20 VAC nodes *: **vac00x**
- We have a local Storage Element (SE) **
- And, of course, everything is within our private VPN (WireGuard)

* Beside the conventional Grid submission system, the Glasgow site also runs a smaller number of VAC machines to provide an Infrastructure-as-a-Service (IaaS) cloud for various other experiments, including LHCb.

** The Storage Element is composed of three sets of physical machines running Ceph, including about 20 dedicated storage machines or disks (**cephd0x**), six physical caches (**cephc0x**) and three entry points or servers (**cephs0x**).

UKI-SCOTGRID-GLASGOW

- Part of the GridPP collaboration providing resources to the Worldwide LHC Compute Grid (WLCG).
- One of 19 institutions comprising 4 distributed Tier-2 sites (SCOTGRID, NORTHGRID, SOUTHGRID and LT2).
- Part of the SCOTGRID Distributed Tier-2 including Glasgow, Edinburgh and Durham Universities.