Kubernetes & Rucio

Thomas Beermann
on behalf of the Rucio team

RUCIO
SCIENTIFIC DATA MANAGEMENT

Introduction

- We started to have a look into Kubernetes to help automate the deployment and scaling of our ATLAS production services as well as simplify the setup of new instances.
- The current ATLAS deployment uses separate VMs deployed on the CERN-IT provided Openstack infrastructure.
- The server and daemon services are split by integration and production:
    - 15 / 2 production / integration server VMs.
    - 25 / 7 production / integration daemon VMs.
    - 3 haproxy load balancers.
    - 2 / 1 production / integration webui servers + a couple of VMs for misc services, e.g., nagios.
- Deployment is fully managed with Puppet.

Limitation of current deployment

- It is running stable and we have a lot of experience with the current model but:
  - Regular problems with Python dependencies that are overwritten by automatic package upgrade on the VMs breaking our deployment.
  - The puppet deployment grew over time and became quite complicated.
  - Adapting the deployment to add or remove new daemons to adapt to different workloads requires manual intervention and is rather slow.
  - Setup of a new deployment is complicated and needs a lot of support for the initial installation.
  - The VM resources for the ATLAS deployment are highly underutilized because of redundancies and the static deployment model with Puppet.
  - Hunting down problems can be tedious sometimes due to the distributed nature of the deployment.
- Could benefit a lot of a more dynamic Kubernetes deployment.

What’s Kubernetes?

- Kubernetes, or k8s, is an open-source system for automating deployment, scaling, and management of containerized applications.
- Originally designed by Google but now managed by the Linux Foundation.
- A lot of cloud providers offer a Kubernetes-based platform.
- At CERN it is provided with OpenShift on top of Openstack.
- But it is also possible to manually deploy a cluster.
- It provides:
  - Possibility to easily deploy containers across multiple hosts.
  - Placement of containers to maximize resources utilization.
  - Centralized ways to control and update deployments and applications.
  - Tools to automatically scale the deployment on the fly.
  - Health-checks and self-heal to automatically restart / replace problematic services.

Why Kubernetes for Rucio

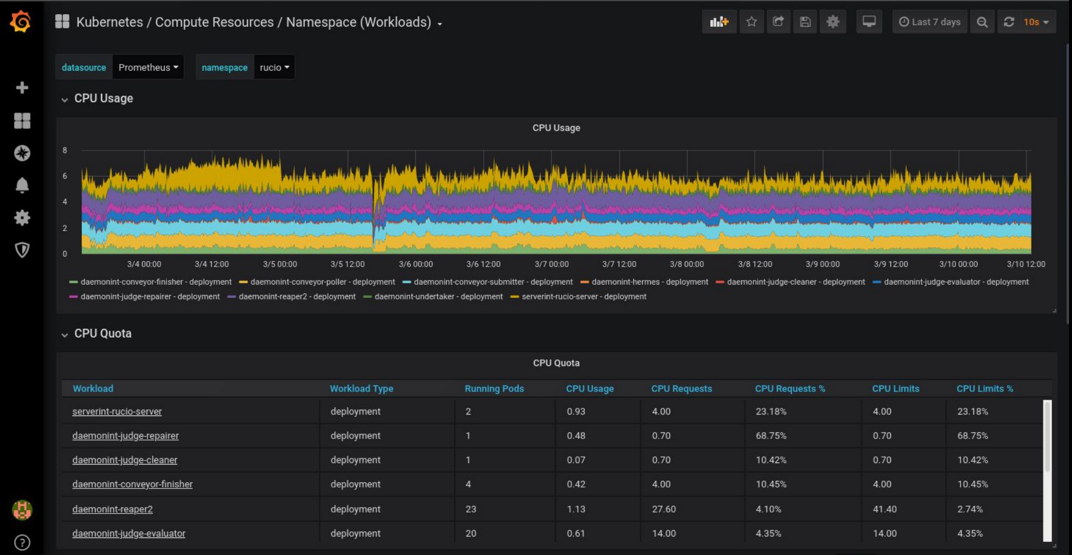
- Containers provide an isolated and minimal environment with only the necessary dependencies needed for the application.
- Initial deployment of new services becomes really easy and is quick thanks to Helm charts.
- Changes in the deployment and software upgrades are quickly propagated through the system.
- Auto-scaling can help in case of spikes in the workload and to better utilize the available resources / better energy efficiency.
- Centralized monitoring and logging can make it easier to find problems.

Deployment with Helm and Flux

- The Rucio server and daemon services are fully packaged with Helm.
- Available in our own repo on Github but we will also make it available on Helm Hub.
- Set up of a new Rucio instance is now as simple as adapting a few configuration parameters and installing the Helm chart.
- Recently started to use Flux for our ATLAS integration deployment:
  - Since we had the Helm charts already available it is rather easy to set up.
  - The Helm values are managed in a github / gitlab repository.
  - An agent on the cluster regularly checks for updates in the repo and automatically deploys them.
  - Changing the deployment is then done by simple git commits, similar to puppet but much quicker.
  - Upgrading to a new version or adding new daemons / servers only takes a few minutes.
  - Adds accountability which is important for us since there can be multiple people trying to change the deployment.
  - Could bridge the gap for of our ops people not having too much experience with Kubernetes, yet.

Monitoring

- The CERN-IT deployed clusters come with a built-in Prometheus time-series database.
- It automatically collect several metrics from the cluster including CPU, memory and network usage.
- It can be extended to also include metrics from Rucio, e.g., waiting transfers / deletion queue, server response times.
- Grafana dashboards are available to monitor the whole cluster, separate workloads, e.g., cpu / memory usage of all judge daemons, and the Rucio specific metrics.
- All of those metrics can be exposed and used for auto-scaling.


Kubernetes / Compute Resources / Namespace (Workloads)

datasource Prometheus · namespace rucio

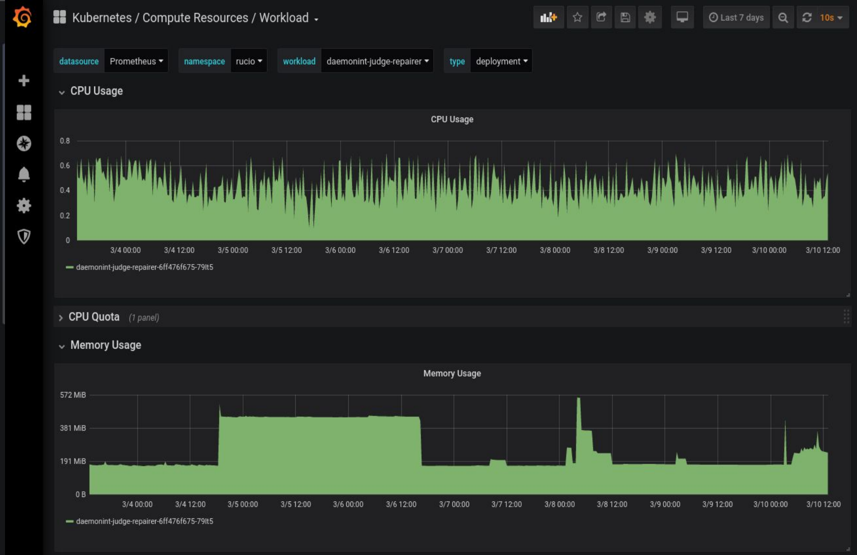
Last 7 days · 10s

CPU Usage

CPU Usage
daemonint-conveyor-finisher - deployment
daemonint-conveyor-poller - deployment
daemonint-conveyor-submitter - deployment
daemonint-hermes - deployment
daemonint-judge-cleaner - deployment
daemonint-judge-evaluator - deployment
daemonint-judge-repairer - deployment
daemonint-reaper2 - deployment
daemonint-undertaker - deployment
serverint-rucio-server - deployment

CPU Quota

Workload	Workload Type	Running Pods	CPU Usage	CPU Requests	CPU Requests %	CPU Limits	CPU Limits %
serverint-rucio-server	deployment	2	0.93	4.00	23.18%	4.00	23.18%
daemonint-judge-repairer	deployment	1	0.48	0.70	68.75%	0.70	68.75%
daemonint-judge-cleaner	deployment	1	0.07	0.70	10.42%	0.70	10.42%
daemonint-conveyor-finisher	deployment	4	0.42	4.00	10.45%	4.00	10.45%
daemonint-reaper2	deployment	23	1.13	27.60	4.10%	41.40	2.74%
daemonint-judge-evaluator	deployment	20	0.61	14.00	4.35%	14.00	4.35%


Kubernetes / Compute Resources / Workload

datasource Prometheus · namespace rucio · workload daemonint-judge-repairer · type deployment

Last 7 days · 10s

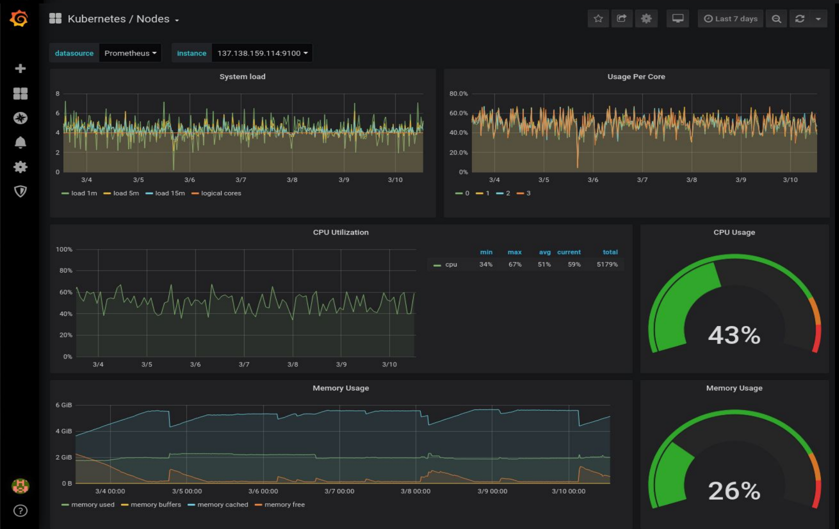
CPU Usage

CPU Usage
daemonint-judge-repairer-6ff476f675-79lt5

CPU Quota (1 panel)

Memory Usage

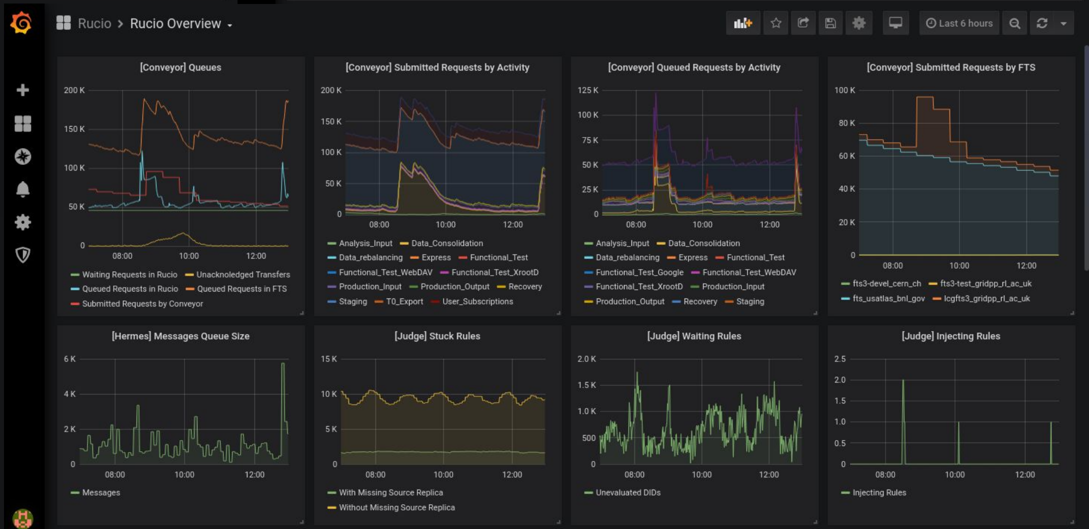
Memory Usage
572 MiB
381 MiB
191 MiB
0 B
daemonint-judge-repairer-6ff476f675-79lt5


Kubernetes / Nodes

datasource Prometheus · instance 137.138.159.114:9100

Last 7 days

	min	max	avg	current	total
cpu	34%	67%	51%	59%	5179%


Rucio > Rucio Overview

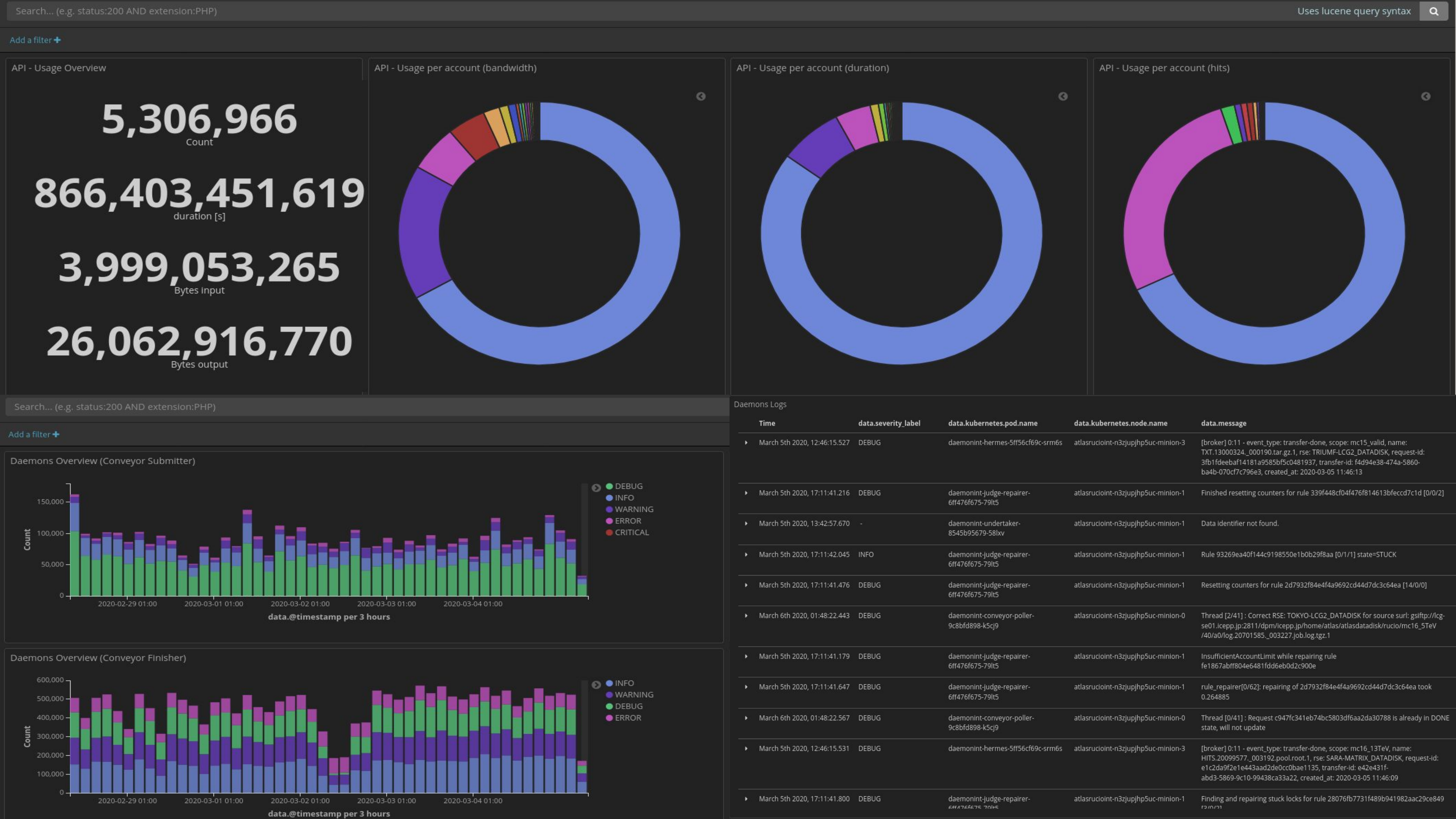
Last 6 hours

Auto-scaling

- A big advantage of Kubernetes is the possibility to automatically scale the cluster based on predefined metrics from Prometheus.
- There are are two types of auto-scaling:
  - Horizontal Pod Autoscaling: increase / decrease the number of pods based on given metrics.
  - Cluster Autoscaling: add / remove nodes to the cluster when pods cannot be scheduled anymore or pods could be rescheduled to other nodes.
- In both cases a minimum / maximum number of pods / nodes are specified.
- For Rucio this could be used to automatically scale the number of server pods when the average response time is going down, increase the number of reapers if a deletion backlogs builds up or deploy more conveyors to submit to FTS.
- Basic functionality has been successfully tested but it still has to be tested in a larger deployment.

Logging

- Kubernetes makes it easy to quickly setup centralized logging.
- There are different tools available like Filebeat or Fluentd that can be deployed on the cluster and automatically collect and process the logs for all pods.
- For ATLAS we are currently using Filebeat and Logstash to send all logs to a central logs monitoring infrastructure provided provided by CERN-IT.
- But any Elasticsearch instance could be used.
- Kibana can be used to quickly search for specific logs.
- Dashboard can be built for different purposes, e.g.:
    - Server API monitoring: showing detailed information about the API usage including hits per endpoint, per account, error codes, etc.
    - Daemon activity monitoring: showing an overview of log messages sent from the different daemons to spot potential problems.



5,306,966
Count

866,403,451,619
duration [s]

3,999,053,265
Bytes input

26,062,916,770
Bytes output

Daemons Logs

Time	data.severity_label	data.kubernetes.pod.name	data.kubernetes.node.name	data.message
March 5th 2020, 12:46:15.527	DEBUG	daemonint-hermes-5ff56cf69c-srm6s	atlasrucioint-n3zjupjhp5uc-minion-3	[broker] 0:11 - event_type: transfer-done, scope: mc15_valid, name: TXT.13000324._000190.tar.gz.1, rse: TRIUMF-LCG2_DATADISK, request-id: 3fb1fdeebaf14181a9585bf5c0481937, transfer-id: f4d94e38-474a-5860-ba4b-070cf7c796e3, created_at: 2020-03-05 11:46:13
March 5th 2020, 17:11:41.216	DEBUG	daemonint-judge-repairer-6ff476f675-79lt5	atlasrucioint-n3zjupjhp5uc-minion-1	Finished resetting counters for rule 339f448cf04f476f814613bfeccd7c1d [0/0/2]
March 5th 2020, 13:42:57.670	-	daemonint-undertaker-8545b95679-58lxv	atlasrucioint-n3zjupjhp5uc-minion-1	Data identifier not found.
March 5th 2020, 17:11:42.045	INFO	daemonint-judge-repairer-6ff476f675-79lt5	atlasrucioint-n3zjupjhp5uc-minion-1	Rule 93269ea40f144c9198550e1b0b29f8aa [0/1/1] state=STUCK
March 5th 2020, 17:11:41.476	DEBUG	daemonint-judge-repairer-6ff476f675-79lt5	atlasrucioint-n3zjupjhp5uc-minion-1	Resetting counters for rule 2d7932f84e4f4a9692cd44d7dc3c64ea [14/0/0]
March 6th 2020, 01:48:22.443	DEBUG	daemonint-conveyor-poller-9c8bfd898-k5cj9	atlasrucioint-n3zjupjhp5uc-minion-0	Thread [2/41] : Correct RSE: TOKYO-LCG2_DATADISK for source surl: gsiftp://lcg-se01.icepp.jp:2811/dpm/icepp.jp/home/atlas/atlasdatadisk/rucio/mc16_5TeV/40/a0/log.20701585._003227.job.log.tgz.1
March 5th 2020, 17:11:41.179	DEBUG	daemonint-judge-repairer-6ff476f675-79lt5	atlasrucioint-n3zjupjhp5uc-minion-1	InsufficientAccountLimit while repairing rule fe1867abff804e6481fdd6eb0d2c900e
March 5th 2020, 17:11:41.647	DEBUG	daemonint-judge-repairer-6ff476f675-79lt5	atlasrucioint-n3zjupjhp5uc-minion-1	rule_repairer[0/62]: repairing of 2d7932f84e4f4a9692cd44d7dc3c64ea took 0.264885
March 6th 2020, 01:48:22.567	DEBUG	daemonint-conveyor-poller-9c8bfd898-k5cj9	atlasrucioint-n3zjupjhp5uc-minion-0	Thread [0/41] : Request c947fc341eb74bc5803df6aa2da30788 is already in DONE state, will not update
March 5th 2020, 12:46:15.531	DEBUG	daemonint-hermes-5ff56cf69c-srm6s	atlasrucioint-n3zjupjhp5uc-minion-3	[broker] 0:11 - event_type: transfer-done, scope: mc16_13TeV, name: HITS.20099577._003192.pool.root.1, rse: SARA-MATRIX_DATADISK, request-id: e1c2da9f2e1e443aad2de0cc0bae1135, transfer-id: e42e431f-abd3-5869-9c10-99438ca33a22, created_at: 2020-03-05 11:46:09
March 5th 2020, 17:11:41.800	DEBUG	daemonint-judge-repairer-	atlasrucioint-n3zjupjhp5uc-minion-1	Finding and repairing stuck locks for rule 28076fb7731f489b941982aac29ce849

Debugging

- Compared to our current deployment model debugging is a bit different and we still need to gain more experience with it.
- There are no tools installed inside the container to debug and fix bugs.
- Furthermore, a restart of the server or daemon inside the container would also restart the pod and therefore reverting the change.
- Ephemeral containers have been recently added to Kubernetes and could be useful at least for debugging:
    - Can be attached to a running container on-the-fly.
    - Can be equipped with editors, debugging tools, etc. to help track down and fix the problem.
- Currently evaluation possibilities to run the application not as PID 1 so that they can be restarted.

Issues in the ATLAS deployment

- We have encountered a couple issues so far, mostly caused by the rather new infrastructure at CERN. But they still need to be addressed before we can move to production:
    - In the early days we had problem with the networking inside the cluster which needed to service restarts on the node but more recently with the newer clusters this did not show up anymore.
    - Some of our daemon pods cause the minion nodes to run out of memory which makes them unavailable in the cluster. Can be avoided by setting resources on the deployment.
    - Problems with slow network in the cluster due to the currently available network driver at CERN (flannel). But work is ongoing by CERN IT to provide a faster network with calico.
    - Our average server response time is much slower in Kubernetes and that is still something that needs to be addressed. Probably also caused by the slow network.

Summary

- We are on a good way to move all our Rucio deployments for ATLAS into Kubernetes.
- It has the potential to help and automate some parts of our daily operations for ATLAS.
- Can make new installations much easier.
- We have smaller instances of Rucio running for a long time now without major problems, e.g., DOMA Third-Party-Copy tests.
- Integration services are running both for ATLAS and CMS on Kubernetes.
- But we still need to gain more experience for larger deployments.
- We will make a tutorial available with a full Rucio instance. It is using minikube and can run directly on your own machine. Can be useful to gain experience.

More information

Website		http://rucio.cern.ch
Documentation		https://rucio.readthedocs.io
Repository		https://github.com/rucio/
Images		https://hub.docker.com/r/rucio/
Online support		https://rucio.slack.com/messages/#support/
Developer contact		rucio-dev@cern.ch
Publications	Rucio – Scientific data management [illegible]	https://rucio.cern.ch/publications.html
Twitter		https://twitter.com/RucioData