How the CKI team keeps its service running

Cyborg Infra Workshop 2021: Day 1

Iñaki Malerba

Michael Hofmann

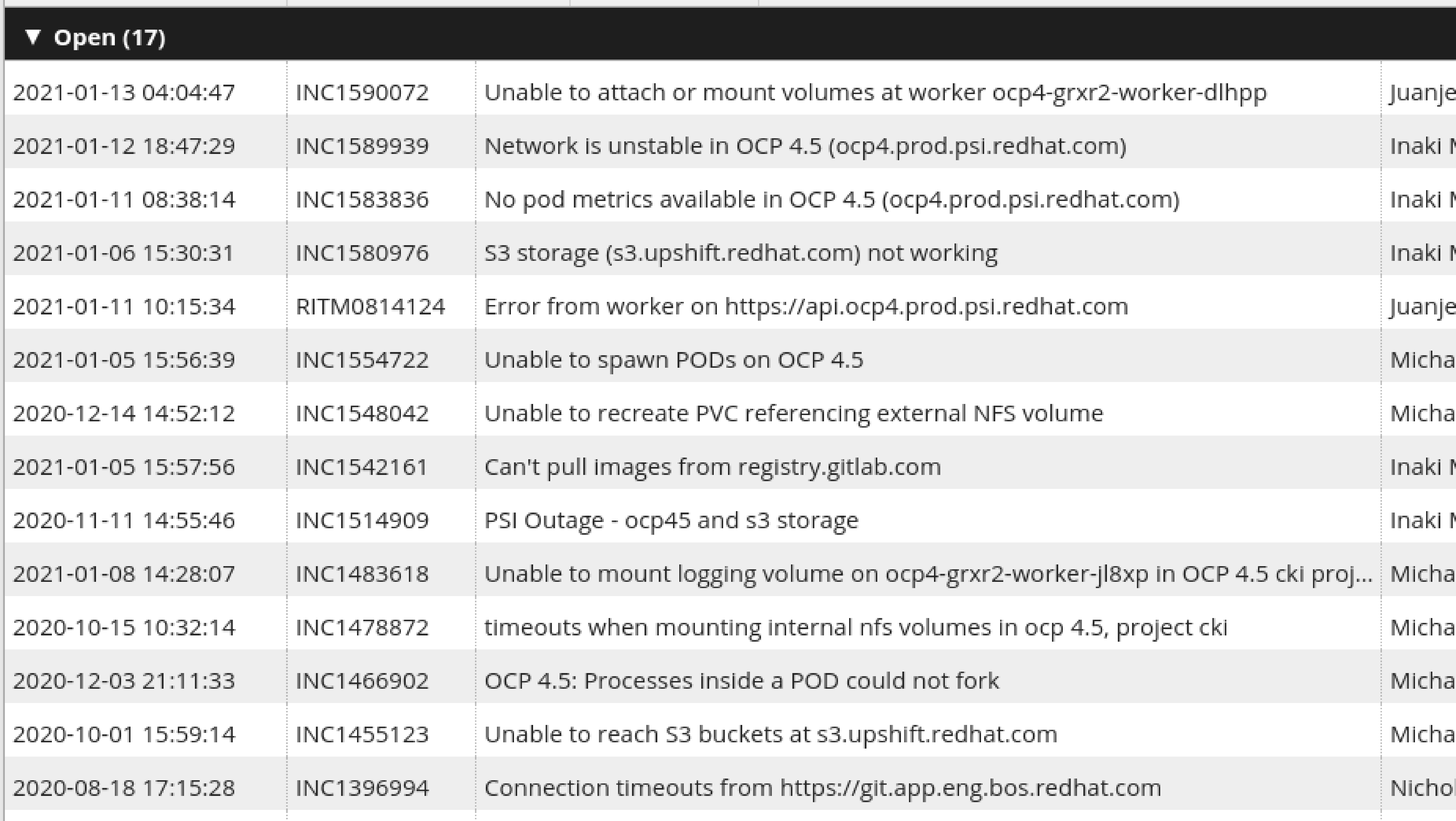
Problem statement

Avoiding grumpy kernel developers

- For each commit under test, run a build+test pipeline to completion
- Ideally that means:
  - detecting a commit
  - triggering a pipeline
  - reporting results
- How hard can it be…

Name or Service not known

https://www.pexels.com/photo/grey-kitten-on-silver-paper-bag-141496/



▼ Open (17)

2021-01-13 04:04:47	INC1590072	Unable to attach or mount volumes at worker ocp4-grxr2-worker-dlhpp	Juanje
2021-01-12 18:47:29	INC1589939	Network is unstable in OCP 4.5 (ocp4.prod.psi.redhat.com)	Inaki M
2021-01-11 08:38:14	INC1583836	No pod metrics available in OCP 4.5 (ocp4.prod.psi.redhat.com)	Inaki M
2021-01-06 15:30:31	INC1580976	S3 storage (s3.upshift.redhat.com) not working	Inaki M
2021-01-11 10:15:34	RITM0814124	Error from worker on https://api.ocp4.prod.psi.redhat.com	Juanje
2021-01-05 15:56:39	INC1554722	Unable to spawn PODs on OCP 4.5	Micha
2020-12-14 14:52:12	INC1548042	Unable to recreate PVC referencing external NFS volume	Micha
2021-01-05 15:57:56	INC1542161	Can't pull images from registry.gitlab.com	Inaki M
2020-11-11 14:55:46	INC1514909	PSI Outage - ocp45 and s3 storage	Inaki M
2021-01-08 14:28:07	INC1483618	Unable to mount logging volume on ocp4-grxr2-worker-jl8xp in OCP 4.5 cki proj...	Micha
2020-10-15 10:32:14	INC1478872	timeouts when mounting internal nfs volumes in ocp 4.5, project cki	Micha
2020-12-03 21:11:33	INC1466902	OCP 4.5: Processes inside a POD could not fork	Micha
2020-10-01 15:59:14	INC1455123	Unable to reach S3 buckets at s3.upshift.redhat.com	Micha
2020-08-18 17:15:28	INC1396994	Connection timeouts from https://git.app.eng.bos.redhat.com	Nichol

General idea: reliable service on unreliable infrastructure

Murphy and It's Always DNS

- Lemmas:
  - Any component/dependency that can fail will fail
  - Some will fail more than others
  - Nearly all failures can be retried successfully
  - But we also have to detect the other ones
- So failures need to be...
  - Detected: logging, monitoring, alerting
  - Prevented: redundancy, fewer dependencies
  - Recovered: retries at all levels, fallbacks

Detection

Detection

Keeping track of many, many pieces

- Lots of moving pieces
  - Long standing pods
  - Cron jobs
  - Different clouds, different clusters
  - Services scaling up/down
- Data in different formats
  - Logs
  - Data points
  - Errors

Logs collection

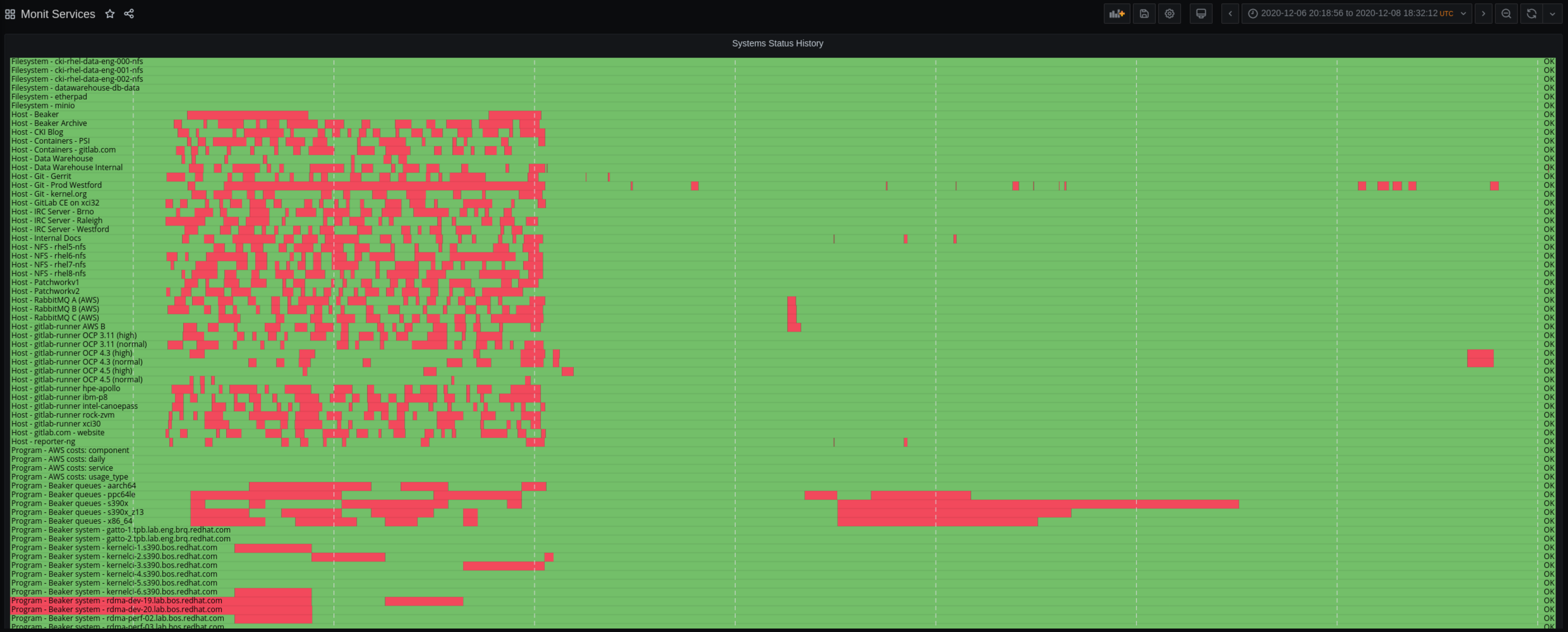
> /dev/null

- Standardized logger names and levels
  - Easier to read and configure
- Putting all the logs on a common place
  - Shared NFS between OCP pods
  - Human friendly, easily grepable
- Grafana Loki stack for processing
  - *'Like Prometheus, but for logs!'*
  - Indexed and easy retention policies

High level monitoring

Just assume no one monitors their services

- Keep track of 3rd party resources that we depend on
- Monit as a simple solution for monitoring
  - Hosts uptime
  - NFS file systems uptime and size
  - Beaker hosts queues
  - S3 bucket sizes
  - RabbitMQ messages and queues
- Store instant statuses and **record** downtimes


Monit Services
2020-12-06 20:18:56 to 2020-12-08 18:32:12 UTC
Systems Status History
Filesystem - cki-rhel-data-eng-000-nfs
Filesystem - cki-rhel-data-eng-001-nfs
Filesystem - cki-rhel-data-eng-002-nfs
Filesystem - datawarehouse-db-data
Filesystem - etherpad
Filesystem - minio
Host - Beaker
Host - Beaker Archive
Host - CKI Blog
Host - Containers - PSI
Host - Containers - gitlab.com
Host - Data Warehouse
Host - Data Warehouse Internal
Host - Git - Gerrit
Host - Git - Prod Westford
Host - Git - kernel.org
Host - GitLab CE on xci32
Host - IRC Server - Brno
Host - IRC Server - Raleigh
Host - IRC Server - Westford
Host - Internal Docs
Host - NFS - rhel5-nfs
Host - NFS - rhel6-nfs
Host - NFS - rhel7-nfs
Host - NFS - rhel8-nfs
Host - Patchworkv1
Host - Patchworkv2
Host - RabbitMQ A (AWS)
Host - RabbitMQ B (AWS)
Host - RabbitMQ C (AWS)
Host - gitlab-runner AWS B
Host - gitlab-runner OCP 3.11 (high)
Host - gitlab-runner OCP 3.11 (normal)
Host - gitlab-runner OCP 4.3 (high)
Host - gitlab-runner OCP 4.3 (normal)
Host - gitlab-runner OCP 4.5 (high)
Host - gitlab-runner OCP 4.5 (normal)
Host - gitlab-runner hpe-apollo
Host - gitlab-runner ibm-p8
Host - gitlab-runner intel-canoepass
Host - gitlab-runner rock-zvm
Host - gitlab-runner xci30
Host - gitlab.com - website
Host - reporter-ng
Program - AWS costs: component
Program - AWS costs: daily
Program - AWS costs: service
Program - AWS costs: usage_type
Program - Beaker queues - aarch64
Program - Beaker queues - ppc64le
Program - Beaker queues - s390x
Program - Beaker queues - s390x_z13
Program - Beaker queues - x86_64
Program - Beaker system - gatto-1.tpb.lab.eng.brq.redhat.com
Program - Beaker system - gatto-2.tpb.lab.eng.brq.redhat.com
Program - Beaker system - kernelci-1.s390.bos.redhat.com
Program - Beaker system - kernelci-2.s390.bos.redhat.com
Program - Beaker system - kernelci-3.s390.bos.redhat.com
Program - Beaker system - kernelci-4.s390.bos.redhat.com
Program - Beaker system - kernelci-5.s390.bos.redhat.com
Program - Beaker system - kernelci-6.s390.bos.redhat.com
Program - Beaker system - rdma-dev-19.lab.bos.redhat.com
Program - Beaker system - rdma-dev-20.lab.bos.redhat.com
Program - Beaker system - rdma-perf-02.lab.bos.redhat.com
OK

* but it works on their computer

Monitoring: InfluxDB

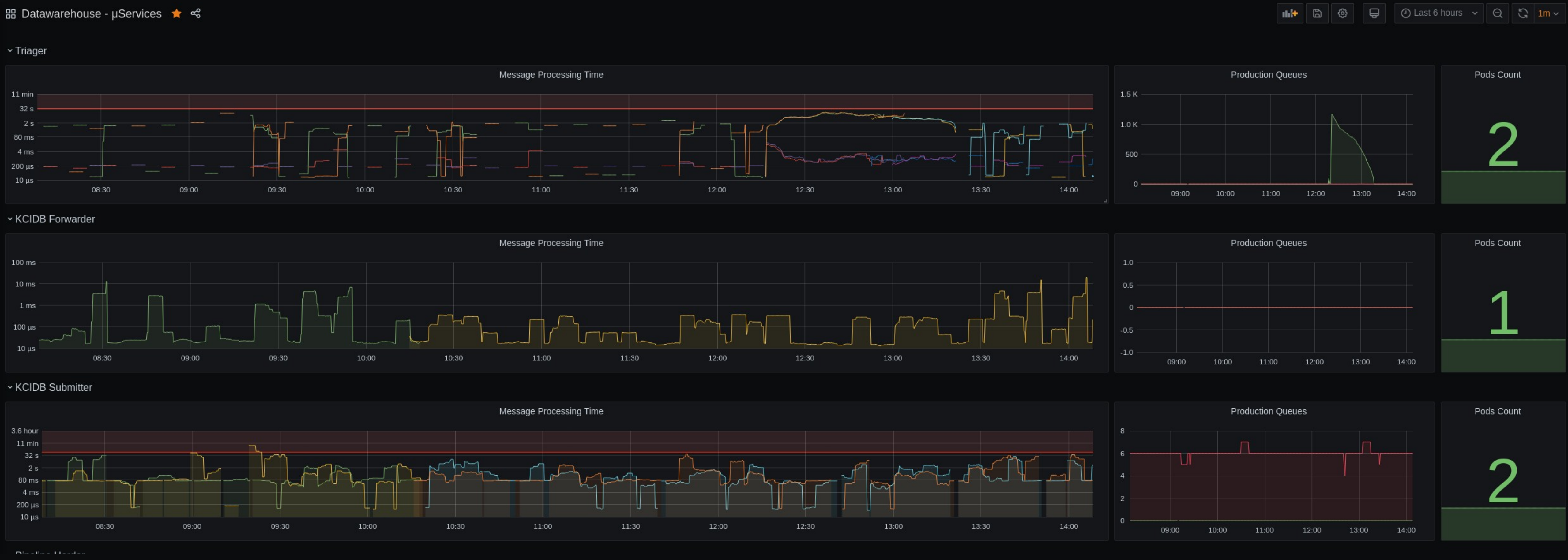
Where we were

- Custom solutions per application
    - Different data and intervals
    - Not generic, simple or safe
- Scrappers to filter logs and convert them into data points
    - Simpler than adapting sensible apps to push
- Telegraf as a PIM to bridge Prometheus to InfluxDB
    - Prometheus is turning into the standard

Monitoring: Prometheus

Where we're going

- Expose services internal status
  - Monitor what a service is doing and how long it's taking
- Prometheus as an import-and-forget solution
  - Python's prometheus-client
  - Built in on many services
- Telegraf sidecar for pods stats
- Kubernetes autodiscover and lay back


Datawarehouse - µServices
Last 6 hours
1m
Triager
Message Processing Time
11 min
32 s
2 s
80 ms
4 ms
200 µs
10 µs
08:30
09:00
09:30
10:00
10:30
11:00
11:30
12:00
12:30
13:00
13:30
14:00
Production Queues
1.5 K
1.0 K
500
0
Pods Count
2
KCIDB Forwarder
Message Processing Time
100 ms
10 ms
1 ms
100 µs
10 µs
Production Queues
1.0
0.5
0
-0.5
-1.0
Pods Count
1
KCIDB Submitter
Message Processing Time
3.6 hour
11 min
32 s
2 s
80 ms
4 ms
200 µs
10 µs
Production Queues
8
6
4
2
0
Pods Count
2

Sentry

How to be the first one to know when everything blows up

- Track errors in real time
- Internal: sentry.engineering.redhat.com
  - Community maintained
  - Works great
- External: sentry.io
  - Thanks packit!

Red Hat

How to find out what’s wrong?

IRC + Grafana FTW

IRC Alerts

- Someone is gonna read that

Grafana

- Easy to hack dashboards
  - Plus there are a ton of templates online !
- Allows combining different data sources
- Quick alerting and templating

Prevention

Queue all the stuff

Avoid losing data

- Message Queues are great for communicating pieces
  - Reliable and distributed
  - Allows to reject a message safely
- Webhooks are unreliable
  - Convert them to messages! [1]
- Schedule and retry messages without reinventing the wheel
- Test staging with production data
- AWS-hosted AMQP cluster becomes SPOF

1_ Webhook receiver https://gitlab.com/cki-project/cki-tools/-/tree/main/cki/cki_tools/webhook_receiver

Webhooks to AMQP

a.k.a. WebHook Receiver

- Plug in any webhook and distribute it reliably

```
                                                        Anywhere
                                                    +------------+
internal &                                       +->+ Consumer_1 |
gitlab.com    AWS Lambda            AWS EC2      |  +------------+
+--------+    +------------------+    +----------+  |  +------------+
| GitLab +--->+ webhook receiver +--->+ RabbitMQ +--+->+ Consumer_2 |
+--------+    +------------------+    +----------+  |  +------------+
                                                    . .           .
                                                    . .           .
                                                    |  +------------+
                                                    +->+ Consumer_n |
                                                       +------------+
```

1_ Resilient Messaging Queues https://cki-project.org/docs/hacking/operations/messaging/#use-case-gitlab-webhooks

Minimize the essentials

Less critical pieces means less critical failures

- Essential components
  - Needed for the service to run
- Necessary components
  - Have to work at least *sometimes*
- Optional components
  - Only provide observability and increase reliability
- Everything wrapped up into container images to freeze time…

Red Hat

Recovery

Rescheduling Messages with RabbitMQ

What goes around comes around

- Endlessly circulate messages until successfully handled
- Use DLX + TTL to requeue messages after some time [1]

```
                   x-dead-letter-exchange = e.retry.in
                    x-dead-letter-routing-key = {name}
+---------------+    +---------------+             +----------+
| e.prod.{name} +--->+ q.prod.{name} +------------>+ consumer |
+---------------+    +--+--------+---+             +----------+
                        |        ^
     +------------------+        +-------------+
     |                                         | routing-key = {name}
     v                                         |
+-----+------+    +----------------+    +-------+-----+
| e.retry.in +--->+ q.retry.{name} +--->+ e.retry.out |
+------------+    +----------------+    +-------------+
                  x-dead-letter-exchange = e.retry.out
                  x-message-ttl = n seconds
```

1_ Resilient Messaging Queues https://cki-project.org/docs/hacking/operations/messaging/

Insist until it works

"Ever tried. Ever failed. No matter. Try again. Fail again. Fail better." - Samuel Beckett

- Retry **every** network access multiple times
  - looping helper for shell code
  - common Python code to setup a retrying session
- Pipeline Herder:
  - Keeps track of failed GitLab jobs
  - Detects common transient errors
  - Retries jobs with increasing interval of time

Fallbacks

When retries are not enough for PSI

- Gitlab Runner's containerized jobs can run anywhere
- Runners set up on OSP, Beaker, different OCP clusters
- AWS-based production runners soon TBD™
- Fallbacks for multi-arch runners are hard to come by

RH IRC: #kernelci

https://cki-project.org

https://gitlab.com/cki-project

Red Hat