Scientific Approach of dCache monitoring.

(Giving me way too much credit)

Christian Voß

15th International dCache Workshop

June 2, 2021

HELMHOLTZ
RESEARCH FOR GRAND CHALLENGES

Motivation

dCache 6.2 and a Lucky Chance

Monitoring Infrastructure at DESY

> Usual collection of tools

Motivation

dCache 6.2 and a Lucky Chance

Monitoring Infrastructure at DESY

> Usual collection of tools

Usage: All over the Place

> Online view of billing data

Motivation

dCache 6.2 and a Lucky Chance

Monitoring Infrastructure at DESY

> Usual collection of tools

Usage: All over the Place

> Online view of billing data

> Custom data queries (typically against the admin interface)

Motivation

dCache 6.2 and a Lucky Chance

Monitoring Infrastructure at DESY

> Usual collection of tools

Usage: All over the Place

> Online view of billing data

> Custom data queries (typically against the admin interface)

> Host specific checks (e.g. database replication status)

Motivation

dCache 6.2 and a Lucky Chance

Monitoring Infrastructure at DESY

> Usual collection of tools

Usage: All over the Place

> Online view of billing data

> Custom data queries (typically against the admin interface)

> Host specific checks (e.g. database replication status)

> No mentioning of logging database

> Not only consolidate but add logging in some form as well

Previous Workflow

Manual Monitoring and Analysis

Functional Analysis

- Client based tests → *e.g.* `gfal-copy` vs. all doors and protocols
- Visualisation of billing data via ELK → Kafka stream since `4.2`
- Simple Pool-status via ELK → manual python scripts in self-deployed ELK stack
- Cron-jobs querying admin-doors → check Resilient status and push to Grafana

Response

- Spotted an issue → hunt for pool or door
- Dedicated manual search in logs on the nodes and admin-door pinboard
- Often: more detailed information in pinboard lost due to rotation

Does not scale well!

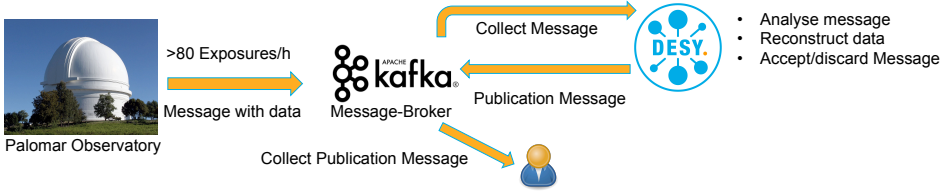
Using Events/Messages

A Motivation

Storage Events in dCache

- Reminder: 4.1 brought Kafka billing stream, idea: dCache as a workflow engine
- At DESY made data obsessed admin happy but never really went anywhere

Inspiration from a different field: AMPEL@DESY-Zeuthen

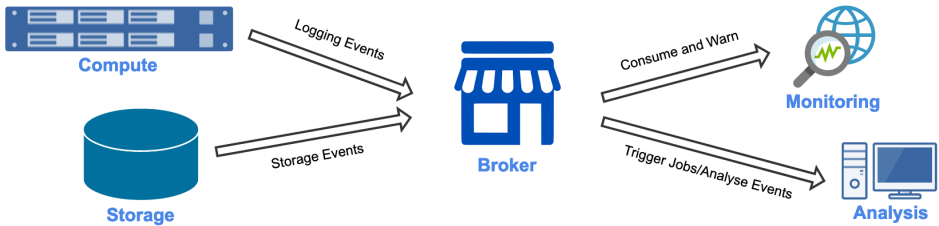


- Designed for real-time astronomy data analysis framework to find transient objects
- Message contains all relevant information, not pointing to data
- Adapt this to our dCache instances

Event Streams

Different Types of Events

Message Producer – Broker – Consumer Model

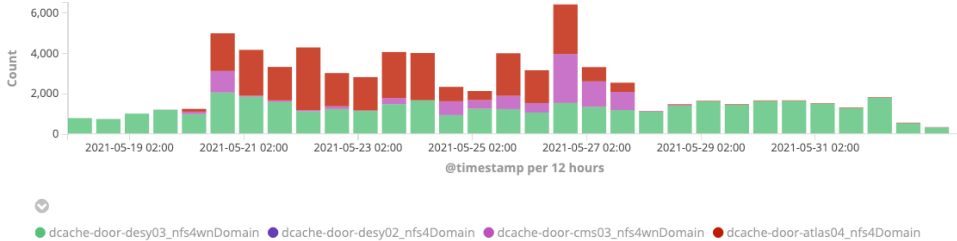


- Close to how dCache works internally (remember assigning `core` Domains?)
- Categorise messages/events in different manner
  - Message points to data, need interface to analyse data: Storage events
  - Message contains all the data: Self-contained events
  - AMPEL events are Self-contained as are our logging events

Logging Events

A Simple Example

- Some time ago dCache logged NFS export errors on some doors



What happened

- No tickets, no request for new exports
- New nodes in HTC cluster received `IPv6`, exports were `IPv4`
- Lessons:
  1. Notification on event!
  2. Automated response?

Logging Events

Other Examples

- Two errors observed relatively frequently:
  `'Internal repository error. Pool restart required'`
  `'Restarting due to fatal JVM error: java.lang.OutOfMemoryError:'`
- In the past errors often slipped us by

Internal repository error. Pool restart required

- Usually fixable by restart (trigger automatically)
- Trigger restart on event, notify on event

Restarting due to fatal JVM error: java.lang.OutOfMemoryError:

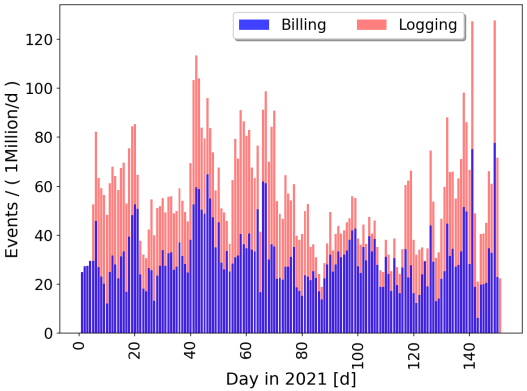
- Usually not fixed by restart, needs admin intervention (increase JVM memory)
- Notify on event

Requirements for our Instances

Expected Billing and Logging Entries

- Scalable system: Observe millions of events per day
- Easily integrated into DESY-IT
  elasticsearch logstash kibana
- Used with data analysis frameworks
  Apache Spark
- Easy to implement API
  python
- Adapt the workflow for billing stream
  kafka

Stacked data for 2021



Starting Point: Billing Events

Established Defined Workflow

Summerstudents in 2018 together with Kafka Integration into dCache

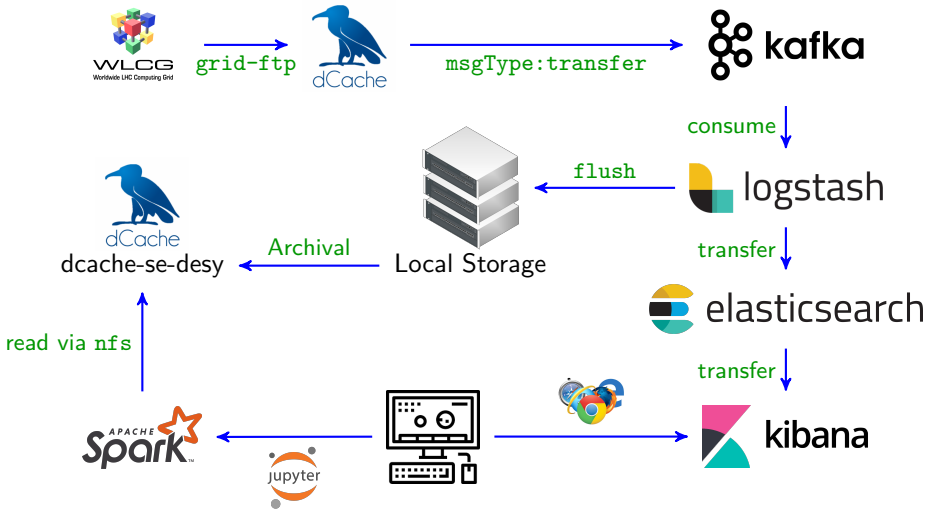
- Analysis of classic billing files, one of them experimented with `Kafka`-Billing stream
- Billing events well suited to monitor stability of system
- Apache Spark as analysis plattform for large scale analyses → JSON/Kafka connector

Necessary Infrastructure

- Zookeeper Cluser → needed for dCache anyway
- Kafka cluster → setup of old pool nodes
- ELK cluster → central IT-service
- Buffer-node to prepare archival → reuse dCache pool node
- Way to store and expose data → space in dCache dedicated for IT data

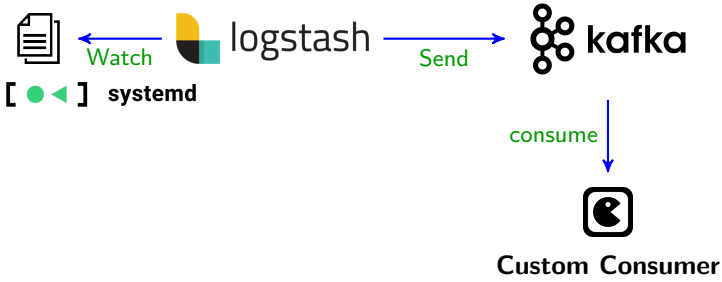
Billing Stream Workflow

Message Transport and Archival



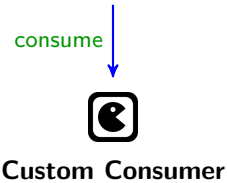
Generating the Events

From Local Node to Kafka-(Consumer)



Generating the Events

From Local Node to Kafka-(Consumer)



- Setup on every dCache node
- pre-6.2: Grok filter as input
- 6.2: use journalbeat as input
- Add additional tags (e.g. host type)
- logstash for additional input data

Generating the Events

From Local Node to Kafka-(Consumer)

- Connect to our existing cluster
- Topic per instance for logging
- Independent from billing

consume

Custom Consumer

- Setup on every dCache node
- pre-6.2: Grok filter as input
- 6.2: use journalbeat as input
- Add additional tags (e.g. host type)
- logstash for additional input data

Generating the Events

From Local Node to Kafka-(Consumer)

- Connect to our existing cluster
- Topic per instance for logging
- Independent from billing

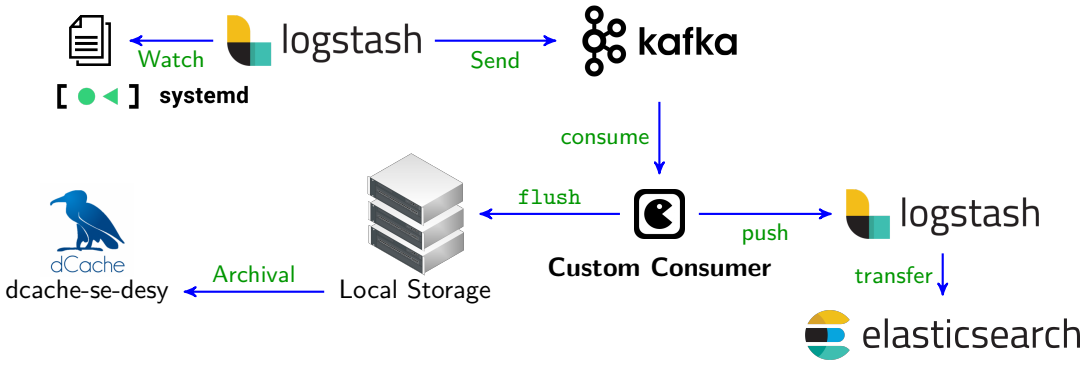
consume

- Setup on every dCache node
- pre-6.2: Grok filter as input
- 6.2: use journalbeat as input
- Add additional tags (e.g. host type)
- logstash for additional input data

- Message generated by journalbeat cluttered
- Retain human readabilty/consistency (pre-)6.2
- Clean up log message (e.g. remove timestamp)
- Add additional tags/fields based on message

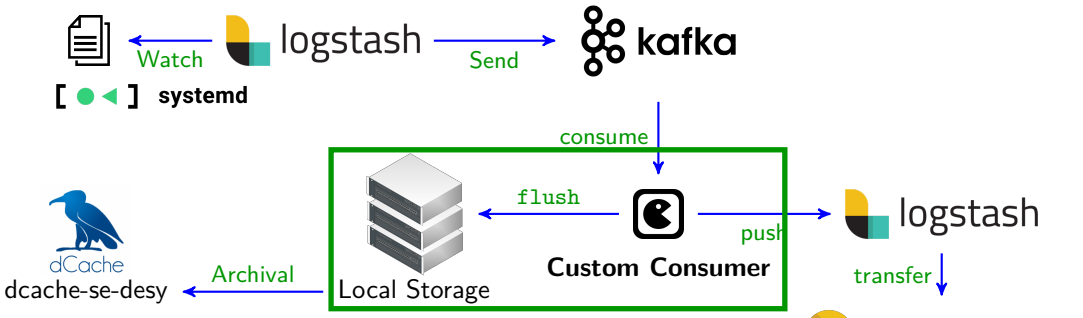
Storing Events

Storing in ElasticSearch and dCache



Storing Events

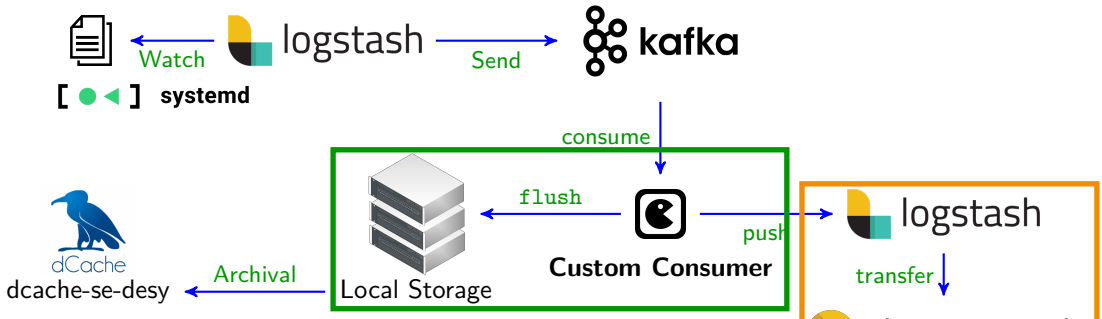
Storing in ElasticSearch and dCache



- Write files to disk
  - Daily
  - Role
  - Instance
- Sync every hour

Storing Events

Storing in ElasticSearch and dCache

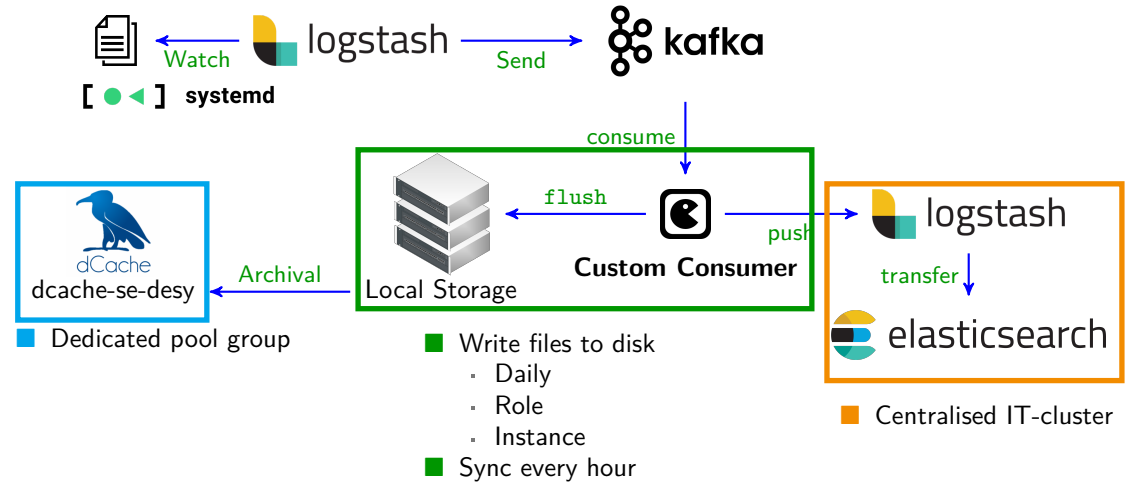


- Write files to disk
  - Daily
  - Role
  - Instance
- Sync every hour

- Centralised IT-cluster

Storing Events

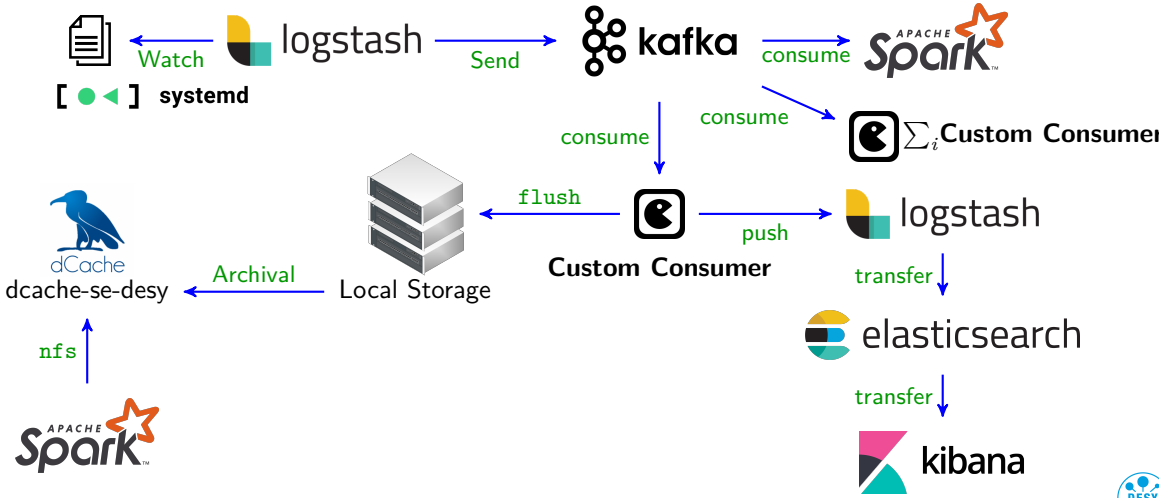
Storing in ElasticSearch and dCache



- Dedicated pool group
- Write files to disk
  - Daily
  - Role
  - Instance
- Sync every hour
- Centralised IT-cluster

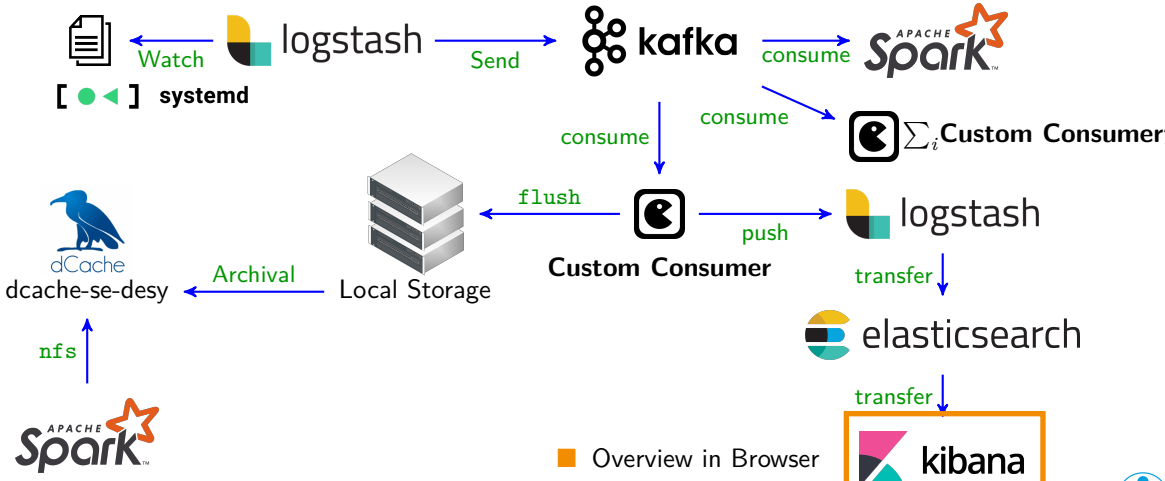
Analysing Events

Using Kibana and Apache Spark



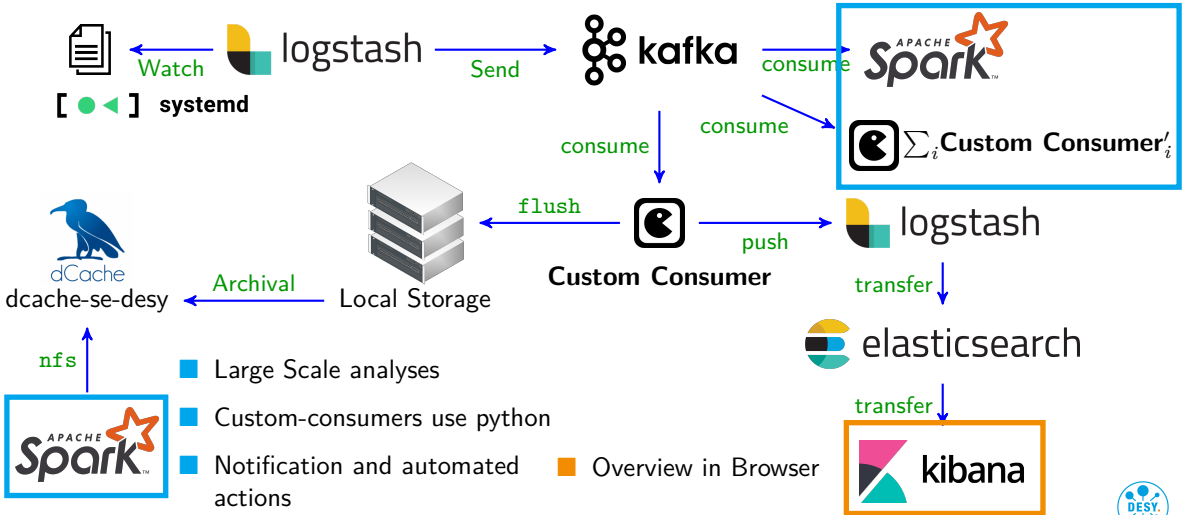
Analysing Events

Using Kibana and Apache Spark



Analysing Events

Using Kibana and Apache Spark



Missing Piece: Status of dCache

Custom Producer to Collect Status of dCache

Available Information so far

- Access to all storage events and to all logging events
- Nothing stored locally → switch logging to `INFO` to get `pinboard` level logging
- Easy to connect failed transactions to broken pools/doors/heads

Missing Information

- Access logs of the doors →just needs to be done
- Overall status
  - Which pools were online
  - Which doors were online
- Information available but not in event form
- Introduced our own custom producer collection this information

dCache Pool-Info-Stream

dCache Status Part I

Poll

tags:pool-info

kafka

- Custom producer (in python)
- Produces message every 15m
- Topic per dCache

consume

logstash

consume

Custom Consumer

transfer

elasticsearch

- Custom Sweeper for Cached data
- Adjust allowed movers
- Automatic migration

transfer

kibana


```
"known_cores": [
  "dcache-core-desy04_messageDomain",
  "dcache-se-desy04_messageDomain",
  "dcache-dir-desy04_messageDomain"
],
"space_removable": 0,
"space_free": 9433703481002,
"n_restores": 0,
"@timestamp": "2021-06-02T12:20:41.400Z",
"space_precious": 27924790280534,
"LargeFileStore": "PRECIOUS",
"n_queued_restores": 0,
"n_files": 19948,
"queues": {
  "ftp-q": {
    "queued": 0,
    "active": 0
  },
```

dCache Door-Info and Head-Info Streams

dCache Status Part II and III

Still to Be Done

- Began working on door-info stream
- Due to structure of different protocols not stright forward
- Active transfers differently tracked across doors
  - High Details for `FTP`, `dCap`, and `NFS`
  - Limited Details for `WebDav` and `Xroot`
- Find unifying structure
- Similar questions for heads (do we even need it?)
- Need for additional information in pool-info as well, e.g. all active movers to map to transfers on `NFS` door

Where Do We Want to Be?

Ambitious dream: doing predictive maintenance for dCache

What is done

- Rather advanced in collecting data to monitor dCache without need to always log on nodes
- Begun to collect additional data beyond logging and billing
- We have an easy to use framework to create alarms based on logging events
- With Spark we have an analysis plattform that can scale
- We deploy a small cluster on old heads, scale out to the NAF@DESY

What me miss – apart from data

- Automated responses instead of notifications (so far only pool-cache automatically freed)
- Deployment janky: python scripts as `systemd` services on small VMs
- Experience on how to do non-supervised learning
- Deploy full workflows (e.g. from door-info to draining on failure)

Conclusions

On Using Kafka with all Things dCache (and leaving lucid ML dreams behind)

- Even without making use of events → aggregating logs is a quality of life improvement
- Writing custom messages helped consolidate into a single entry point
- Build dashboards for customers showing both status of transfers and (re-)stores
- Archive of data for later forensics

Happy to Share

- Apologies for being terrible at documentation
- Request from BNL to share our journalbeat and logstash configuration
- Created repositories within the dCache GitHub for journalbeat and logstash
- Python Kafka code not public but easy to do as well
- Feel free to contact us, and remind me if I forget about it