Updating and Operating a Systemd-dCache.

Thoughts and Experiences

Christian Voß

dCache Workshop on systemd-Integration

November 18, 2020

HELMHOLTZ
RESEARCH FOR GRAND CHALLENGES

dCache Update Report

Update to 6.2-dCache from 5.2, 6.0, 6.1

We operate three 6.2-dCache instances at the moment

Pre-Production Cluster

- dCache-Operations dCache (update from 6.1)
- WLCG/EU-project dCache (update from 6.0)

Production Cluster

- ATLAS dCache (update on 21th October 2020 from 5.2)
- CMS dCache (update planned for early December from 5.2)

Updates themselves were smooth as usual, however ...

dCache on Systemd

Changes on the dCache Environment

- Update and start without problem:

```
[root@dcache-head-dot01 ~]# systemctl list-dependencies dcache.target
dcache.target
● ├─dcache@dcache-head-dot01_cleanerDomain.service
● ├─dcache@dcache-head-dot01_coreDomain.service
● ├─dcache@dcache-head-dot01_gplazmaDomain.service
● ├─dcache@dcache-head-dot01_messageDomain.service
● ├─dcache@dcache-head-dot01_namespaceDomain.service
● ├─dcache@dcache-head-dot01_nfs3Domain.service
● ├─dcache@dcache-head-dot01_nfs4Domain.service
● ├─dcache@dcache-head-dot01_resilientDomain.service
● ├─dcache@dcache-head-dot01_srmDomain.service
● └─dcache@dcache-head-dot01_utilityDomain.service
[root@dcache-head-dot01 ~]#
```

- What happened to `dcache status` (couple of dependencies on):

```
[root@dcache-head-dot01 ~]# dcache status

The dCache is managed by systemd.
Please use 'systemctl' instead.
```

  - Loadbalancing scripts to check for active doors
  - Overall service status monitoring (became obsolete)
  - Small custom monitoring scripts

- What happened to the log-files

Alternatives to dcache status

Use Systemd Calls directly

- Get status of each service `systemctl status dcache@dcache-atlas154-01Domain.service`
- For all domains: expect using `systemctl list-dependencies dcache.target`:

```
[root@dcache-head-dot01 ~]# systemctl list-dependencies dcache.target
dcache.target
● ├─dcache@dcache-head-dot01_cleanerDomain.service
● ├─dcache@dcache-head-dot01_coreDomain.service
● ├─dcache@dcache-head-dot01_gplazmaDomain.service
● ├─dcache@dcache-head-dot01_messageDomain.service
● ├─dcache@dcache-head-dot01_namespaceDomain.service
● ├─dcache@dcache-head-dot01_nfs3Domain.service
● ├─dcache@dcache-head-dot01_nfs4Domain.service
● ├─dcache@dcache-head-dot01_resilientDomain.service
● ├─dcache@dcache-head-dot01_srmDomain.service
● └─dcache@dcache-head-dot01_utilityDomain.service
[root@dcache-head-dot01 ~]#
```

- Not machine readable, output colour coded, return code 0, can't monitor restarting services
- No easy way to get the information, but
  - `systemctl show --property=ConsistsOf dcache.target` list all services in the target
  - `systemctl is-active dcache@dcache-atlas154-01Domain.service` checks status

Using Systemd Funcionality

Two Commands Fit Our Needs

- Commands listed before applicable to our cases
- Wrote simple replacement script for our admins

```
[root@dcache-head-dot01 ~]# dcache_status
dcache@dcache-head-dot01_utilityDomain.service :                active
dcache@dcache-head-dot01_messageDomain.service :                active
dcache@dcache-head-dot01_nfs3Domain.service :           active
dcache@dcache-head-dot01_resilientDomain.service :              active
dcache@dcache-head-dot01_srmDomain.service :            active
dcache@dcache-head-dot01_namespaceDomain.service :              active
dcache@dcache-head-dot01_nfs4Domain.service :           activating
dcache@dcache-head-dot01_cleanerDomain.service :                active
dcache@dcache-head-dot01_gplazmaDomain.service :                active
dcache@dcache-head-dot01_coreDomain.service :           active
```

```
#!/usr/bin/bash

RED='\033[0;31m'
GREEN='\033[0;32m'
BLUE='\033[0;34m'
NC='\033[0m' # No Color

for SERVICE in $(systemctl show --property=ConsistsOf dcache.target | cut -d '=' -f 2)
do
    status=$(systemctl is-active $SERVICE)
    statuscode=$?

    if [ "$statuscode" -eq "0" ]; then
        printf "${BLUE}$SERVICE${NC} : \t\t ${GREEN}$status${NC}\n"
    fi

    if [ "$statuscode" -ne "0" ]; then
        printf "${BLUE}$SERVICE${NC} : \t\t ${RED}$status${NC}\n"
    fi

done
```

Using Systemd Flexibility

More Flexible Decentralised dCache Instances

- Make use of mapping dCache Domain → systemd service
  - Rewrote Puppet manifests to allow easily have services started/stopped
  - Increase centralisation and decrease neccessity for remote logins
- Systemd allows for easier monitoring through suites like `Icinga`
- Easier resource monitoring due to process isolation
- A lot more to learn about the possibilities systemd might offer

Revisiting Log-File Strategy

Trigger to Change Paradidgm

- 6.2-dCache retires classic log files by default
- Pre-6.2-dCache stored log files on nodes
- Rapidly growing number of domains (2065 XFEL)/nodes (198 XFEL) makes it infeasible
- Work ongoing on using `Logstash` and `Kafka` to collect and archive logging when 6.2 released
- Revisit ideas with regard to `systemd`
- System should be invariant with regard to dCache release

Kafka: Event-Driven Logging and Monitoring

Adapt Billing Stream Workflow for Other Uses

Why Centre around Kafka

- Infrastructure in place for archival/analysis of the billing stream
- Experience with custom producers/consumers collecting information on pools/doors
- Event driven nature suites monitoring
- Easy integration into other systems

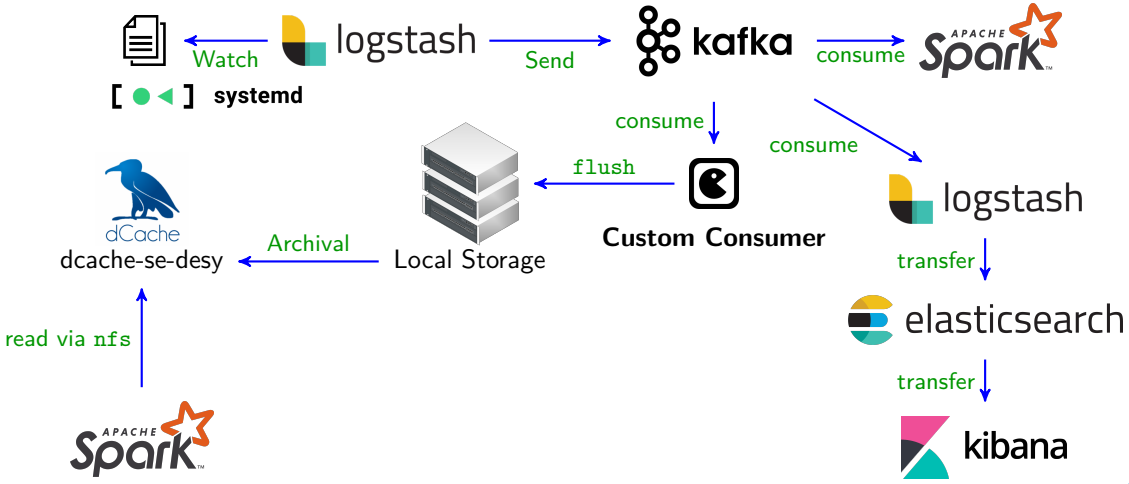
Connecting dCache with Kafka

- dCache ships with producer for billing and alarms
- Collect logging data through `Journalbeat` and/or `Logstash` on nodes
- Send Logstash data to Kafka and connect to existing infrastructure

Idea: Consolidate all streams into a single repository of data

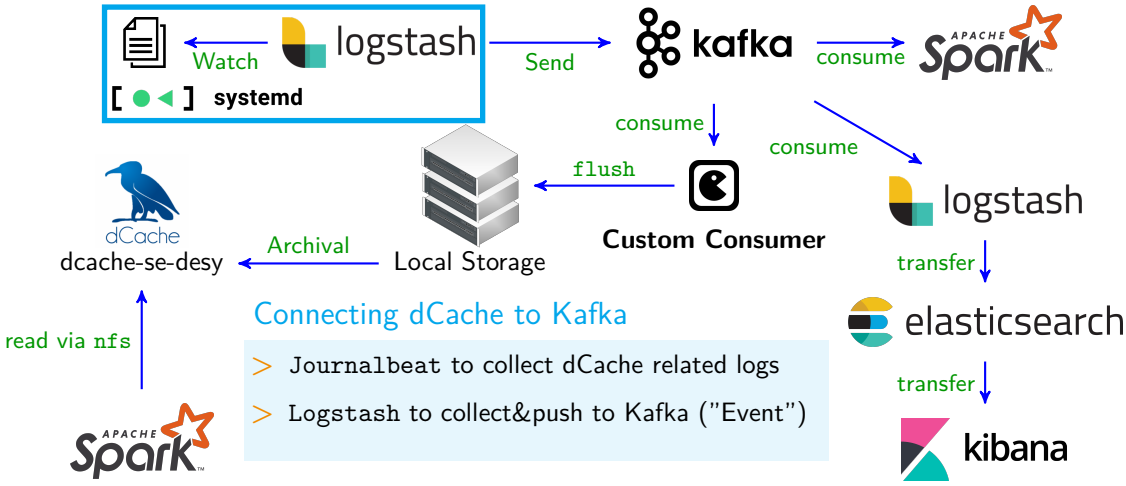
dCache Kafka Based Logging

Migration into New Paradigm



dCache Kafka Based Logging

Migration into New Paradigm

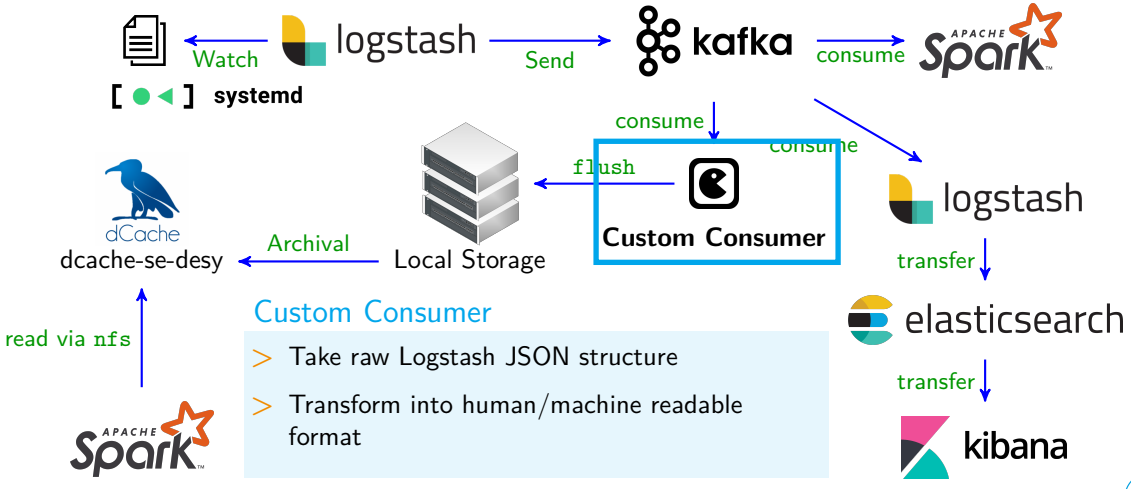


Connecting dCache to Kafka

- Journalbeat to collect dCache related logs
- Logstash to collect&push to Kafka ("Event")

dCache Kafka Based Logging

Migration into New Paradigm

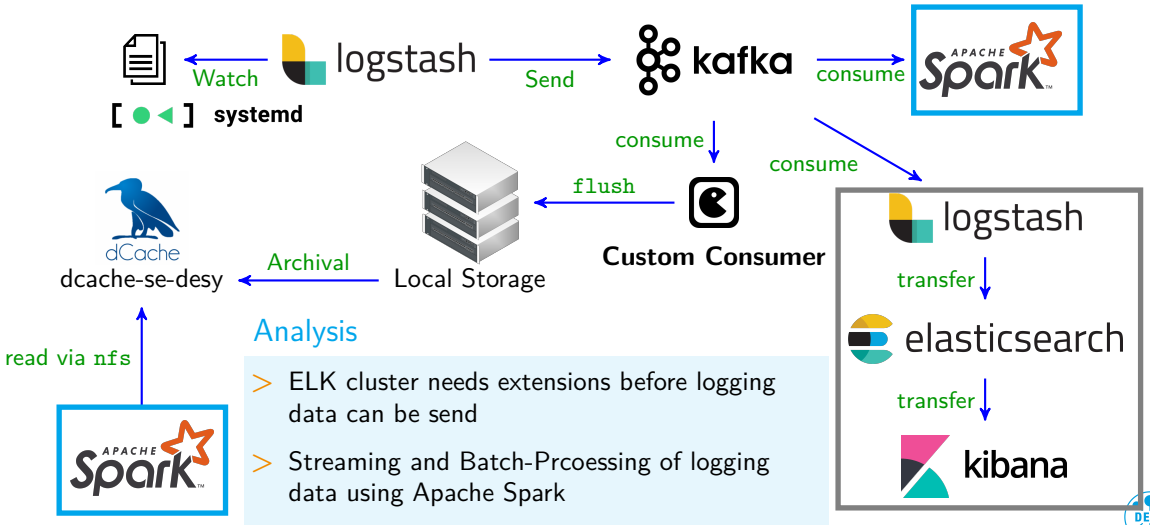


Custom Consumer

- Take raw Logstash JSON structure
- Transform into human/machine readable format

dCache Kafka Based Logging

Migration into New Paradigm



Analysis

- ELK cluster needs extensions before logging data can be send
- Streaming and Batch-Prcoessing of logging data using Apache Spark

Individual Configurations

Journalbeat, Logstash, Custom Consumer

Journalbeat

- Select `system-dcache.slice` to catch all dcache related logging (including spawned processes such as `hsm` scripts)
- Different output choices: Logstash, Kafka, File, ...
- Choice for Logstash for consistency with pre-6.2-dCache and message manipulation

```
name: dcache-head-dot01
tags: [ "journald" ]

journalbeat.inputs:
- paths: []
  seek: cursor
  include_matches :
     - "_SYSTEMD_SLICE=system-dcache.slice"

output:
  logstash:
    hosts:
    - localhost:5044
```

Individual Configurations

Journalbeat, Logstash, Custom Consumer

Local Logstash

- Input file or Journalbeat
- Remove systemd related fields to stream-line messages

```
input {
  beats {
    port => 5044
    tags => [ "dot",  "dir",  "logs"]
  }
}
filter {
  mutate { remove_field => [ "event", "agent", "journald", "syslog", "ecs" ] }
}
output {
  kafka {
        codec => json
        client_id => "dcache-head-dot01"
        topic_id => "logging-dot"
        bootstrap_servers => "broker01.desy.de:9092,broker02.desy.de:9092,broker03.desy.de:9092"
        id => "dcache-logging-dot"
      }
  }
```

Individual Configurations

Journalbeat, Logstash, Custom Consumer

Custom Consumer

- Use the `pykafka` library and small custom tool selection
- Systemd-service per instance on a collector nodes → archival in to dCache
- Human/Machine readable JSON files invariant of (pre-)6.2-dCache split by node-role
- establish consumer acting on certain logging events

```
{
  "instance": "atlas",
  "host_type": "dir",
  "host": "dcache-dir-atlas04",
  "Domain": "dcache-dir-atlas04_cleanerDomain",
  "date": "2020-11-17T15:18:23.869Z",
  "message": "17 Nov 2020 16:18:23 (cleaner) [dcache-atlas157-02 PoolRemoveFilesFromHSM] Received failure from pool:
  Pool has no tape backend",
  "pid": 129370
}
```

Current stream-lined JSON structure

Closing Thoughts

Lessons and Acces to Configs

- Updating from 5.2,6.0,6.1 to 6.2 without problems
- Enable systemd on all 6.2-dcache instances
- Rewrite a lot of dependencies on old status output
- Systemd increases ease of monitoring
- Systemd increases centralisation
- Use Journalbeat and Kafka to establish event based logging
- Happy to share our code
  - for Puppet modules for: Journalbeat, Logstash, and Kafka (maybe under dCache github)
  - for custom Kafka producer and consumers

  just let me know

Thanks a lot and please feel free to ask questions