Archive Data Migration to AWS

How not to use a Year's Budget in a Week

Tobias Schiebeck
University of Manchester

The Task

- Make sure Opencast does not access S3 unnecessarily
  - Adding a dublincore_xml column to the archive_episode table so that the DC can be read from there rather than the filesystem
- Move the database into RDS
  - Sanitize the database
  - Remove old jobs
- Transfer the data into S3

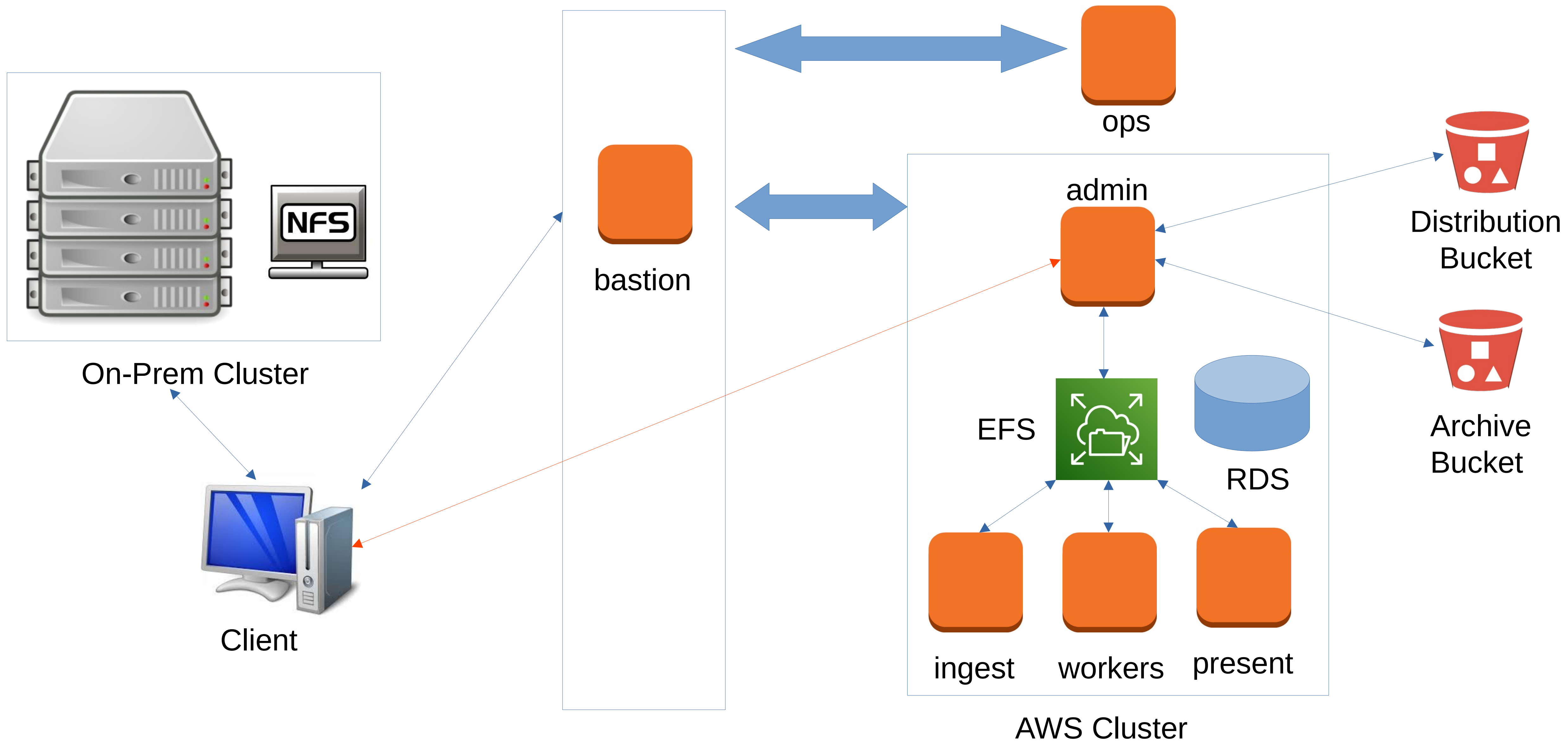
MANCHESTER
1824
The University of Manchester

The Task in Numbers (On-Prem)

- Archive has
  - ~ 2,000,000 assets
  - ~ 400,000 episodes
  - ~ 200,000 “latest versions”
- In the file system this equates to:
  - ~200 TB archive data
  - ~20 TB distribution data

TO BE TRANSFERRED

Infrastructure



MANCHESTER
1824
The University of Manchester

The Database

- The database transfer was pretty trivial using DBeaver, transferring the on-Prem MS-SQL DB as export straight into AWS RDS.
  - Minor changes in the structure
  - Flatten episode versions
    - Only move latest version and renumber it to 0
  - Drop table content on jobs table

Transferring the Data

- Find the mediapackage data
- Check md5sum of mediapackage asset on NFS (On-Prem)
- Open a tunnel through the bastion host to write data on the admin node into EFS
- Copy the mediapackage asset onto EFS
- Check md5sum of mediapackage asset in EFS
- Call REST endpoint on AWS admin to move the mediapackage to S3 bucket

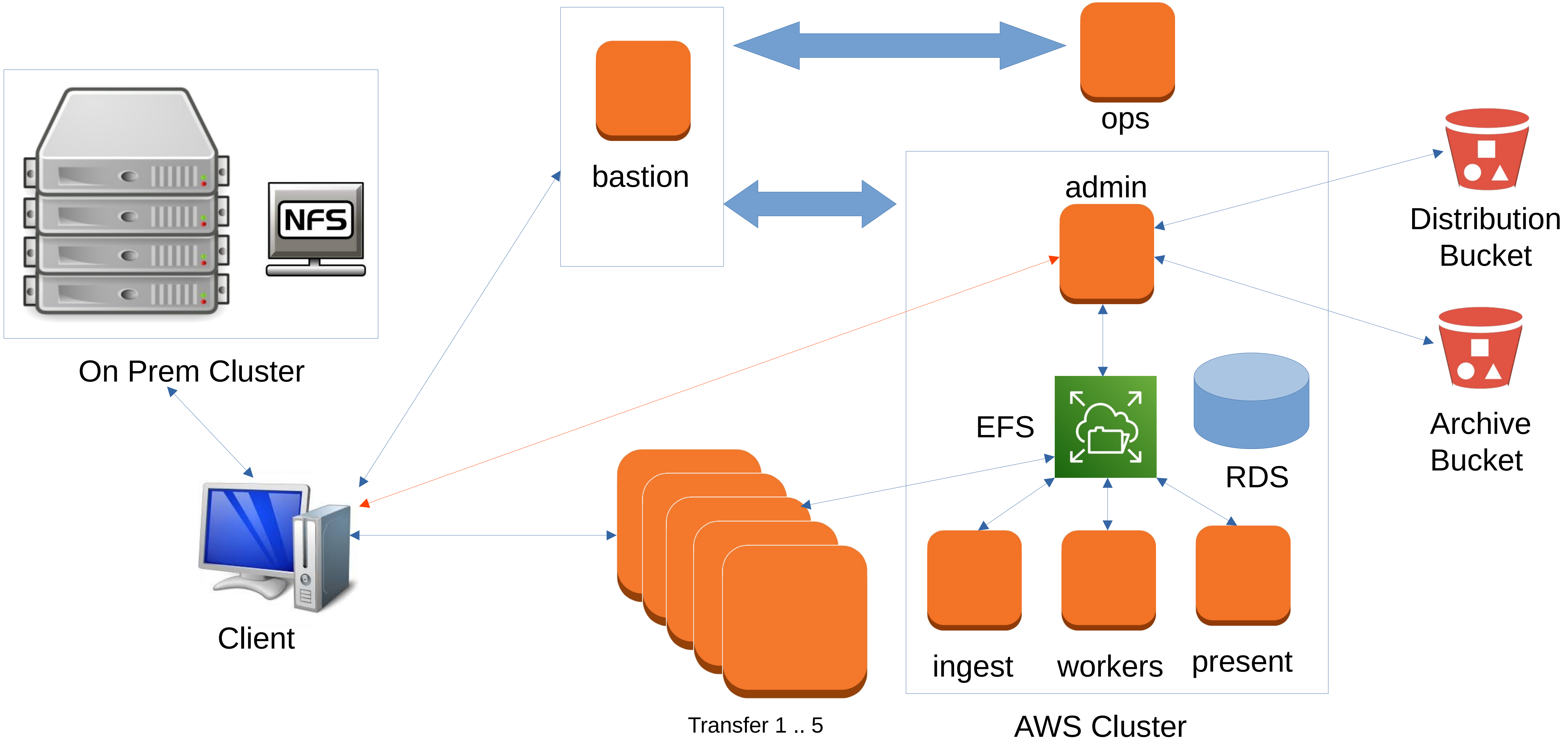
This takes two months solid

Two Months weren’t an Option

- We added 5 transfer nodes to the AWS infrastructure
- Transfer nodes can write directly into EFS

This reduces the transfer time to two weeks

Altered Infrastructure



And so we Transferred …

- Everything worked well and transferred happily
  - Costs were rising (… and so they spent happily ever after) …
    - **We didn't know what costs to expect!**

- What was going on:
  - Data got onto EFS and it grew !!!!
  - Should the data not be moving on?
    - It did … but admin could not keep up with moving the data across to S3
    - Why was our EFS data still not getting less?
      - The data was copied into the workspace area … and left in there (clean-up didn't seem to work)

- **EFS grew to about 20TB, at $2,500/day, when we pulled the emergency break and tried to figure out what was going wrong.**

And so we Investigated …

- Admin was overwhelmed with the incoming data and could not keep up
  - ➔ We reduced the number of transfer nodes to 3
- The transfer workflow didn’t clean up
  - ➔ We modified the transfer scripts to clean the workspace once the mediapackage was moved.

- **EFS was slowly cleaned out and the data got transferred.**

Results on AWS

- Archive now has
  - ~ 1,000,000 assets
  - ~ 200,000 episodes
  - ~ 200,000 “latest versions”

- In the S3 buckets this equates to:
  - ~170 TB archive data
  - ~20 TB distribution data

DONE !

MANCHESTER 1824
The University of Manchester

Lessons Learned …

… **Expensively** … We ran up a **$25,000** bill for EFS while transferring data with a total AWS bill of **$32,500** over **one month**.

Therefore:

- Know what costs to expect when transferring data into AWS
- Check that all parts of your operation can cope with the amount of data you send
- Verify that you get rid of intermediate data in the process (especially on EFS)

Thank you.
Any questions?

Tobias.Schiebeck@manchester.ac.uk