Macaroons

Paul Millar
(on behalf of the dCache team)

CHEP 2018 at Sophia, Bulgaria; 2018-07-09
https://indico.cern.ch/event/587955/

Macaroons “cheat-sheet”

- Macaroon is a **bearer token**.
- Macaroon contains zero or more **caveats**.
- Each caveat **limits** something about the macaroon:

  who can use it,

  when they can use it, or

  what they do with it.
- Anyone can **add a caveat** to a macaroon

  … creating a new, more limited macaroon.
- No one can **remove a caveat** from a macaroon

Bearer Tokens

Photo by Alan Cleaver (CC-

How caveats work?

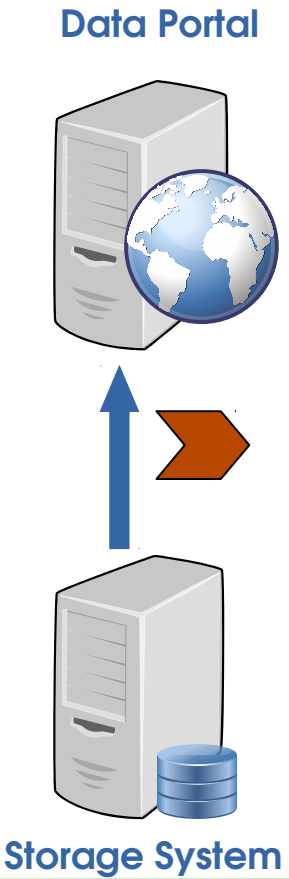


Diagram idea stolen from Tigran

How caveats work?

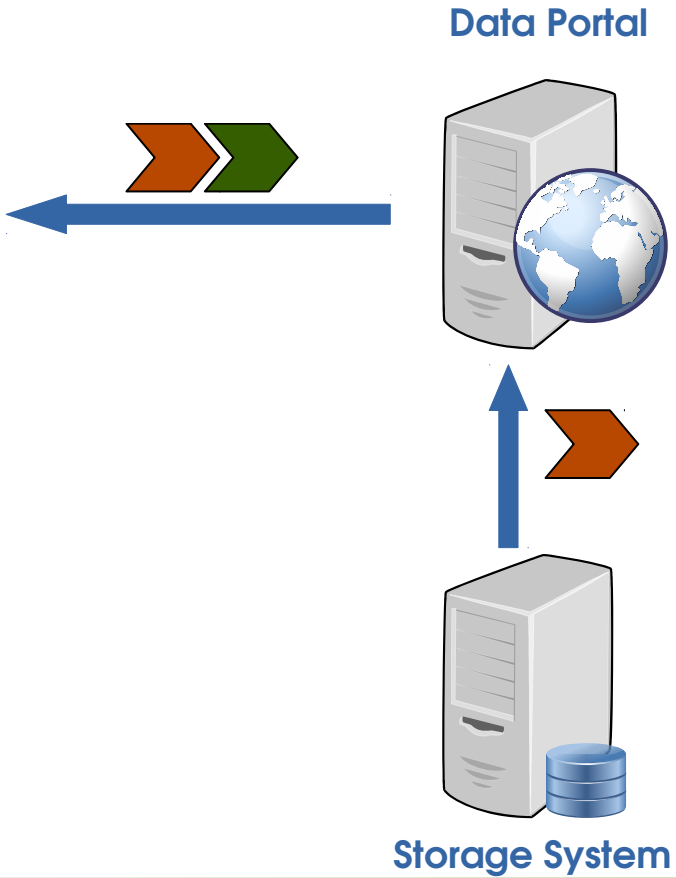


Diagram idea stolen from Tigran

How caveats work?

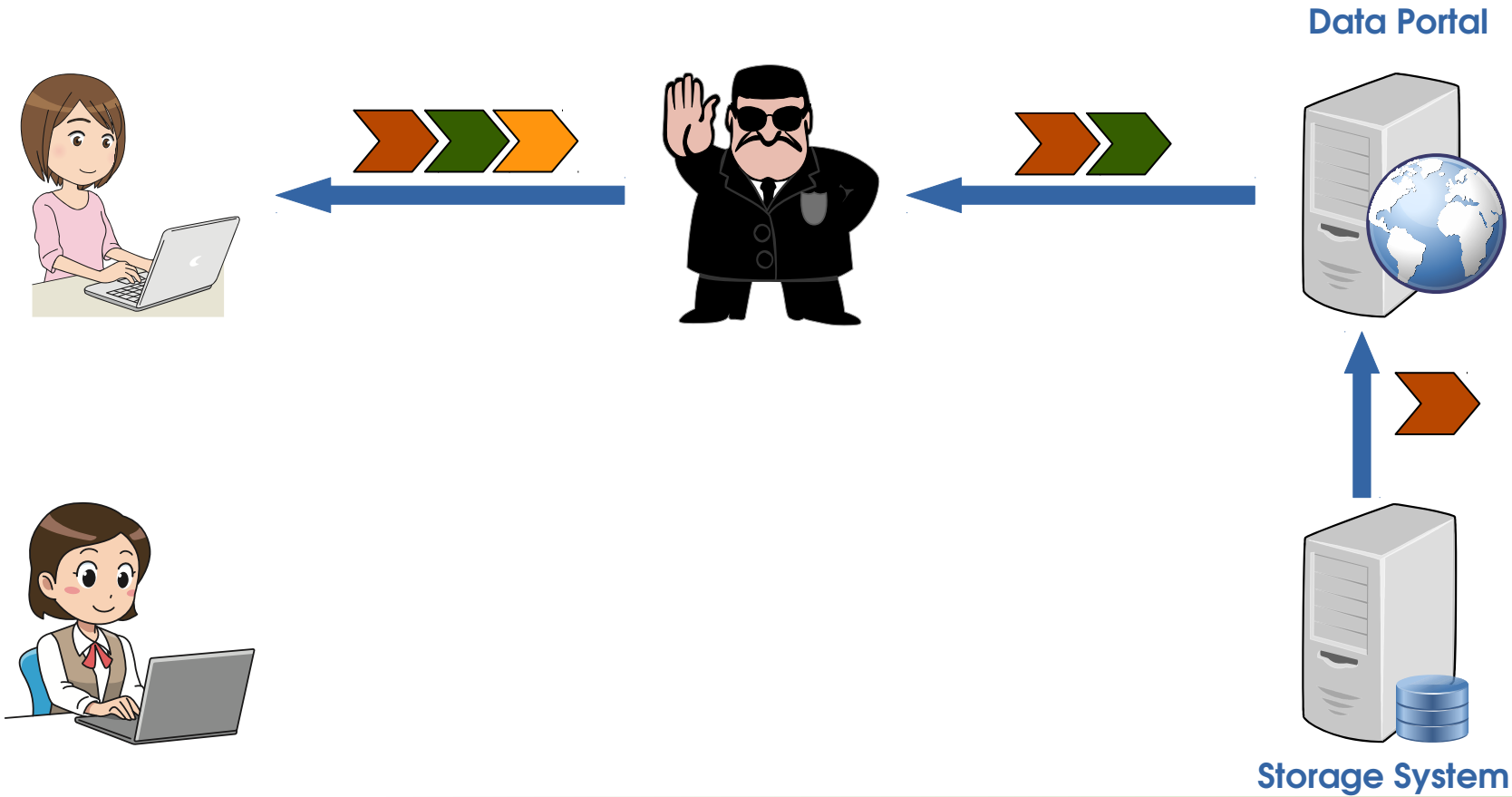


Diagram idea stolen from Tigran

How caveats work?

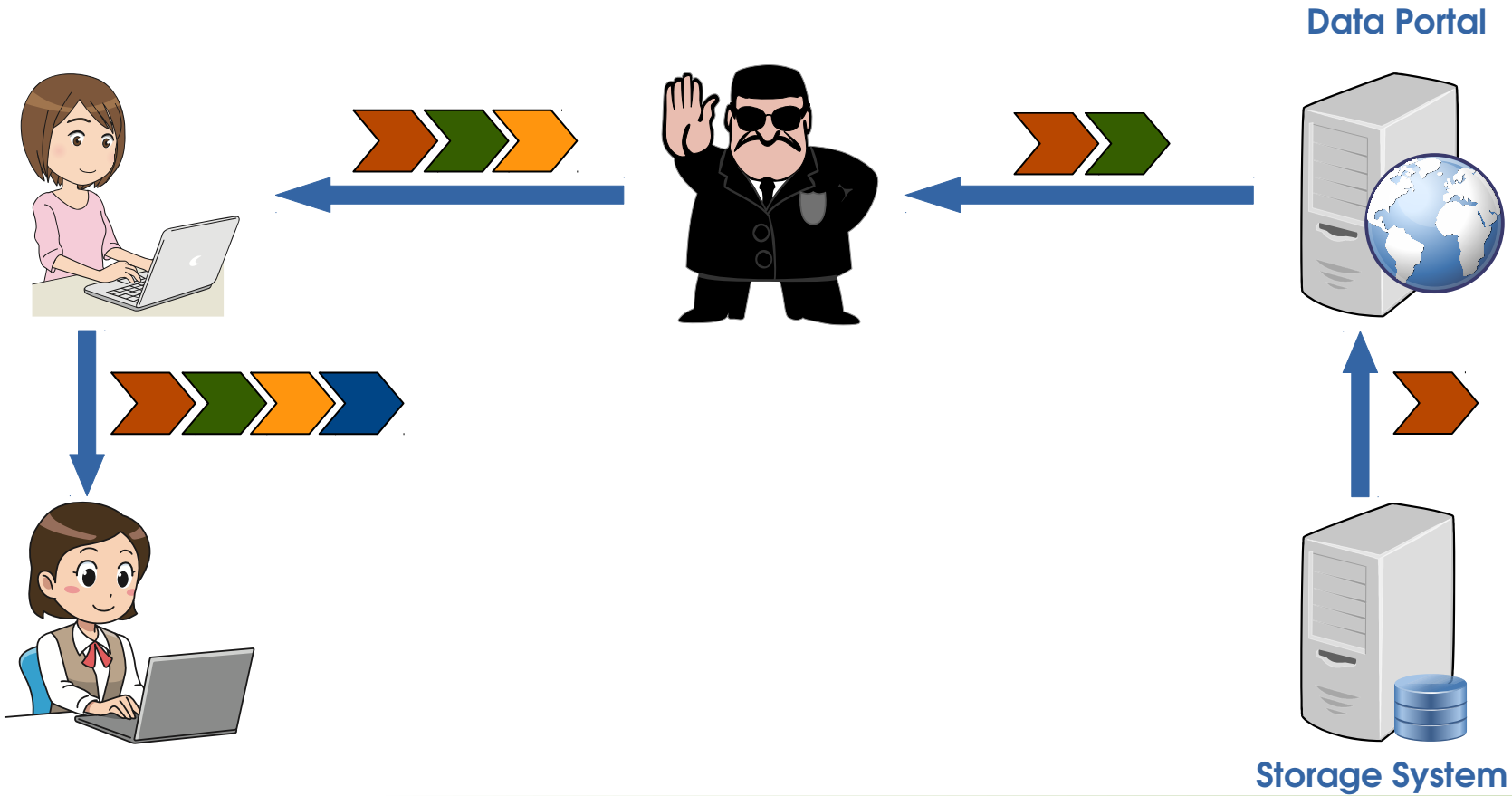


Diagram idea stolen from Tigran

How caveats work?

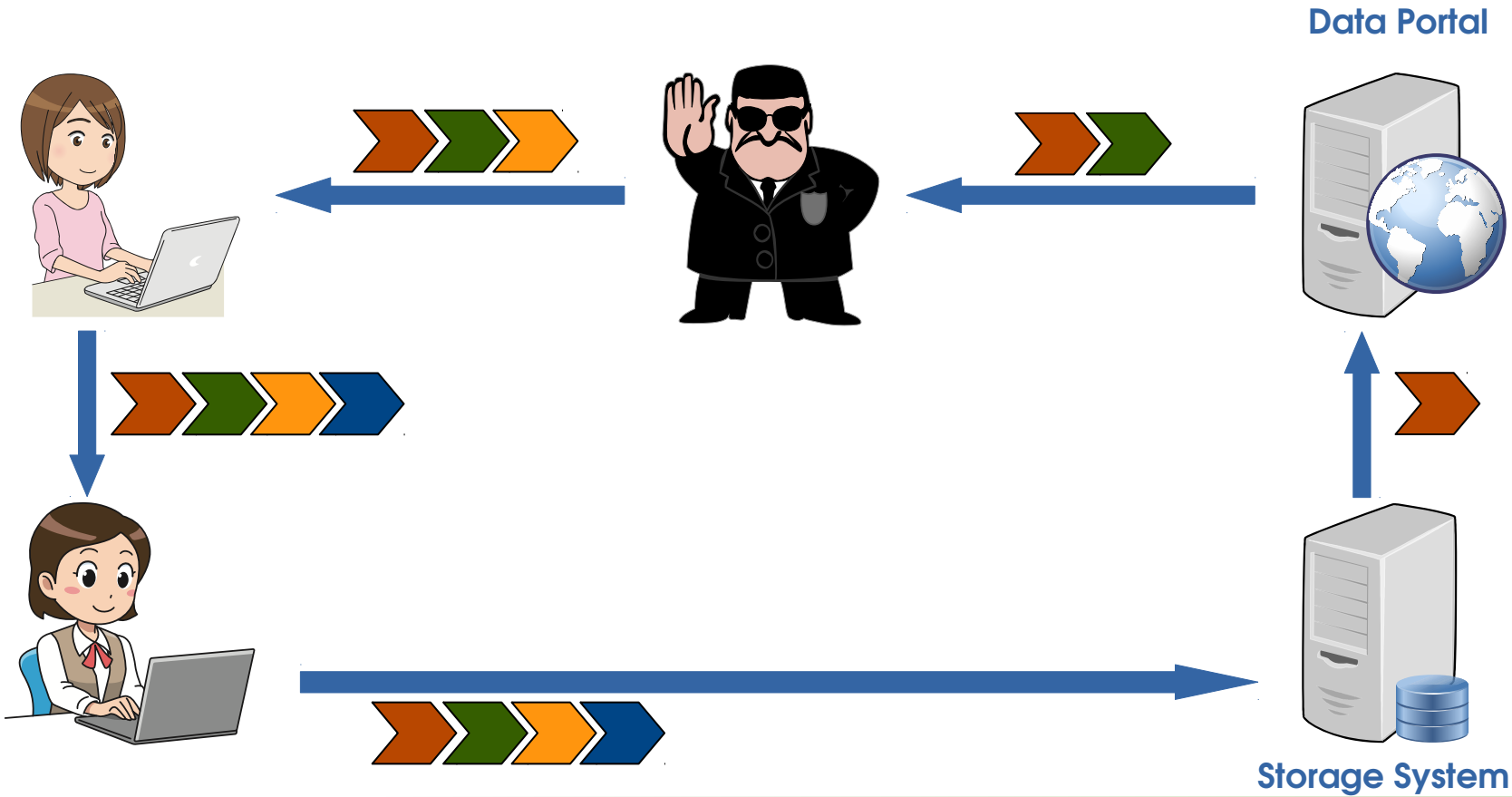


Diagram idea stolen from Tigran

How caveats work?

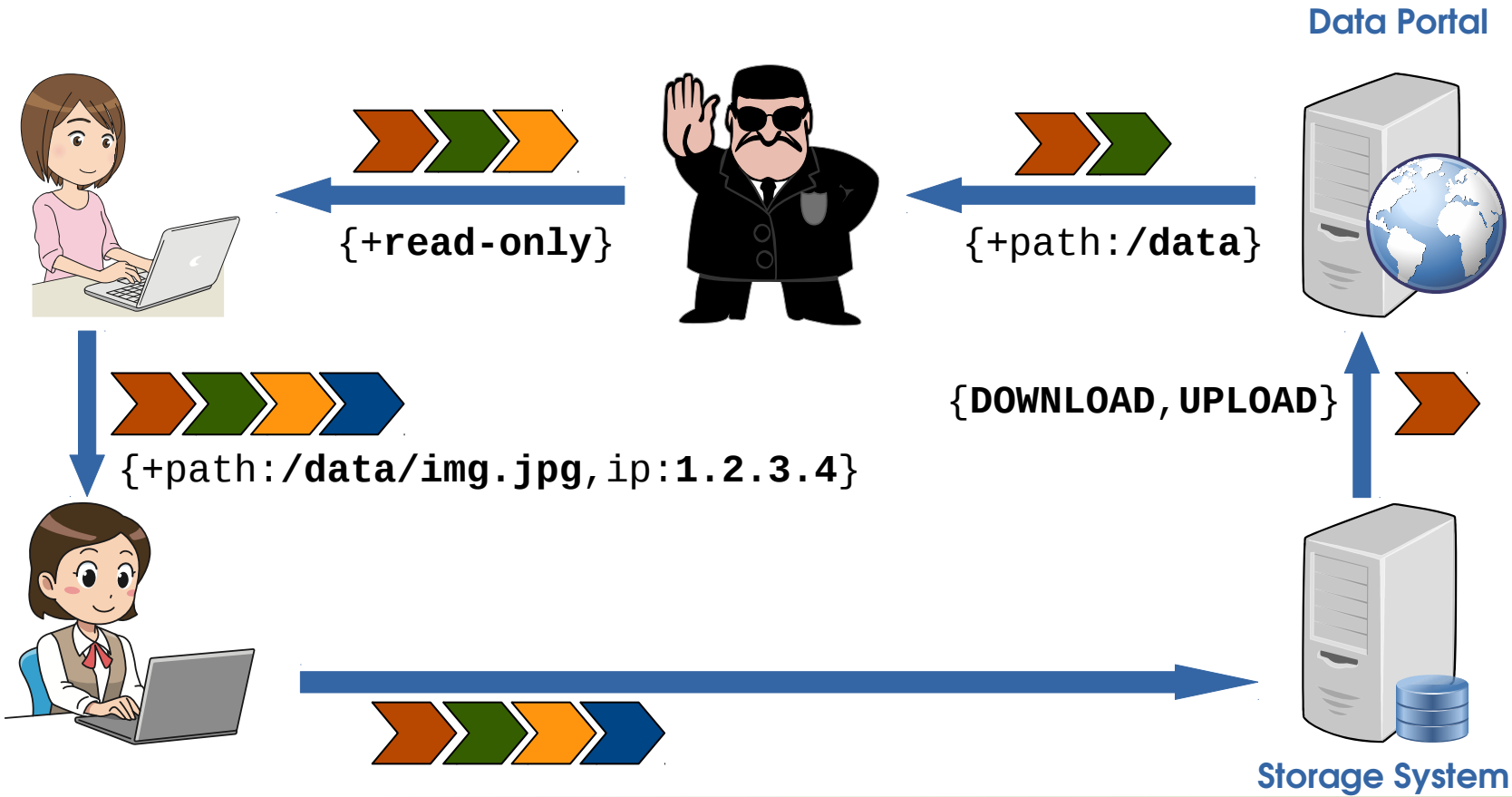
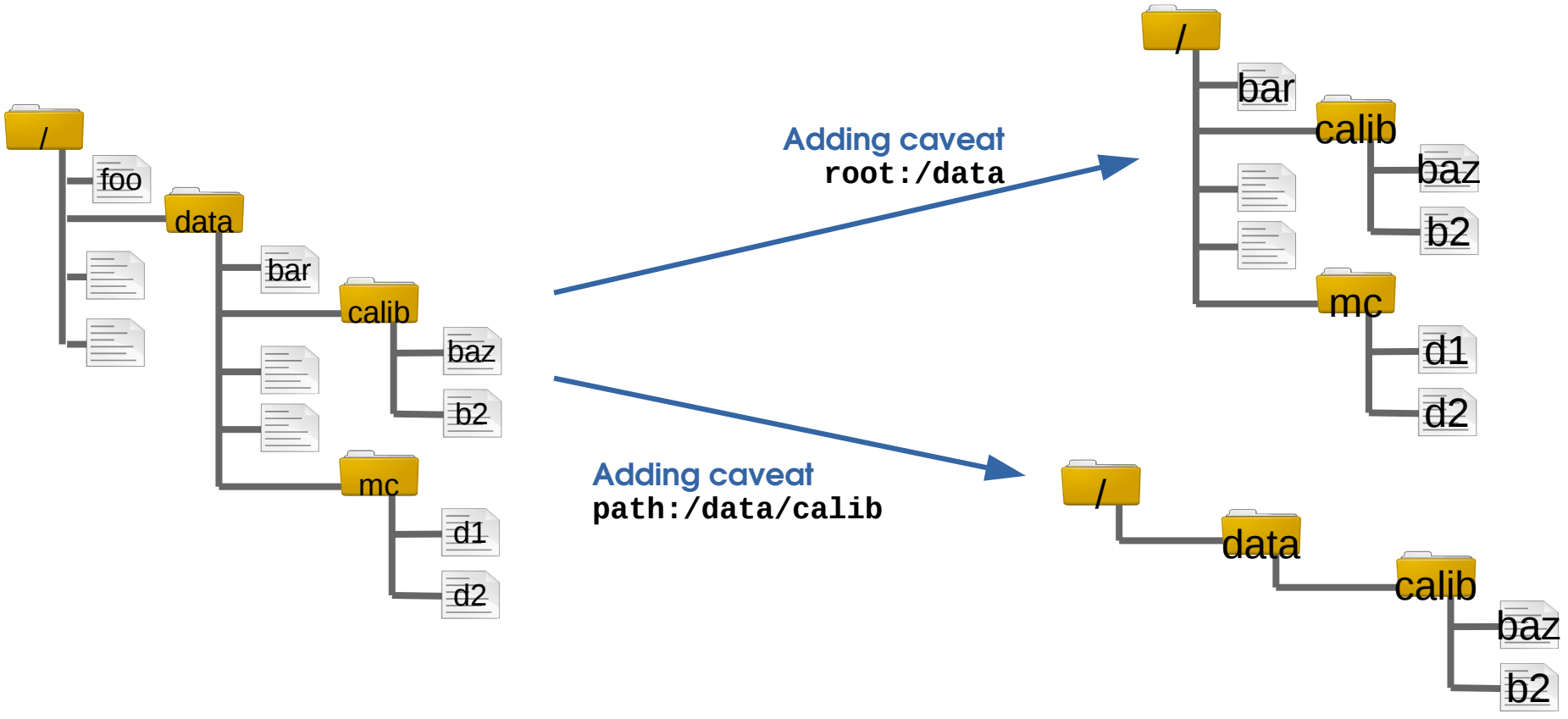


Diagram idea stolen from Tigran

Six caveats supported

- Unfortunately, there are no standard caveats. Here are those that dCache understands:
- Three path caveats:
  - **`root:`**<path> – chroot into this directory,
  - **`home:`**<path> – the user's home directory (not currently used),
  - **`path:`**<path> – only show this path.
- Two context caveats:
  - **`before:`**<timestamp> – when macaroon expires,
  - **`ip:`**<netmask list> – reduce which clients can use macaroon.
- One permissions caveat:
  - **`activity:`**<comma-list> – what operations are allowed.

How path caveats affect namespace



Activity caveats – limited what is allowed

activity:<activity-list>

where **<activity-list>** is a comma-separated list of allowed activities; e.g.,

activity:DOWNLOAD,LIST

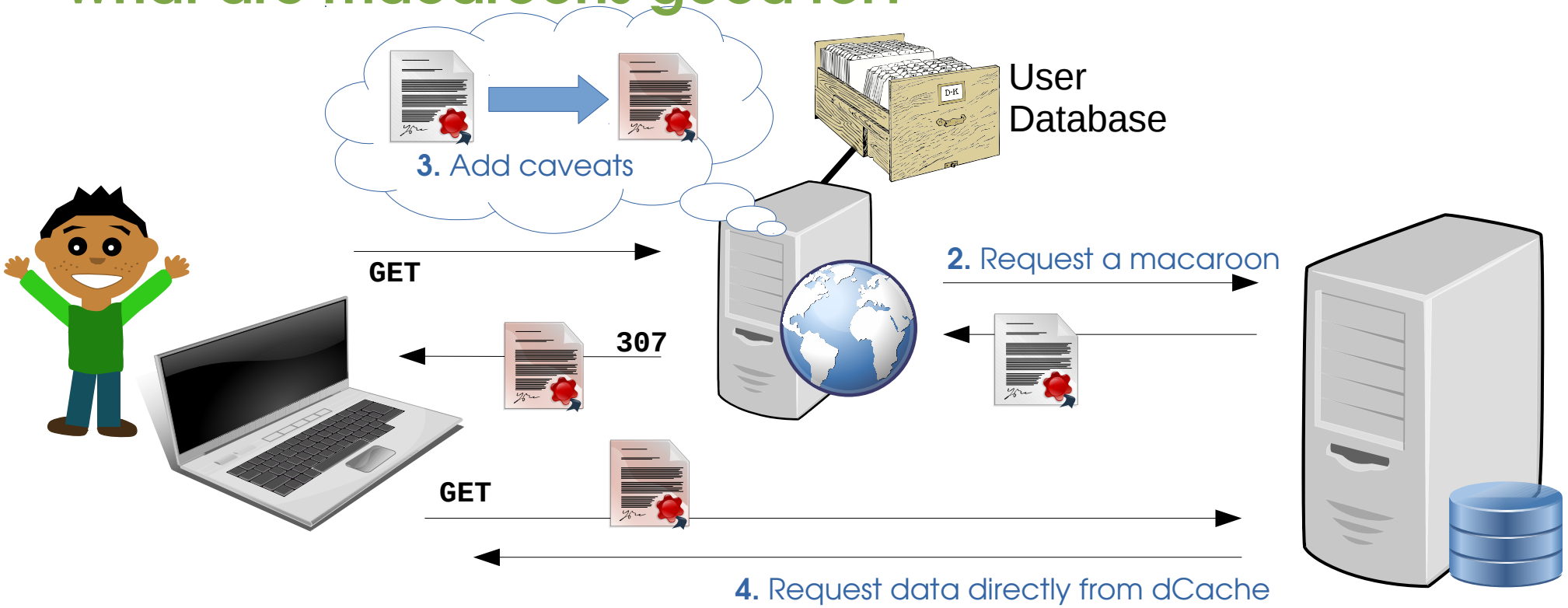
- Possible activities are:

  DOWNLOAD, **UPLOAD**, **DELETE**, **MANAGE**, **LIST**, **READ_METADATA**, **UPDATE_METADATA**.

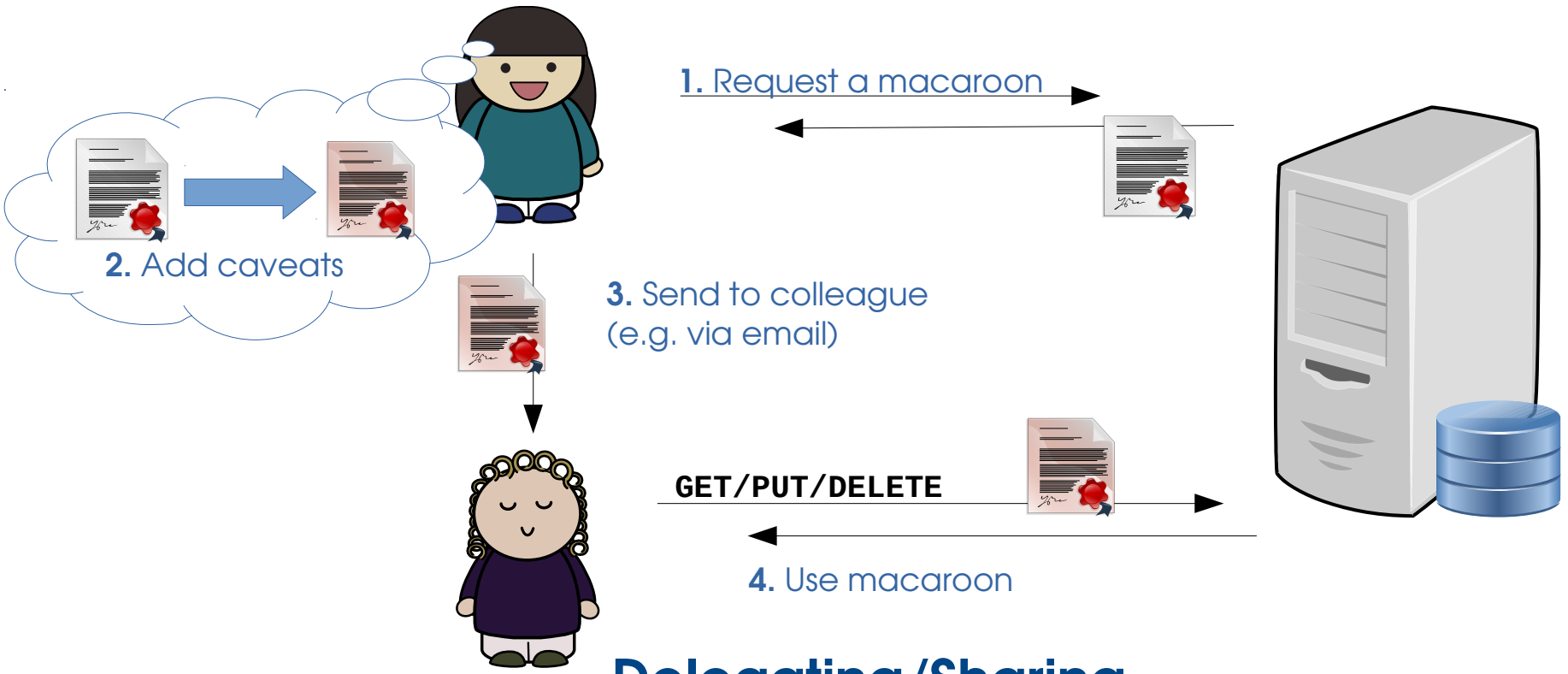
- Allowed activity may be further reduced by adding more **activity:** caveats.

  No **activity:** caveat means client can do whatever the user requesting the macaroon can do.

What are macaroons good for?

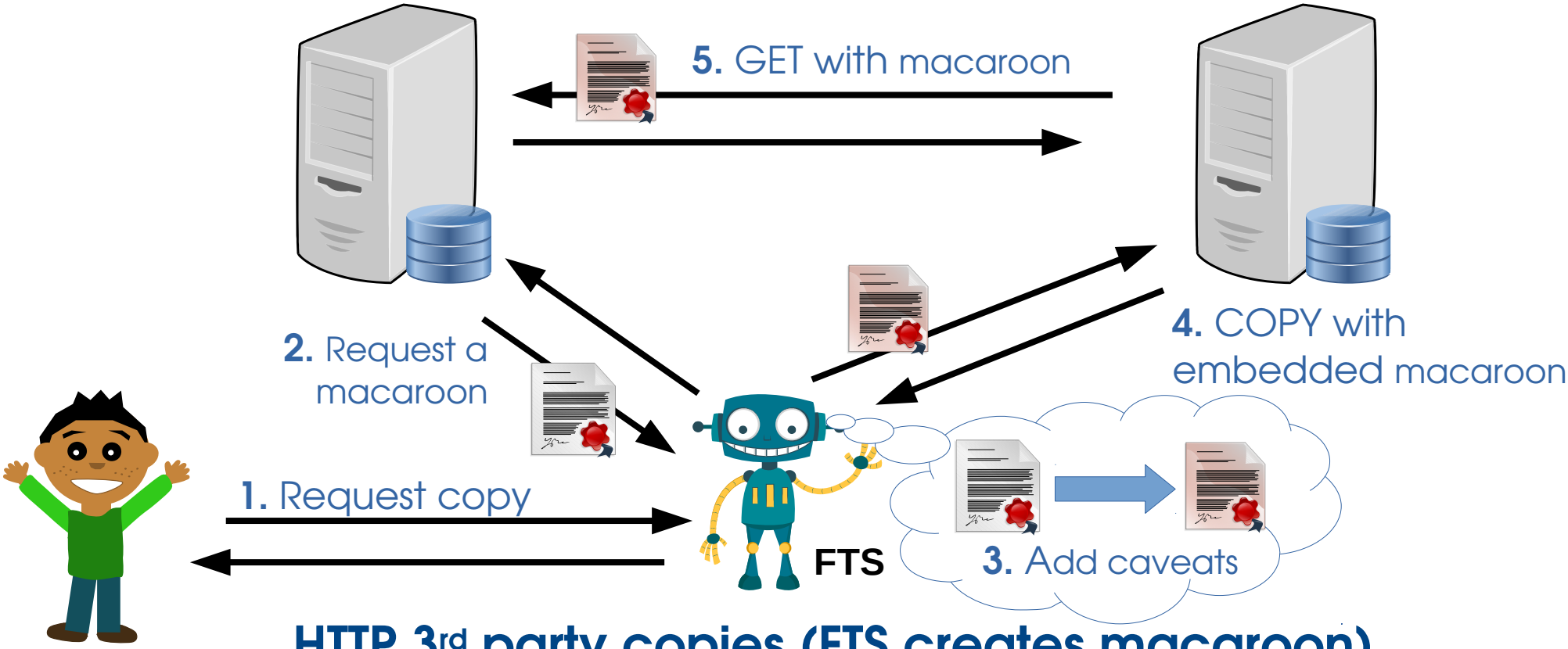


What are macaroons good for?



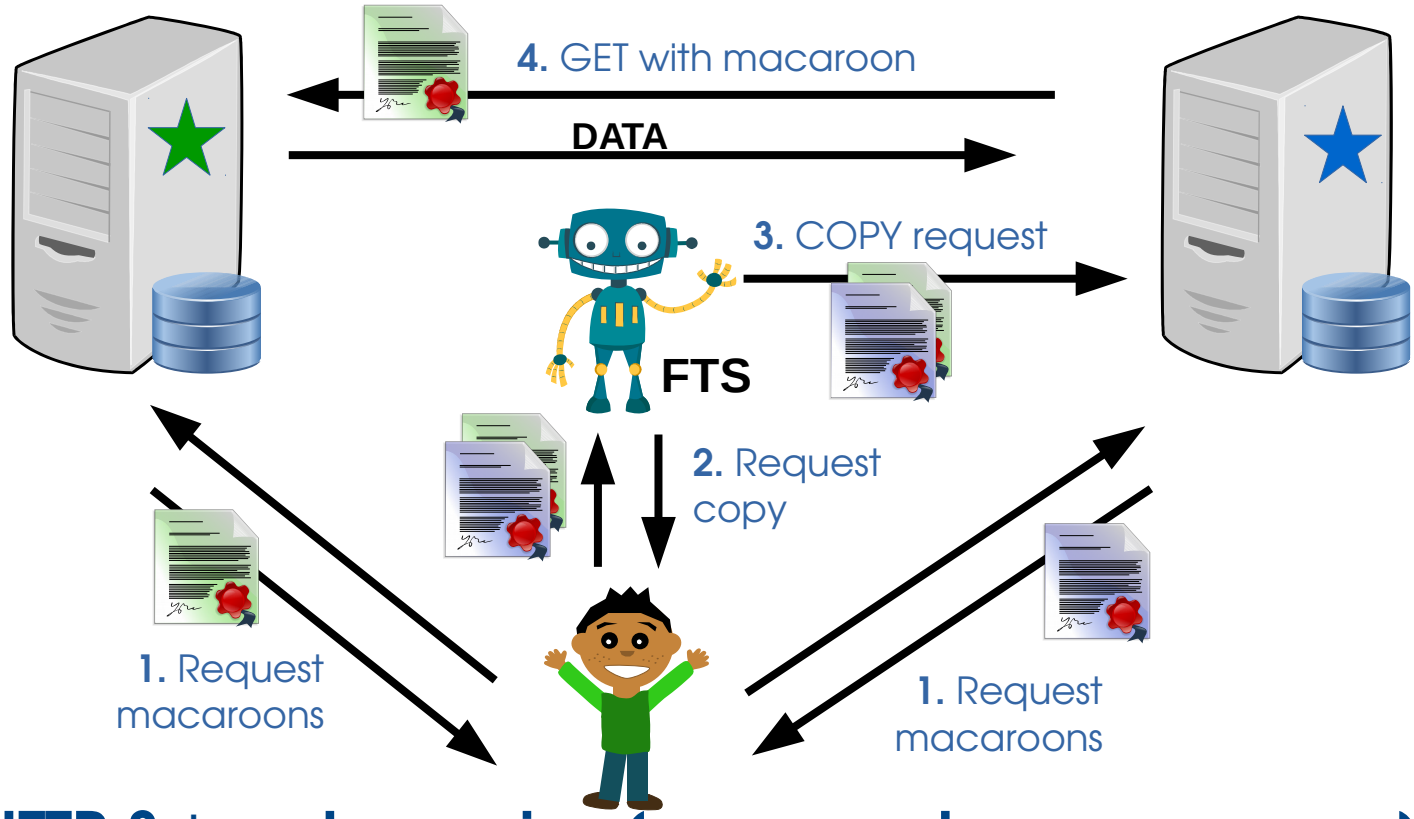
Delegating/Sharing

What are macaroons good for?



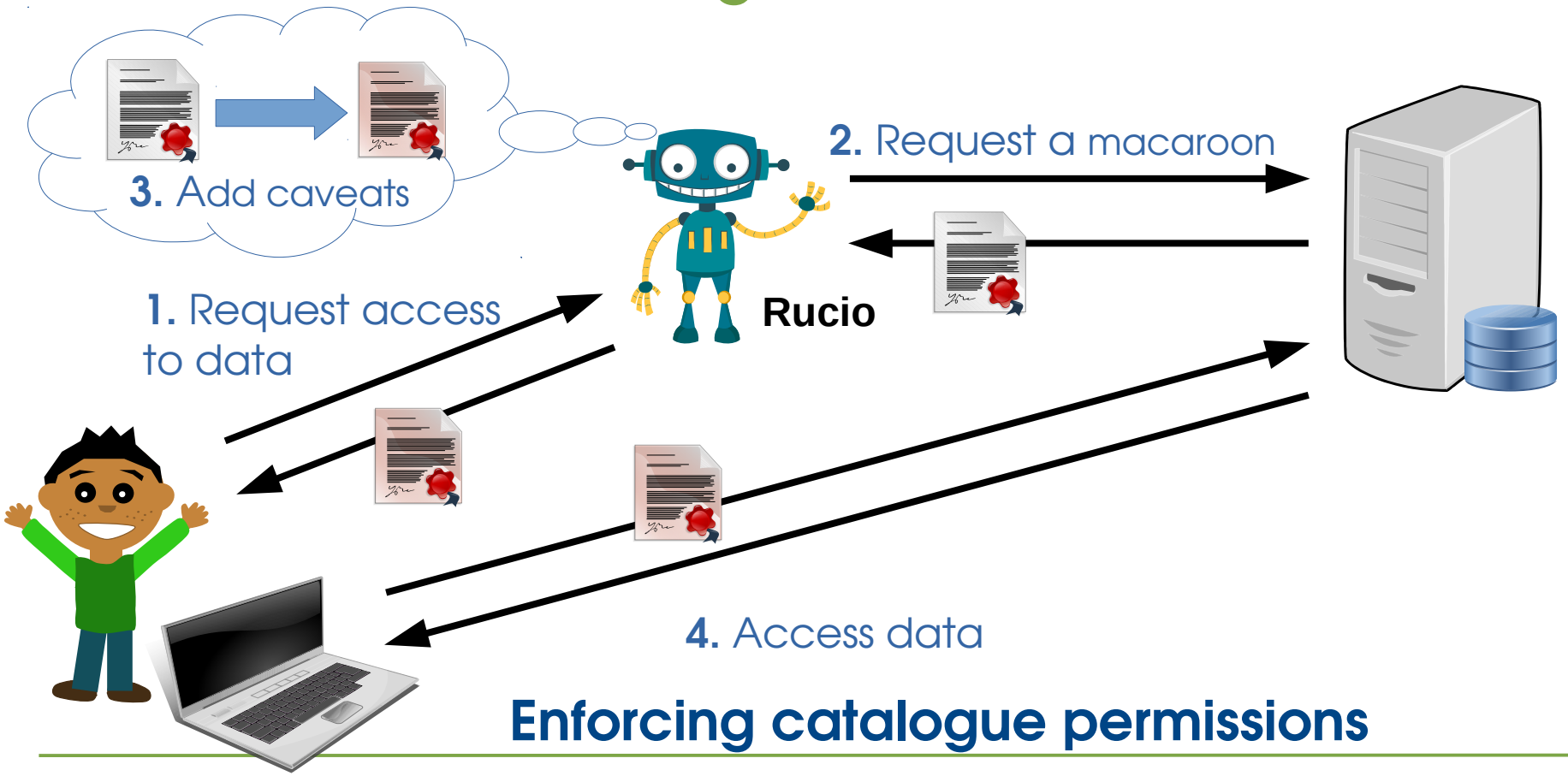
HTTP 3rd party copies (FTS creates macaroon)

What are macaroons good for?



HTTP 3rd party copies (user creates macaroons)

What are macaroons good for?



Enforcing catalogue permissions

Usage of Macaroons

- Nothing yet in production, but …
- SurfSARA have multiple projects exploring macaroons:
  - As **dataset export** for LOFAR (currently separate server)
  - Project MinE: **outsource authz** decision to UMCU (university medical center Utrecht)
  - **Sharing data** without moving it from dCache to ownCloud
  - **Delegated access** to storage; i.e., jobs without X.509 proxy.
- SWESTORE – the **portal use-case**: avoid proxying data transfers.

Current macaroon support in storage systems

- **dCache** fully supported since v3.2
  - available in all supported versions of dCache
- **DPM** experimental support in v1.10
  - currently not recommended in production
- **xrootd** coming soon ("this year")
  - code currently being accepted upstream
- **EOS** not yet, but would add if there's demand
  - would use the xrootd plugin – can investigate once plugin finalised.
- **StoRM** plans to add bearer token authn
  - Initial work focusing on JWT

What's coming next?

- **New features** (in dCache) …
  - ability to cancel subset of macaroons.
  - client identifier caveat.
  - ability to request macaroon outside of WebDAV.
  - support in more protocols (dcap, ftp, …).
- Work with dCache sites to **gain experience**.
- Explore **WLCG use-cases**:

  HTTP 3rd party transfer, …

Summary

- Macaroons provide a solution for **delegated authorisation**.
- Autonomous attenuation means **macaroons scale**.
- Macaroons have **many potential uses**.
- Sites are now **exploring** how to use macaroons.
- **Other storage systems** are exploring macaroons.

Thanks for listening!

Backup slides

Aren't these like SciTokens?

SciTokens vs macaroons: comparison cheat-sheet

- Who issues them?

  (SciToken: "central" service, macaroon: service)

- How expensive to generate?

  (SciToken: a few Hz, macaroon: a few kHz)

- Autonomous reduced token?

  (SciToken: no, macaroon: yes)