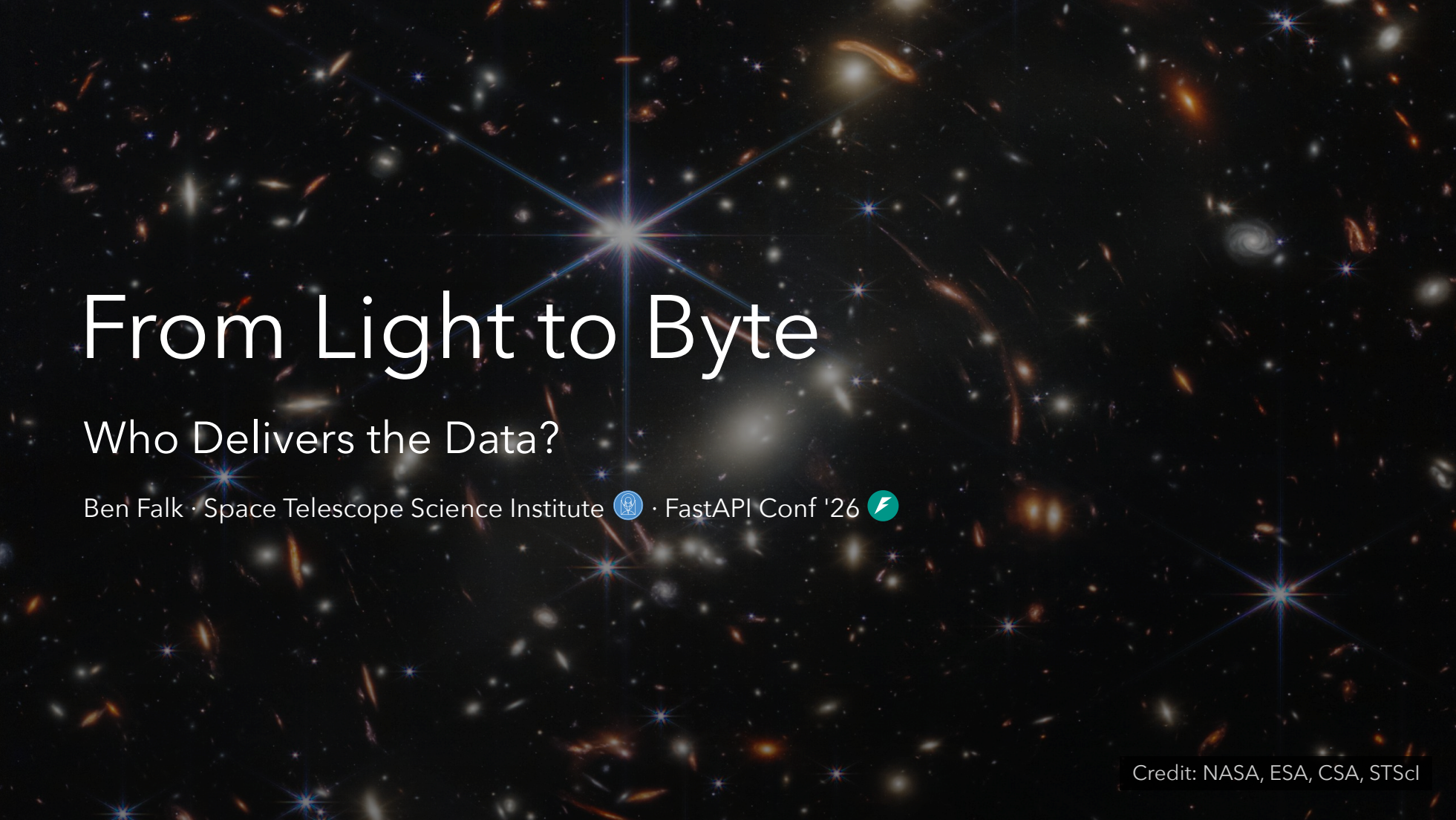


From Light to Byte

Who Delivers the Data?

Ben Falk · Space Telescope Science Institute · FastAPI Conf '26

Credit: NASA, ESA, CSA, STScI

James Webb Space Telescope

Launched in 2021 on Ariane 5 from French Guiana



Credit: NASA/Chris Gunn

ESA-ARIANESPACE / Optique Vidéo du CSG - JM GUILLON

Cosmic Cliffs: 7,600 years of light

One of the earliest images from the James Webb Space Telescope



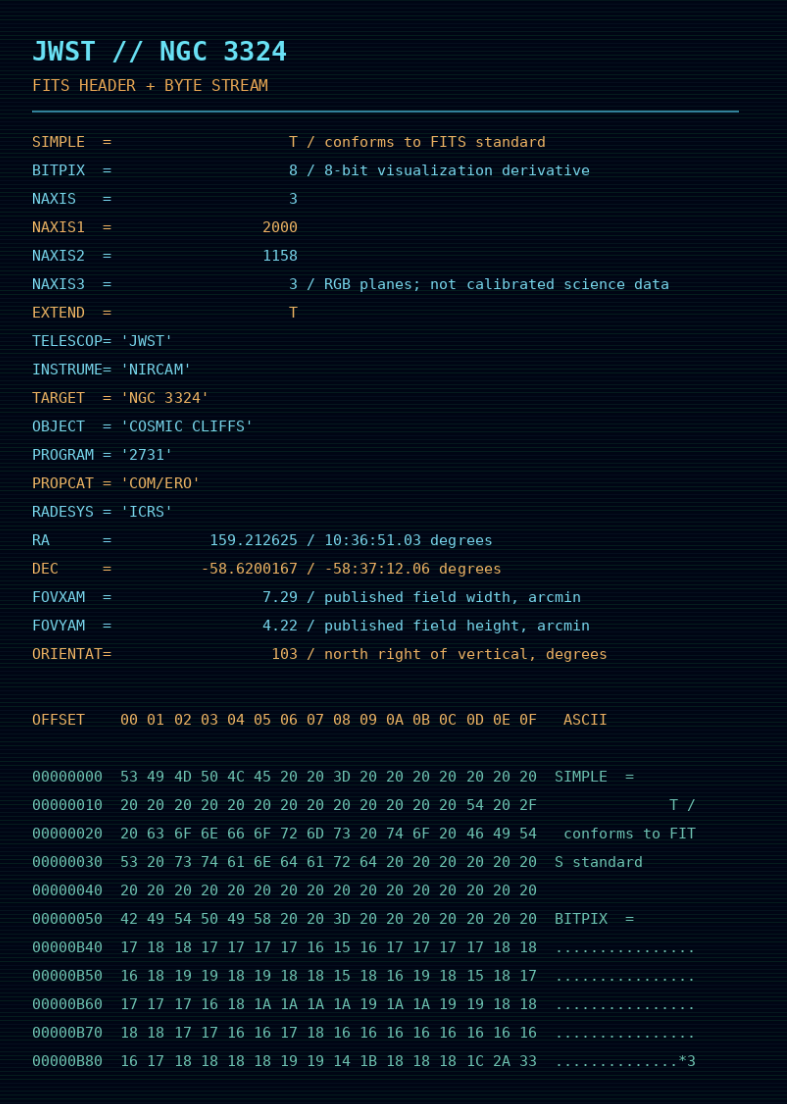
Credit: NASA, ESA, CSA, STScI

Operations

Webb is entering its 5th year of operations. Archive handles data for Hubble, TESS and 20+ other missions.

Data delivered last year

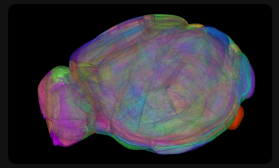
- 2.6 billion file requests
  - 854 million from our own disks
  - 1.8 billion from the cloud
- ~13 PB delivered
  - ~6 PB from our own disks
  - ~7 PB from the cloud
- Some data only on-prem, some only in the cloud
- Every request to the archive goes through FastAPI



About Me



PhD: bat echolocation



Neuroscience Machine Learning Lab, Johns Hopkins

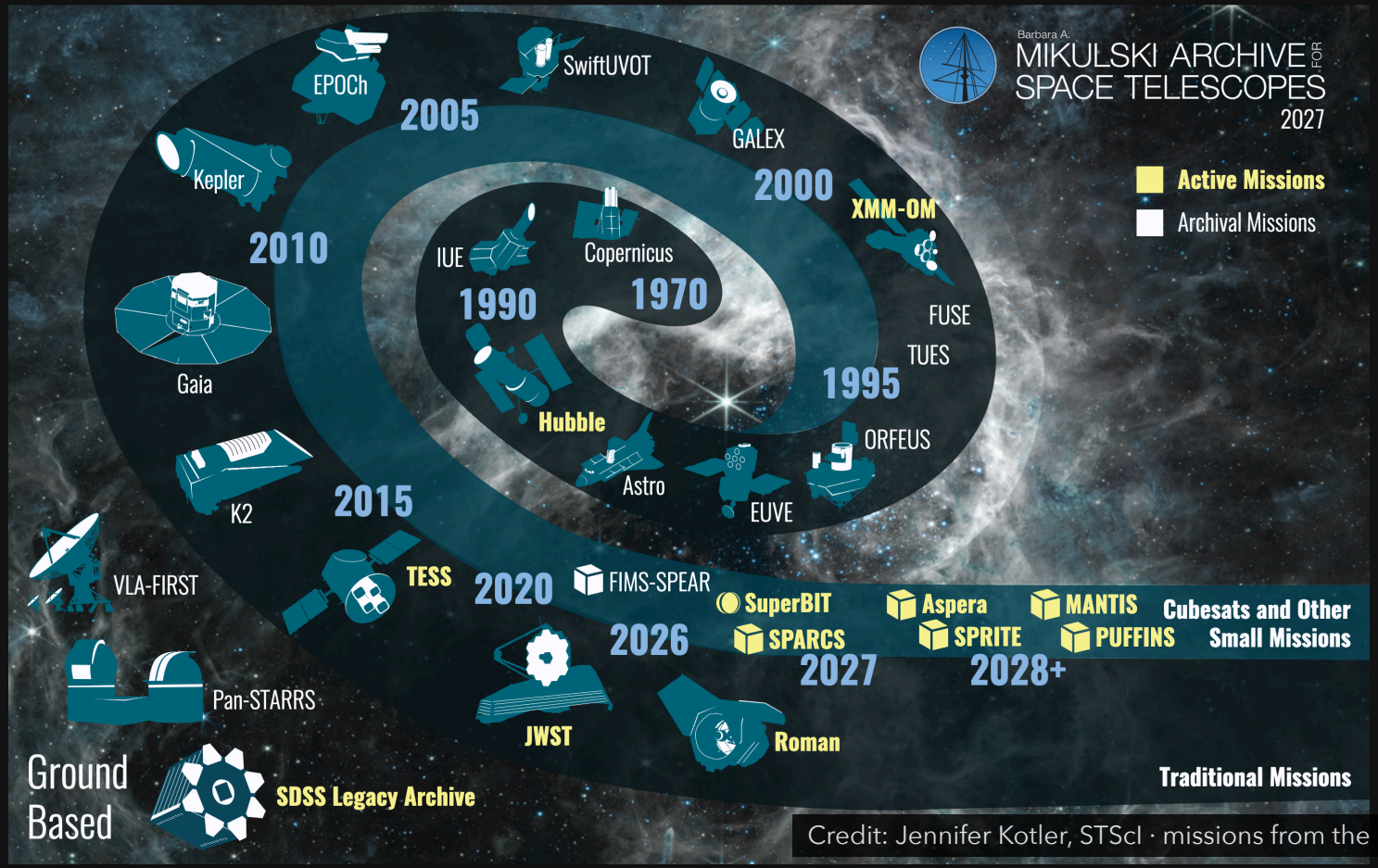


Space Telescope Science Institute (STScI) since 2020

Have been using FastAPI since
from Tornado



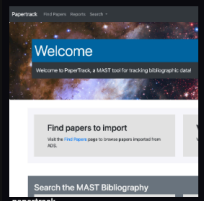
MAST: the archive behind Hubble, Webb and others



Credit: Jennifer Kotler, STScI · missions from the 1970s to today

MAST: the archive behind Hubble, Webb and others

30+ FastAPI services; most traffic is through our API, not browsers



papertrack



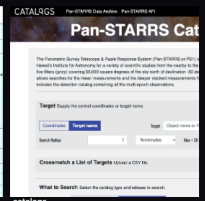
tesscut



schema-browser



subscriptions



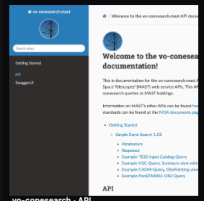
catalogs



missions-search



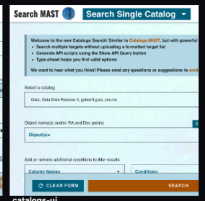
z-mast



vo-conesearch · API



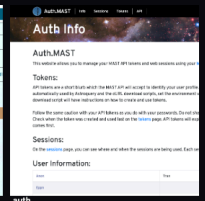
cutouts-service



catalogs-ui



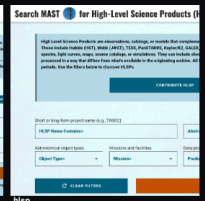
edp



auth



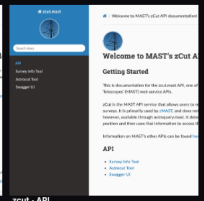
duplication



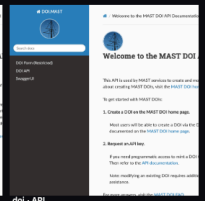
hlsp



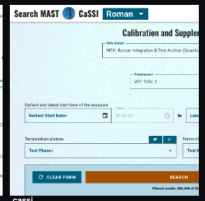
motd · API



zcut · API



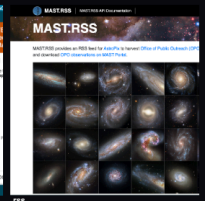
doi · API



cassi



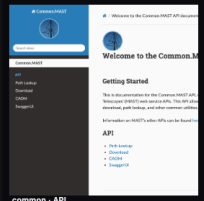
exo



rss



vo-sia



common · API



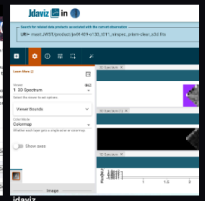
hapcut · API



dochub



vo-tap



jdaviz

Who delivers the data?

	QUESTION
1	How to serve billions of rows?
2	How to serve protected files efficiently?
3	How to egress 7+ PB a year?
4	How to cut 10x10 pixels from a 400 GB file?
5	What if a request takes minutes?
6	How to bundle files that live in multiple places?

Keep the logic. Give away the bytes.



OPTIMIZATION 1 OF 6

How to serve billions of rows?

Over a million requests a month

1.8 billion stars from ESA's Gaia. Every one is a row someone can query.

Catalog search: SQL in, rows out

- Sky catalogs have billions of rows: stars, galaxies, spectral measurements
- A user asks for 100,000 of them
- We receive over a million requests a month
- Can’t build each one into a 100,000-row list

Generators to the rescue

```
def fetch_rows(conn, query):
    cursor = conn.execute(query)
    while rows := cursor.fetchmany(1000):
        yield from rows

@app.get("/catalogs/{catalog}/search")
def search(query: str, resp_format: str, conn=Depends(get_conn)):
    row_gen = fetch_rows(conn, query)
    stream, media_type = format_rows(row_gen, resp_format)  # csv, xml, or json
    return StreamingResponse(stream, media_type=media_type)
```

- 1,000 rows at a time from the database, streamed directly
- Memory stays flat
- Formatted as `CSV`, `XML`, or `JSON` based on the user's request

Quiz: Which keeps serving 100 users at once?

A `def`

```python
@app.get("/search")
def search(query: str):
    return run_query(query)  # runs in thread
```

B `async def`

```python
@app.get("/search")
async def search(query: str):
    return run_query(query)  # blocks
```

- **A.** Our MS SQL Server driver (pyodbc) blocks, and there's no mature async driver
- `async def` runs on the event loop: one slow query stalls every request
- Whereas a plain `def` runs in FastAPI's threadpool
- Even streaming doesn't block, as Starlette iterates a sync generator in a threadpool

Gotcha: StreamingResponse with an active connection?

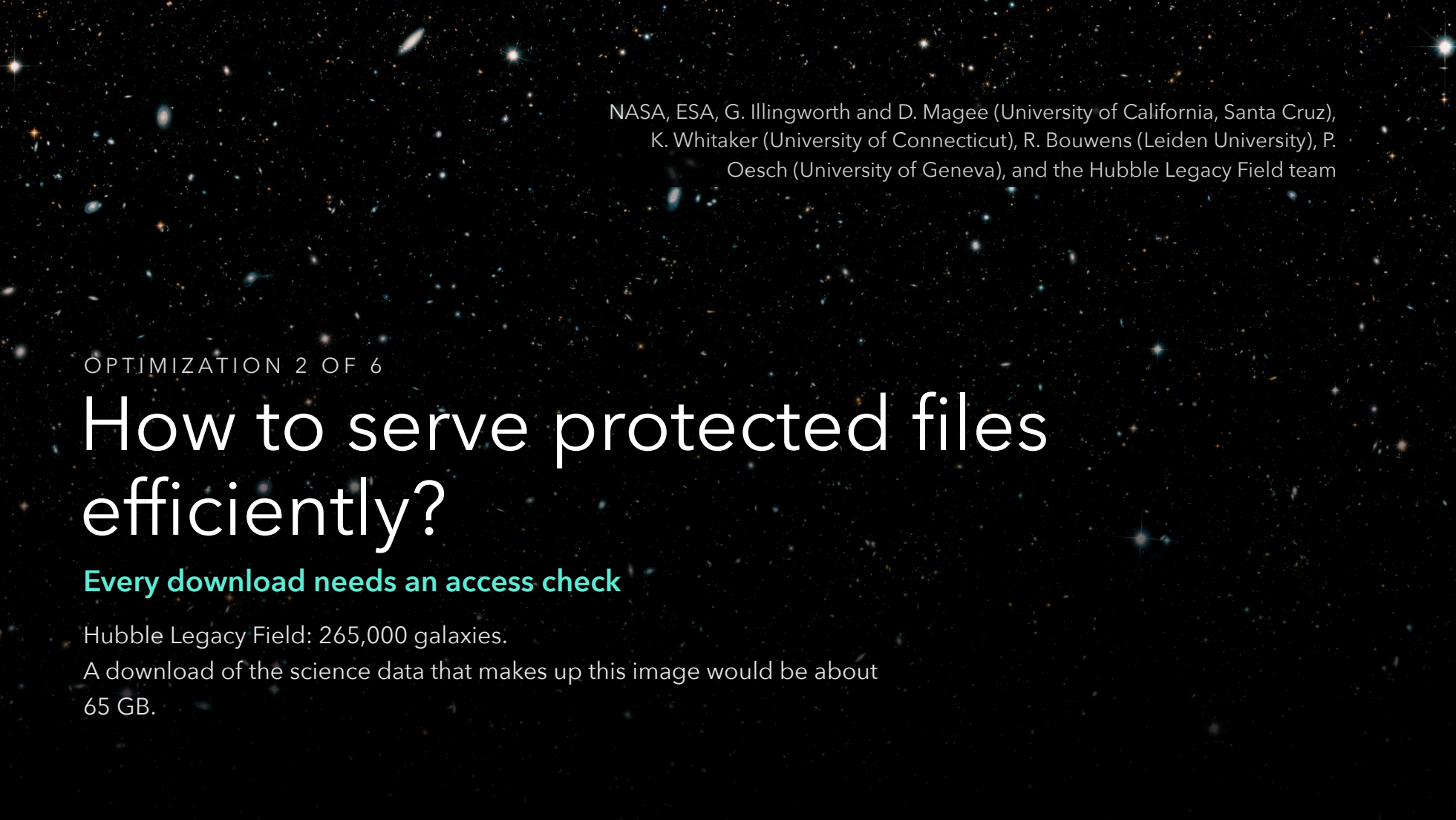
```python
# Works: a Dependency using `yield` closes after the response is sent
def get_conn():
    with connect() as conn:
        yield conn

@app.get("/search")
def search(query: str, conn=Depends(get_conn)):
    return StreamingResponse(fetch_rows(conn, query))
```

- The endpoint *returns* the stream. Streaming happens afterward.
- A `yield` dependency cleans up after the response finishes

Heavy lifting: FastAPI

FastAPI 0.106 to 0.117 cleaned up `yield` dependencies before the response; fixed in 0.118.



OPTIMIZATION 2 OF 6

How to serve protected files efficiently?

Every download needs an access check

Hubble Legacy Field: 265,000 galaxies.
A download of the science data that makes up this image would be about 65 GB.

The download endpoint

```python
@app.get("/download")
def download(file_id: str, request: Request, user=Depends(current_user)):
    file = lookup(file_id)  # lookup file location and access privileges
    if not file.readable_by(user):
        raise HTTPException(403)
    if "X-Accel-Enable" in request.headers:  # set by nginx
        return Response(headers={"X-Accel-Redirect": file.internal_path})
    return FileResponse(file.local_path)  # local dev and tests
```

FastAPI answers with an empty body and one header. nginx sends the file.

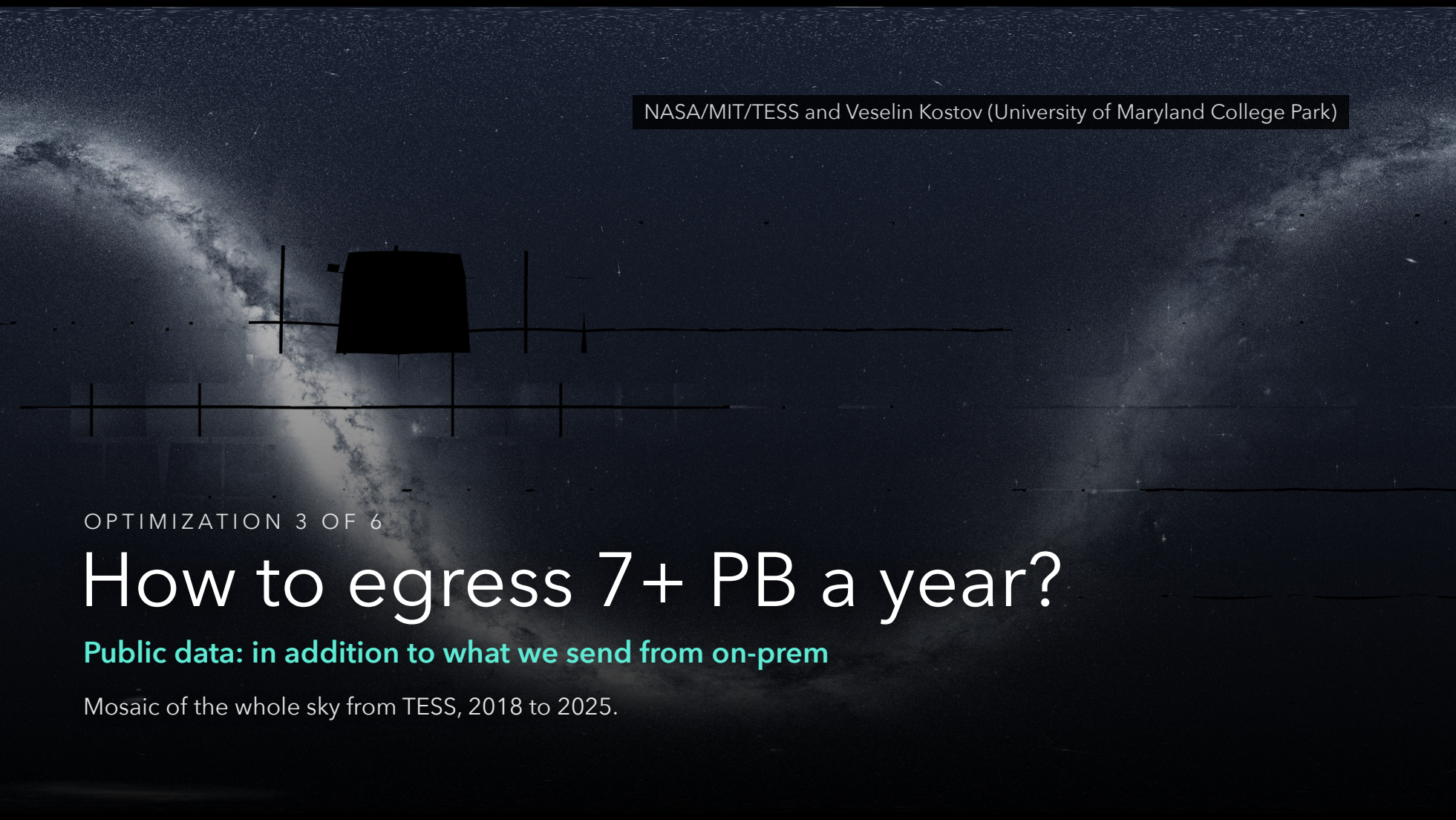
nginx does the copying

```
location /download {
    proxy_pass http://app;
    proxy_set_header X-Accel-Enable True;  # enable nginx to send the file
}

location /files/ {
    internal;  # unreachable from outside
    alias /mnt/archive/;  # on-prem location
}
```

- Keep the logic in Python: access checks, lookup, logging
- nginx copies with `sendfile`: Range requests, resumed downloads, slow clients all work
- **854M downloads, ~6 PB a year** from on-prem
- Works for any app behind nginx

Heavy lifting: nginx



OPTIMIZATION 3 OF 6

How to egress 7+ PB a year?

Public data: in addition to what we send from on-prem

Mosaic of the whole sky from TESS, 2018 to 2025.

Who pays for the bytes?

Same endpoint, with a redirect:

```python
@app.get("/download")
def download(file_id: str, request: Request, user=Depends(current_user)):
    file = lookup(file_id)
    if not file.readable_by(user):
        raise HTTPException(403)
    if file.public_url:  # redirect to AWS Open Data
        return RedirectResponse(file.public_url, status_code=302)
    if "X-Accel-Enable" in request.headers:
        return Response(headers={"X-Accel-Redirect": file.internal_path})
    return FileResponse(file.local_path)
```

Or let AWS cover it: public files are mirrored to the AWS Open Data bucket.

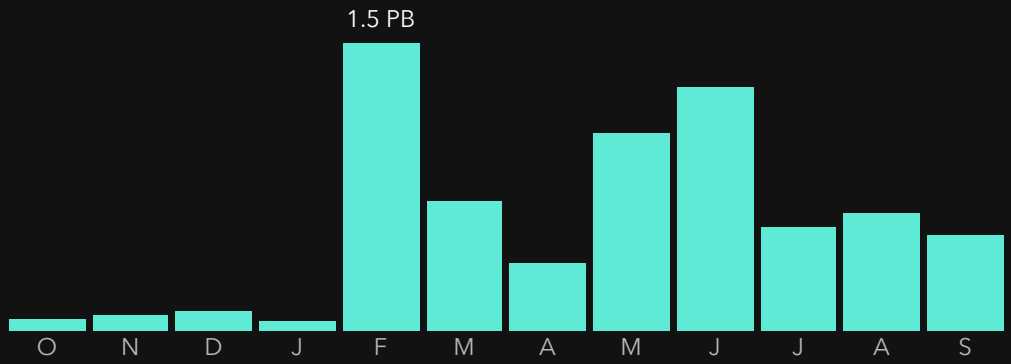
```
GET /download?file_id=tess2018206045859-s0001-0000000206409997-0120-s_lc.fits
HTTP/1.1 302 Found
Location: https://stpubdata.s3.amazonaws.com/tess/public/tid/s0001/ … / … -s_lc.fits
```

AWS covers the cost

~7 PB

out of the Open Data Bucket last year

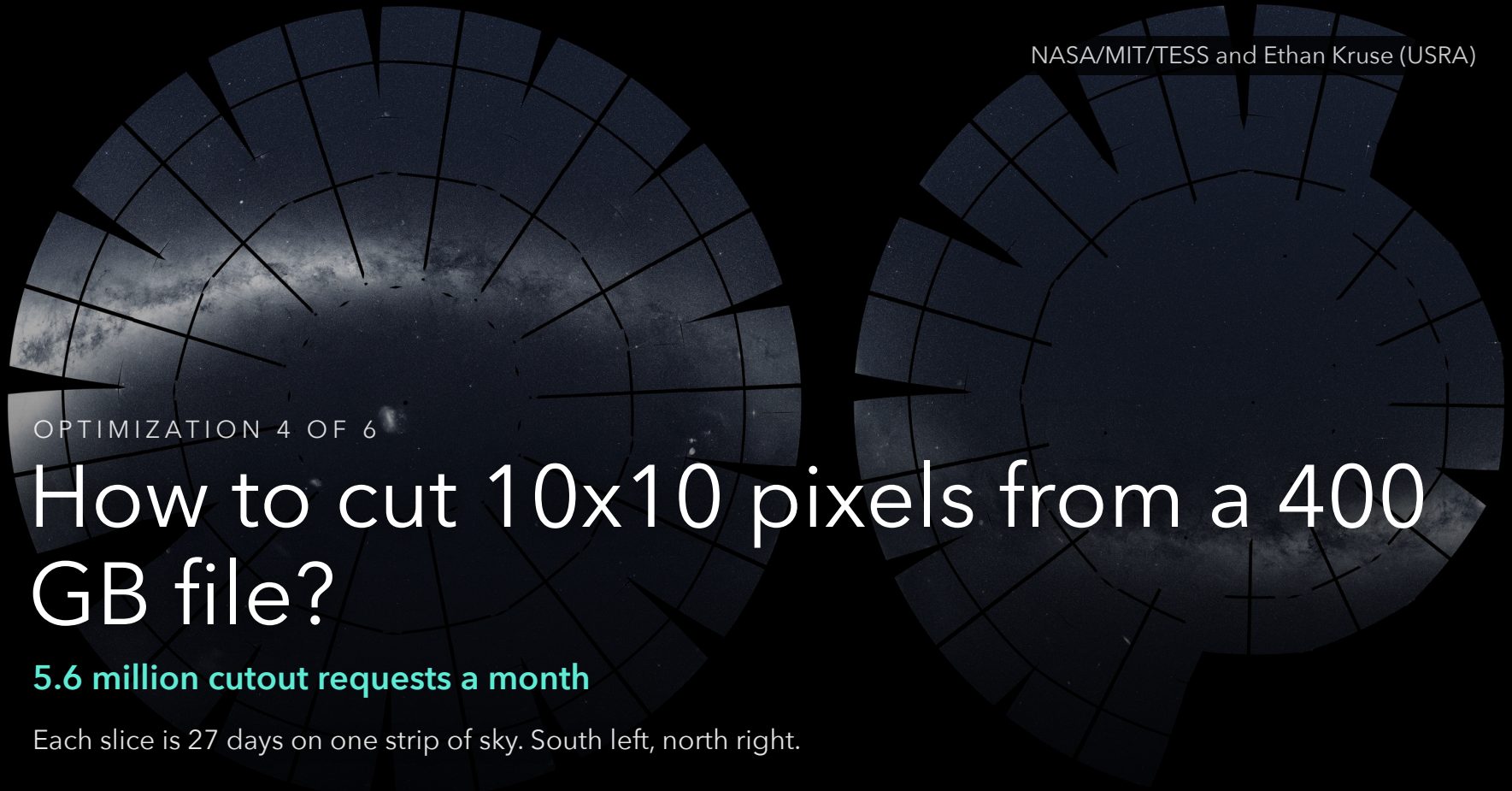
1.8 billion requests



Egress per month, Oct 2025 to Sep 2026

- **~$350K** a year we don't pay
- Grateful to the Open Data Sponsorship Program
- A cheeky way out: we didn't solve egress, we found a sponsor
- Egress is often the biggest bill. Get creative:
  - ask for sponsorship
  - compute next to the data
  - send fewer bytes (next)

Heavy lifting: AWS S3



NASA/MIT/TESS and Ethan Kruse (USRA)

OPTIMIZATION 4 OF 6

How to cut 10x10 pixels from a 400 GB file?

5.6 million cutout requests a month

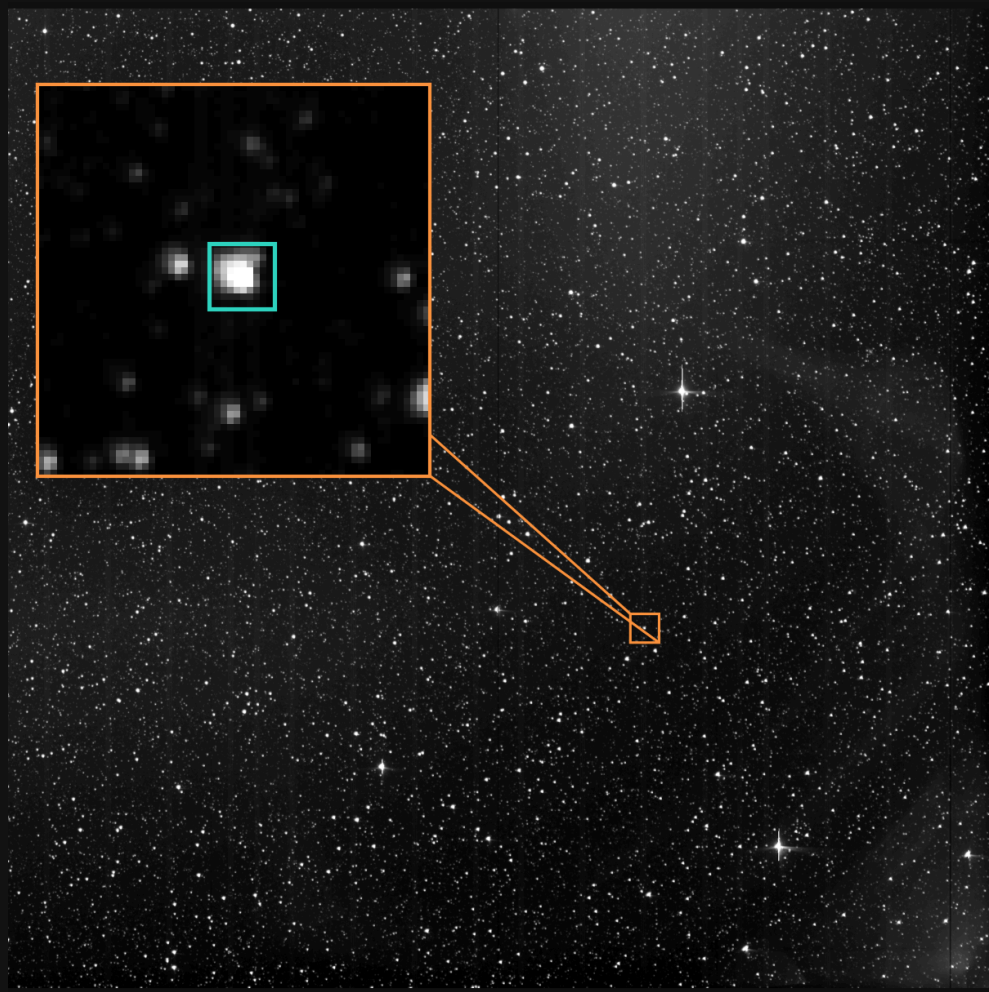
Each slice is 27 days on one strip of sky. South left, north right.

A 385 GB file, a 10×10 pixel request

- TESS photographs the same patch of sky every 200 seconds for ~27 days
- Each camera sensor's images are stacked into one **cube**: up to 418 GB per file
- Users want a cutout around one star, through time:

```
Tesscut.get_cutouts(
    objectname="WASP-18",
    size=10,
)
```

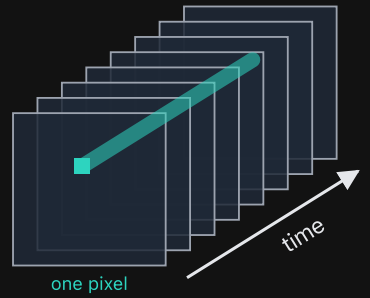
- Download the cube for every request?
- Keep it on a local disk?



One TESS full-frame image: 2048 × 2048 pixels
Teal box: the 10×10 pixel request around WASP-18

Cube format: Store it with time on the "fastest" axis

- On disk: row, column, **time**, value/error
- One pixel's whole history is ~87 KB in a row
- So each row of the stamp is one contiguous byte range

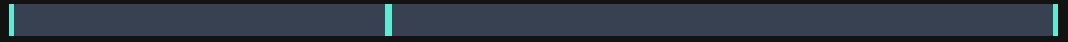


What goes over the wire

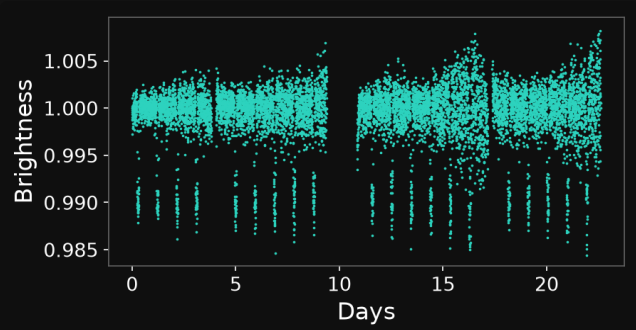
```
GET /tess/public/mast/tess-s0069-2-2-cube.fits HTTP/1.1
Host: stpubdata.s3.amazonaws.com
Range: bytes=138494521920-138495400319

HTTP/1.1 206 Partial Content
Content-Range: bytes 138494521920-138495400319/385462851840
Content-Length: 878400
```

- WASP-18, 10×10: **16 requests, 23 MB of 385 GB** (0.006%)
- Then zip, upload to S3, and `303 See Other` (more in a minute)
- 5.6M requests a month, a few seconds each
- Lay data out for how it's read: Range makes S3 a random-access disk

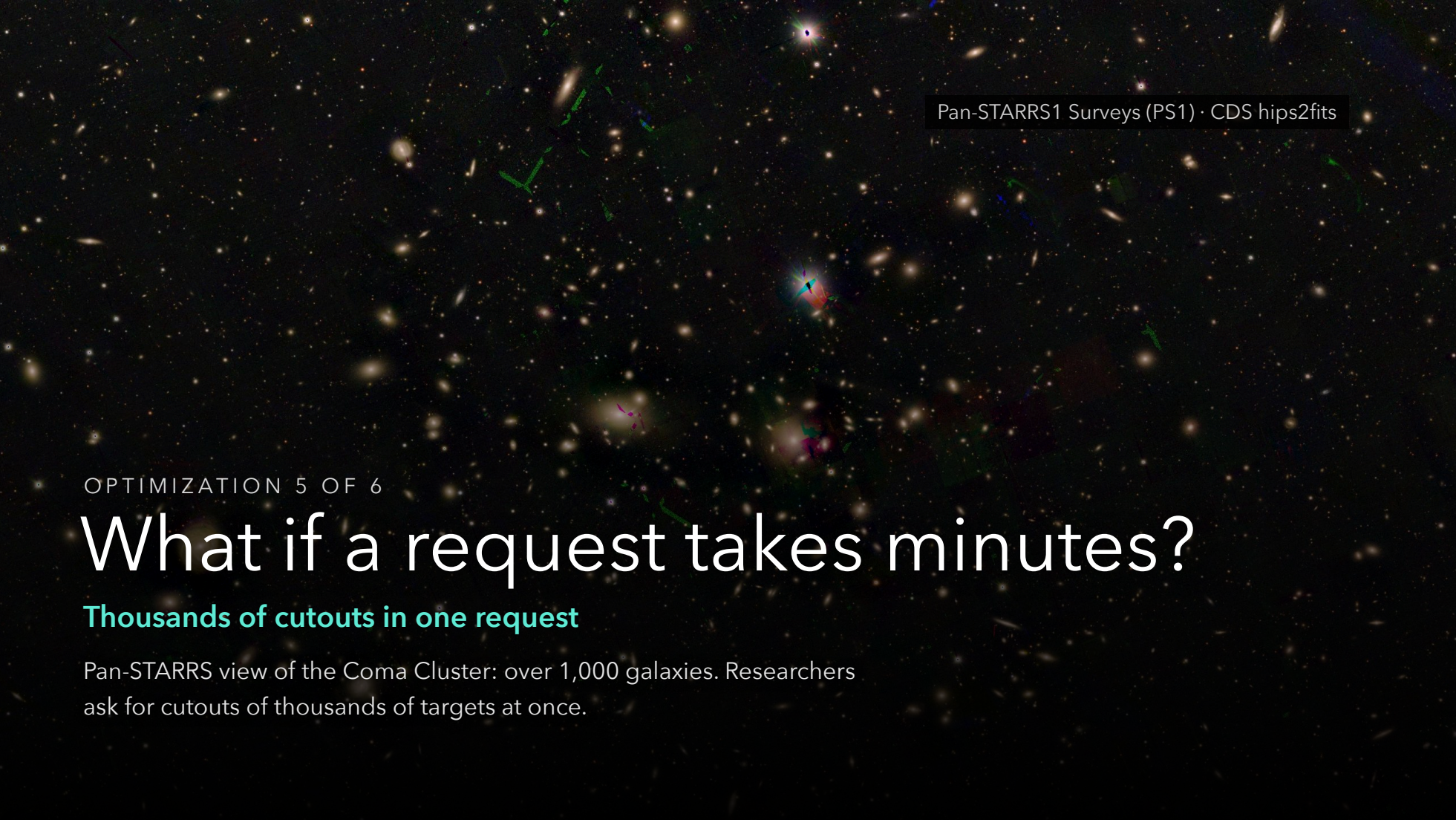


The 385 GB file. Teal: the bytes actually read (header, 10 rows, metadata).



The cutout's brightness over time. Each dip is a planet crossing its star.

Heavy lifting: HTTP Range reads



OPTIMIZATION 5 OF 6

What if a request takes minutes?

Thousands of cutouts in one request

Pan-STARRS view of the Coma Cluster: over 1,000 galaxies. Researchers ask for cutouts of thousands of targets at once.

Minutes of work in one request?

- **Seconds:** do it in the request. **Minutes, or thousands of items:** return a job URL.
- Redis holds the queue, the job state, and each user's quota.
- Celery workers (Python, on AWS Fargate) make the cutouts.
- Results land in S3 as signed links that expire.

```
POST /cutouts/async            → 303 See Other, Location: /cutouts/async/3f9a2c1e
GET  /cutouts/async/3f9a2c1e   → {"phase": "EXECUTING"}
GET  /cutouts/async/3f9a2c1e   → {"phase": "COMPLETED", "results": ["https:// … "]}
GET  https://<bucket>.s3.amazonaws.com/ … /cutout.fits?X-Amz-Expires=3600& …
```

Create the job

```python
@router.post("/cutouts/async")
async def create_job(
    request: Request,
    params: CutoutRequest,
    user: User = Depends(current_user),
):
    job_id = uuid.uuid4().hex[:8]
    try:
        await limiter.reserve(
            user, job_id, len(params.positions)
        )
    except LimitExceeded as exc:
        raise HTTPException(
            429, headers={"Retry-After": str(exc.retry_after)}
        )
    await enqueue(run_job, job_id=job_id)
    return RedirectResponse(
        f"{request.url.path}/{job_id}", status_code=303
    )
```

- Reserve the quota in Redis before queuing any work. Over the limit: `429` with `Retry-After`.
- Queue the job in Celery.
- `303 See Other`: the POST becomes a GET on the job URL. `RedirectResponse` defaults to 307, which would re-POST.

Heavy lifting: Celery workers



OPTIMIZATION 6 OF 6

How to bundle files that live in multiple places?

Some only on prem, some only in AWS

Webb's Stephan's Quintet, built from almost 1,000 separate image files.

What’s a bundle?

- On our search web page, a user selects 200 files
- Clicks "Download bundle"
- Gets one zip: the files plus `MANIFEST.HTML`.
- One bundle can mix files from multiple sources (cloud and on-prem)

Manifest for File: MAST_2026-10-25T2345

Total Files: 63 of 63

URI	File	Access	Status	Logged In User
mast:HST/IDK804010/idk804010_asn.fits	MAST_2026-10-25T2345/HST/idk804010_asn.fits	PUBLIC	OK	anonymous
mast:HST/IDK804010/idk804010_drc.fits	MAST_2026-10-25T2345/HST/idk804010_drc.fits	PUBLIC	OK	anonymous
mast:HST/IDK804010/idk804010_drz.fits	MAST_2026-10-25T2345/HST/idk804010_drz.fits	PUBLIC	OK	anonymous
mast:HST/IDK804010/idk804010_drc.jpg	MAST_2026-10-25T2345/HST/idk804010_drc.jpg	PUBLIC	OK	anonymous
mast:HST/IDK804010/idk804010_drz.jpg	MAST_2026-10-25T2345/HST/idk804010_drz.jpg	PUBLIC	OK	anonymous
mast:HST/IDK804010/idk804sgq_flt.fits	MAST_2026-10-25T2345/HST/idk804sgq_flt.fits	PUBLIC	OK	anonymous
mast:HST/IDK804010/idk804sgq_flc.fits	MAST_2026-10-25T2345/HST/idk804sgq_flc.fits	PUBLIC	OK	anonymous
mast:HST/IDK804010/idk804sgq_flc.jpg	MAST_2026-10-25T2345/HST/idk804sgq_flc.jpg	PUBLIC	OK	anonymous
mast:HST/IDK804010/idk804sgq_flt.jpg	MAST_2026-10-25T2345/HST/idk804sgq_flt.jpg	PUBLIC	OK	anonymous
mast:HST/IDK804010/idk804shq_flc.fits	MAST_2026-10-25T2345/HST/idk804shq_flc.fits	PUBLIC	OK	anonymous
mast:HST/IDK804010/idk804shq_flt.fits	MAST_2026-10-25T2345/HST/idk804shq_flt.fits	PUBLIC	OK	anonymous
mast:HST/IDK804010/idk804shq_flt.jpg	MAST_2026-10-25T2345/HST/idk804shq_flt.jpg	PUBLIC	OK	anonymous
mast:HST/IDK804010/idk804shq_flc.jpg	MAST_2026-10-25T2345/HST/idk804shq_flc.jpg	PUBLIC	OK	anonymous
mast:HST/IDK804010/idk804siq_flt.fits	MAST_2026-10-25T2345/HST/idk804siq_flt.fits	PUBLIC	OK	anonymous
mast:HST/IDK804010/idk804siq_flc.fits	MAST_2026-10-25T2345/HST/idk804siq_flc.fits	PUBLIC	OK	anonymous
mast:HST/IDK804010/idk804siq_flc.jpg	MAST_2026-10-25T2345/HST/idk804siq_flc.jpg	PUBLIC	OK	anonymous
mast:HST/IDK804010/idk804siq_flt.jpg	MAST_2026-10-25T2345/HST/idk804siq_flt.jpg	PUBLIC	OK	anonymous

Who builds the zip?

```
def stream_zip(bundle):
    yield b"PK\x03\x04\x0a\x00\x00\x00"  # first zip bytes, right away
    bundle.download_remote_files()  # AWS → our server, just to zip it
    yield from run(["zip", "-r", "-q", "-1", "-", bundle.dir])
```

- Zip on-prem: pull AWS files through us.
- Zip in AWS: push our files up.
- Either way we proxy one side.

Who actually uses bundles?

99.8%

not from API clients

The browser can reach both places. Let it build the zip.

The browser builds the zip

```
const opfs = await navigator.storage.getDirectory()  // browser-managed disk
await runWithConcurrency(4, files.map((f) => async () => {
  const handle = await opfs.getFileHandle(f.name, { create: true })
  await (await fetch(f.url)).body.pipeTo(await handle.createWritable())
}))
const zip = new ZipWriter(output, { compressionMethod: 0 })  // store, don't compress
for (const f of files) {
  const file = await (await opfs.getFileHandle(f.name)).getFile()
  await zip.add(f.path, file.stream())
}
await zip.close()
```

- Each `fetch` goes straight to the file's home: 302 to AWS, or nginx from our disks
- No bytes cross between on-prem and cloud
- OPFS (Origin Private File System): multi-GB bundles without the browser running out of memory

Heavy lifting: the user's browser

Who delivers the data?

	OPTIMIZATION	FASTAPI	HEAVY LIFTING BY
1	**Stream, don't buffer**	Runs the query, yields rows	FastAPI
2	**Let nginx send the file**	Checks access, finds the file	nginx
3	**Redirect to the cloud**	Looks up the public URL	AWS S3
4	**Read only the bytes you need**	Cuts with astropy range reads	HTTP Range reads
5	**Queue the work**	Creates a job, returns its URL	Celery workers
6	**Let the browser build the zip**	Points to each file's home	The user's browser

Every request to the archive goes through FastAPI.

Keep the logic. Give away the bytes.

Keep the logic. Give away the bytes.

1. **Plain `def` for blocking drivers.** The threadpool is a feature, not a fallback.
2. **A `yield` dependency outlives the stream.** Open connections there, not in a `with`.
3. **`RedirectResponse` defaults to 307.** Use 303 after a POST.
4. **Measure who calls you.** 0.2% of bundle requests came from APIs.

NASA/Joel Kowsky

NEXT

Nancy Grace Roman Space Telescope

~20 PB over five years. Same six paths.

Roman launched 30 Aug 2026.



Nancy Grace Roman

Keep the logic.
Give away the bytes.

Thank you

Ben Falk

Space Telescope Science Institute

bfalk@stsci.edu

@ben-falk.bsky.social

LINKS

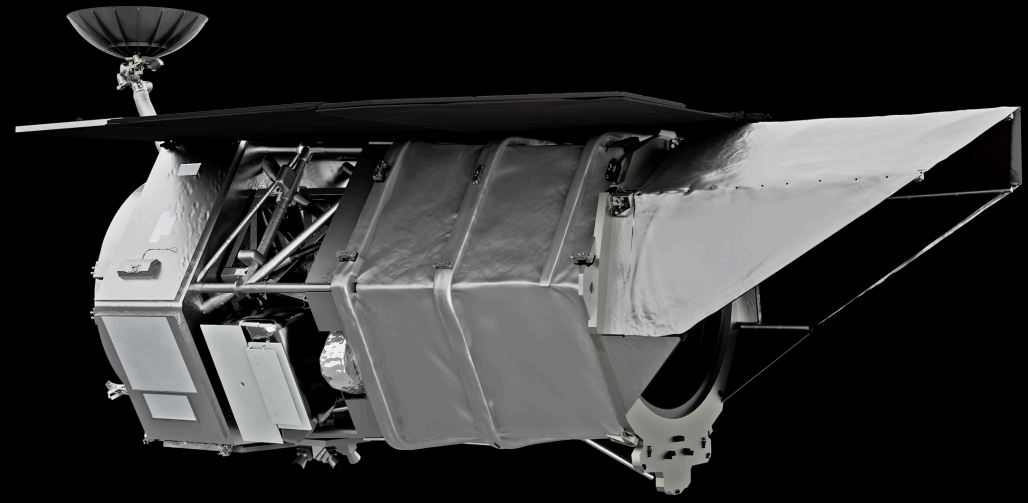
archive.stsci.edu · the archive

github.com/orgs/spacetelescope/

registry.opendata.aws · search "STScI"

github.com/nasa-fornax/fornax-cutouts

docs.astropy.org · FITS subsets from the cloud



Credit: NASA's Goddard Space Flight Center/Conceptual Image Lab