Measuring the Growth of Your Open Source AI Project

Avi Press

April 16, 2024

Outline

Outline

Quick notes

- These slides are generated from an org-mode document which is available here:
  `https://github.com/aviaviavi/talks/blob/master/beyond-the-download-count/presentation-04-2024-ai.org`
- I'm doing a talk on Thursday that goes deeper into these topics, join me!

Outline

Stats from registries are limited

Docker Hub

- Total downloads for repo

Stats from registries are limited

Docker Hub

- Total downloads for repo

Others

GitHub Packages & Containers

- Total downloads for repo
- Total downloads by tag

PyPI

- Total downloads over time
- Downloads by version over time
- Downloads by mirror vs not mirror over time

What else might we want to understand?

Metrics

- Unique downloads (10 downloads from 10 people vs 10 downloads from the same person)
- Downloads by:
  - Host platform
  - Client
  - Country
  - Architecture
  - Companies
- Invocations versus downloads of the container?

Granularity

- Should be able to understand metrics as a time-series
- For any given metric, what was the count for:
  - Yesterday?
  - Past week?
  - Past month?
  - Every Tuesday this year?
  - 2021?

Outline

So what else can the registry see?

- Headers
- Time series information
- Identifiers (IP address, auth, etc)

Headers in downloads give important clues

Headers per request

```
X-Request-ID: <request id>
X-Forwarded-For: <ip>
authorization: Bearer <token>
user-agent: Helm/3.13.3
```

Headers in downloads give important clues

Headers per request

```
X-Request-ID: <request id>
X-Forwarded-For: <ip>
authorization: Bearer <token>
user-agent: Helm/3.13.3
```

This info can tell us

- A notion of uniqueness(!!)
- IP request metadata
- Client, container runtime, etc.
- Is this a production deployment?
- Platform

Headers are rich in information

A notion of uniqueness

You may have had 1000 downloads today but from only 5 distinct sources

IP request metadata

- Where are your users distributed geographically?
- Are your downloads coming from companies or individuals? Which companies?
- Which clouds?

Platform

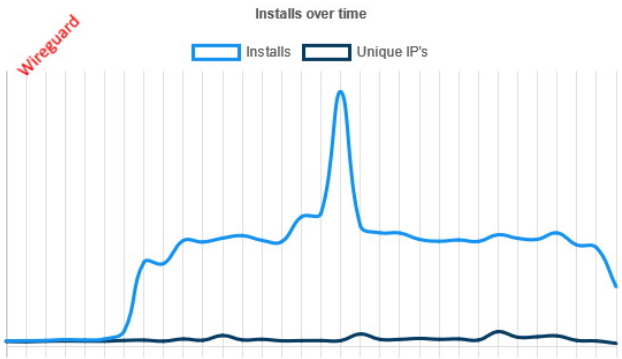
- Client / runtime versions. Is this happening in a k8s cluster or on someone's laptop?
  - If lots old versions are involved, your download counts are likely inflated!
- What is the breakdown of host OS? Architecture?

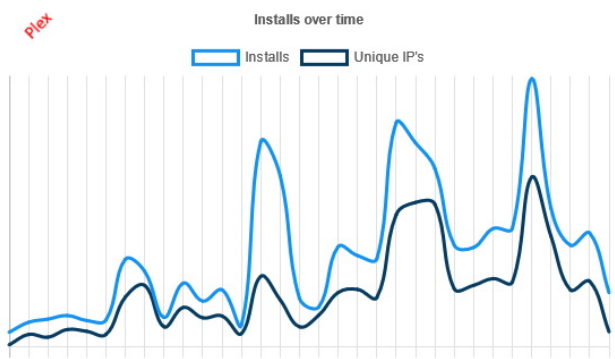
Uniques can be extremely useful

> *Two users are responsible for 73,000 pulls between them, with the next 10 being responsible for 55,000 between them. Almost half of our pulls through Scarf can be attributed to 20 users with misconfigured or overly aggressive deployment/update services*

- *LinuxServer.io Blog*

link - `https://www.linuxserver.io/blog/unravelling-some-stats`





So what else can the registry see?

Time series of requests

Time	Origin ID	Request Type	Path
12:00	abc	`HEAD`	`/v2/image-name/manifests/latest`
12:00	abc	`GET`	`/v2/image-name/manifests/latest`
12:05	abc	`HEAD`	`/v2/image-name/manifests/latest`
12:10	abc	`HEAD`	`/v2/image-name/manifests/latest`
12:15	abc	`HEAD`	`/v2/image-name/manifests/latest`
12:20	abc	`HEAD`	`/v2/image-name/manifests/latest`

This info can tell us

- Invocations of the container vs downloads of the container
- Gives clues to activity / behavior

Time series data tells us about usage

Consider this access patterns

Time	Origin ID	Request Type	Path
12:00	abc	`HEAD`	`/v2/image-name/manifests/latest`
12:00	abc	`GET`	`/v2/image-name/manifests/latest`
12:05	abc	`HEAD`	`/v2/image-name/manifests/latest`
12:10	abc	`HEAD`	`/v2/image-name/manifests/latest`
12:15	abc	`HEAD`	`/v2/image-name/manifests/latest`
12:20	abc	`HEAD`	`/v2/image-name/manifests/latest`

Time series data tells us about usage

Consider this access patterns

Time	Origin ID	Request Type	Path
12:00	abc	`HEAD`	`/v2/image-name/manifests/latest`
12:00	abc	`GET`	`/v2/image-name/manifests/latest`
12:05	abc	`HEAD`	`/v2/image-name/manifests/latest`
12:10	abc	`HEAD`	`/v2/image-name/manifests/latest`
12:15	abc	`HEAD`	`/v2/image-name/manifests/latest`
12:20	abc	`HEAD`	`/v2/image-name/manifests/latest`

Relevant info

- Highly regular intervals, polling for latest version

Possible explanations

- Production deployment
- Internal tooling deployment

Time series data tells us about usage

Versus this one

Time	Origin ID	Request Type	Path
12:00	abc	`HEAD`	`/v2/image-name/manifests/latest`
12:00	abc	`GET`	`/v2/image-name/manifests/latest`
13:09	abc	`HEAD`	`/v2/image-name/manifests/latest`
13:09	abc	`HEAD`	`/v2/image-name/manifests/latest`
13:10	abc	`HEAD`	`/v2/image-name/manifests/latest`
09:01	abc	`HEAD`	`/v2/image-name/manifests/1.0.1`
09:01	abc	`GET`	`/v2/image-name/manifests/1.0.1`
09:03	abc	`HEAD`	`/v2/image-name/manifests/latest`
09:10	abc	`HEAD`	`/v2/image-name/manifests/latest`

Time series data tells us about usage

Versus this one

Time	Origin ID	Request Type	Path
12:00	abc	HEAD	/v2/image-name/manifests/latest
12:00	abc	GET	/v2/image-name/manifests/latest
13:09	abc	HEAD	/v2/image-name/manifests/latest
13:09	abc	HEAD	/v2/image-name/manifests/latest
13:10	abc	HEAD	/v2/image-name/manifests/latest
09:01	abc	HEAD	/v2/image-name/manifests/1.0.1
09:01	abc	GET	/v2/image-name/manifests/1.0.1
09:03	abc	HEAD	/v2/image-name/manifests/latest
09:10	abc	HEAD	/v2/image-name/manifests/latest

Relevant info

- Irregular intervals
- Multiple versions

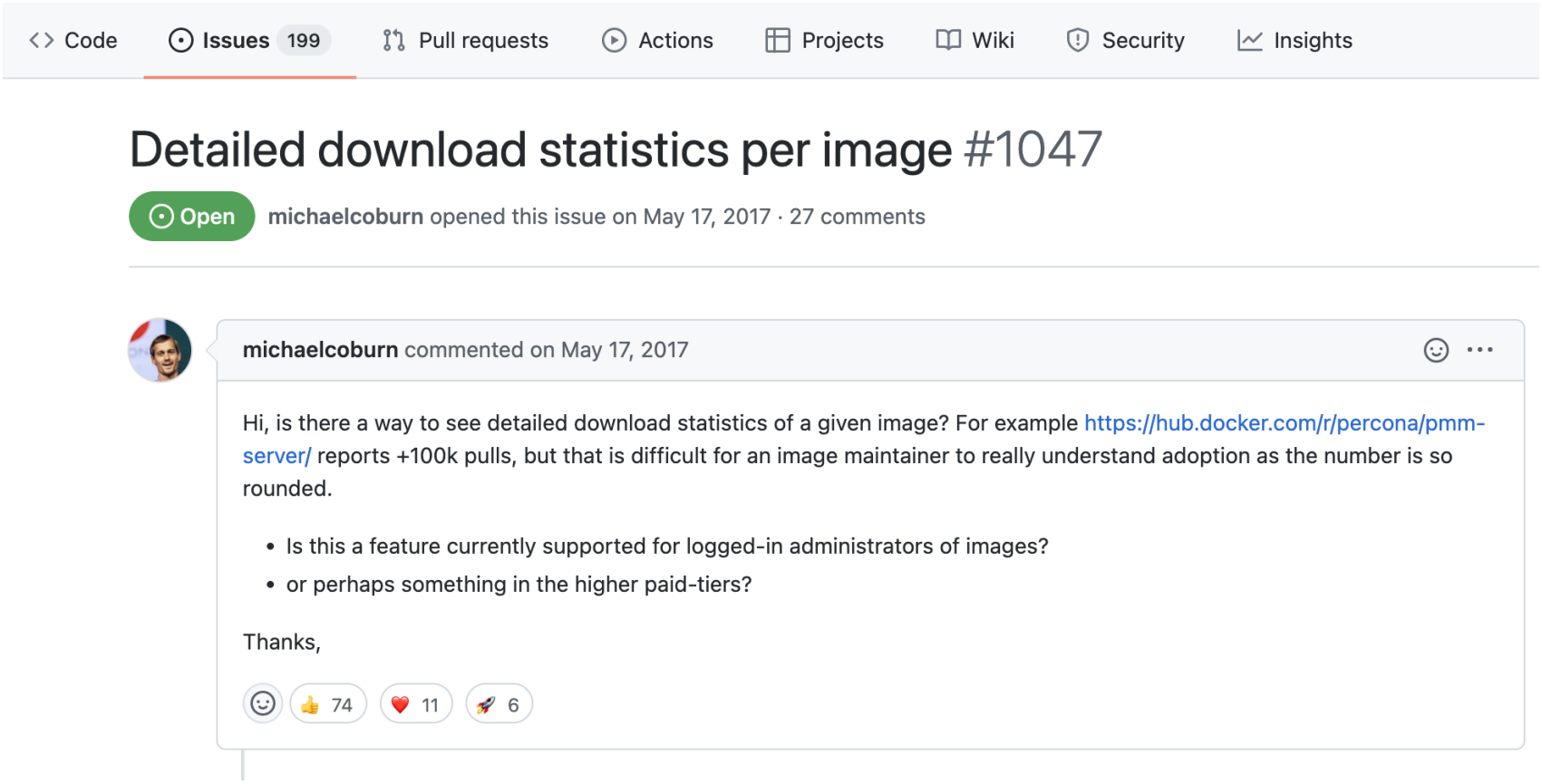
Possible explanations

- Local development

Outline

Convince your registry to give it you

Let me know how it goes!



Host a registry

```
$ docker pull yourdomain.com/your-image
```

Pros

- Open source solutions (eg distribution)
- Distribute from your own domain
- Full access (publishing, data handling, insights, etc)

Cons

- Bandwidth is expensive
- Availability and performance are on you
  - *How long will it take your us-west-2 machine to stream a 1GB artifact to a user in Mumbai?*

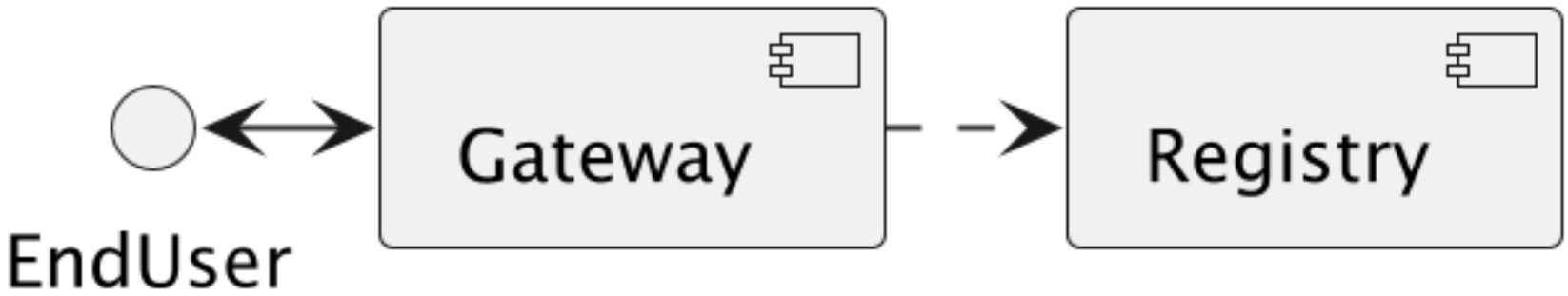
Registry Gateway

Idea

Put a service in front of the registry that:

- Passes traffic transparently to the registry that hosts the artifact via a redirect
- Processes traffic to process pull data

```
$ docker pull yourdomain.com/your-image
```



Registry Gateway

Pros

- Can access all request data
- Lightweight service - redirection can be very dumb
- Robust to API changes from the the client/registry
- Simply(*) redirecting rather than proxying means minimal overhead (bandwidth and speed)
- Decoupling from registry
- Distribute from your own domain

Cons

- Added complexity
  - Failure point
  - Performance choke point

Simple!(*)

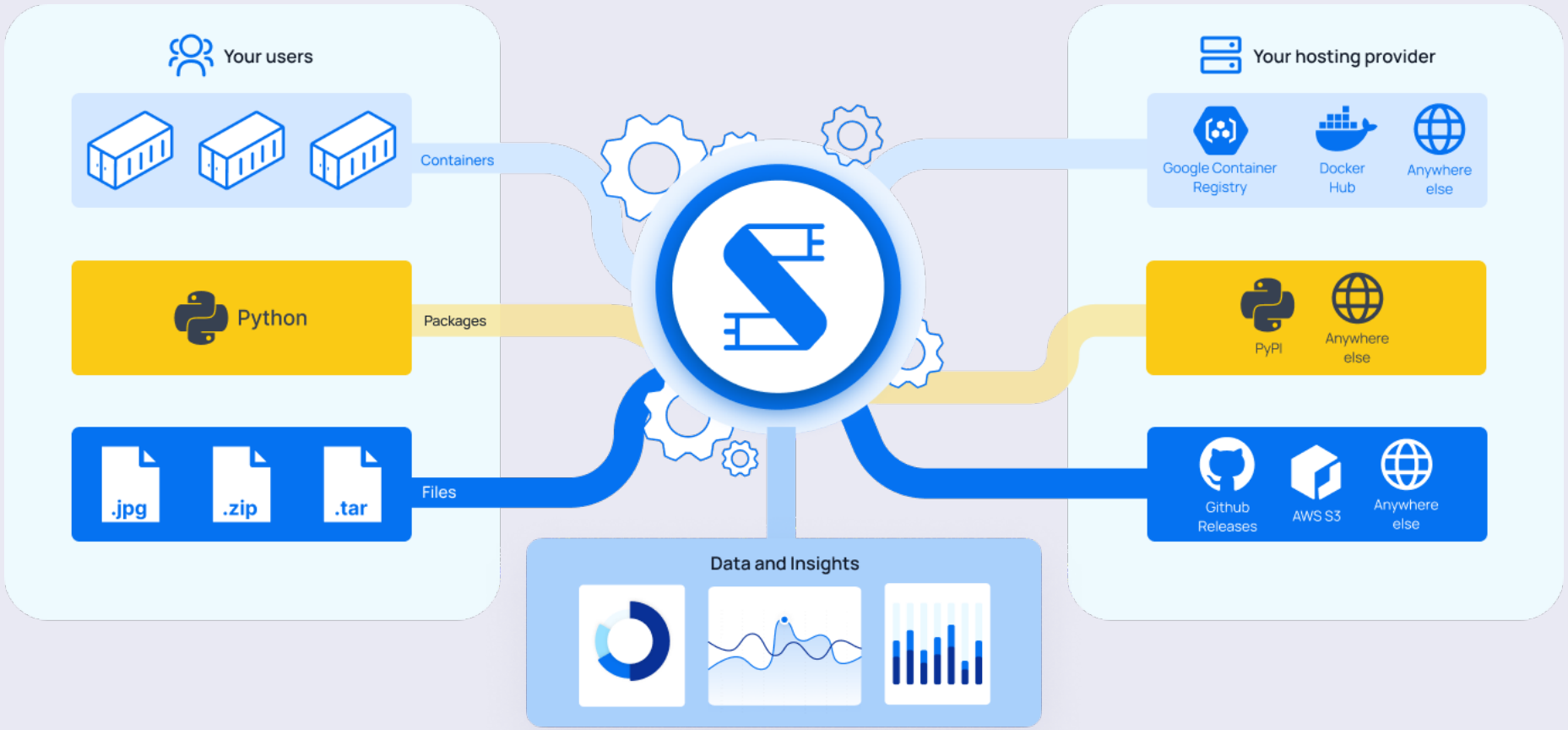
```
server {
  server_name a.domain.com
  listen 443;
  rewrite (.*) https://registry-1.docker.io$1 permanent;
}
```

. . . Almost

- Gateway still needs to be available and fast globally
- Can't actually always redirect, some clients will choke :'(

How Scarf built its artifact registry gateway

Scarf Gateway



How Scarf built its artifact registry gateway

Open source: `https://github.com/scarf-sh/gateway`

Phase 1

A general recommended approach to anyone wanting to get started building their own

- Nginx
  - Send access logs to storage (we were using AWS Cloudwatch)
  - Lua for any custom business logic you might want, eg reading configs from Redis
- Process logs asynchronously to generate analytics & insights

Phase 2

- Server as hand-written Haskell code
- Configuration in-memory
- Send access logs to time series storage, eg Kafka
- `distribution` as a pull-through-cache when we are forced to proxy

Data privacy

This can be done while still completely preserving end-user privacy.

- Depending on how you store and process this data, you may or may not run into compliance considerations like GDPR
- Recommendations:
  - Don't touch PII you don't need
  - Delete it once you are done processing it
  - Leverage 3rd parties to handle it on your behalf
  - Consult legal counsel

Other benefits of the gateway approach

- Distribute from your own domain, not someone else's
- Ability to switch registries on-the-fly without breaking anything downstream.
    - Dual publishing can keep your artifacts online when primary registry goes down

Notable challenges

- Easy to build, harder to scale
  - Multi-region availability, redundancy, etc is where the real complexity lives
- Proxying as little as possible
- Many competing container runtimes / clients -> edge-case bugs

Tying it together

- Registry data can be useful!
- Your current registry provider doesn't provide access to pull data, but there are still ways to get to it.
- Registry gateways can be a reasonable option

Outline

Scarf + Linux Foundation Partnership

All LF projects receive:

- Free Scarf licenses for your entire team
- Unlimited data history retention
- Tracking for all of your packages, containers, models, other artifacts
  - *also your website, OSS docs, and any other content!*
- Hands-on support from Scarf team

Follow the lead of other projects in the LFAI & Data, CNCF, and LF broadly:

- Flyte
- Falco
- Linkerd
- StarRocks
- Cert Manager
- Open Telemetry
- Litmus Chaos
- Janssen
- Emissary Ingress
- Dapr
- Fluent Bit
- KrakenD
- More!

Thank you!

Avi Press

Website	`https://avi.press`
Twitter	@avi_press
GitHub	aviaviavi
LinkedIn	link

Scarf

Website	`https://scarf.sh`
Twitter	scarf-oss
GitHub	scarf-sh
LinkedIn	link